RNE: A Scalable Network Embedding for Billion-scale Recommendation

Jianbin Lin[1], Daixin Wang[1,2], Lu Guan[3], Yin Zhao[3], Binqiang Zhao[3], Jun Zhou[1], Xiaolong Li[1], Yuan Qi[1]

[1] Ant Financial Services Group, Hangzhou, China
[2] Computer Science and Technology, Tsinghua University, Beijing, China
[3] Alibaba Group, Hangzhou, China

Abstract. Nowadays designing a real recommendation system has been a critical problem for both academic and industry. However, due to the huge number of users and items, the diversity and dynamic property of the user interest, how to design a scalable recommendation system, which is able to efficiently produce effective and diverse recommendation results on billion-scale scenarios, is still a challenging and open problem for existing methods. In this paper, given the user-item interaction graph, we propose RNE, a data-efficient Recommendation-based Network Embedding method, to give personalized and diverse items to users. Specifically, we propose a diversity- and dynamics-aware neighbor sampling method for network embedding. On the one hand, the method is able to preserve the local structure between the users and items while modeling the diversity and dynamic property of the user interest to boost the recommendation quality. On the other hand the sampling method can reduce the complexity of the whole method theoretically to make it possible for billion-scale recommendation. We also implement the designed algorithm in a distributed way to further improves its scalability. Experimentally, we deploy RNE on a recommendation scenario of Taobao, the largest E-commerce platform in China, and train it on a billion-scale user-item graph. As is shown on several online metrics on A/B testing, RNE is able to achieve both high-quality and diverse results compared with CF-based methods. We also conduct the offline experiments on Pinterest dataset comparing with several state-of-the-art recommendation methods and network embedding methods. The results demonstrate that our method is able to produce a good result while runs much faster than the baseline methods.

1 Introduction

With the exponential growth of data and information on the Internet, recommendation system plays a critical role in reducing information overload. Recommendation systems are widely deployed on many online services, including E-commerce, social networks and online news systems. How to design an effective recommendation system has been a fundamental problem in both academia and industry.

The key for recommendation system is to model the users' preferences based on their interactions (e.g., clicks and rating) with the items. One of the most popular recommendation methods are known as collaborative filtering (CF) [13]. Its basic idea is to match the users with similar item preferences. Among the various collaborative filtering methods, matrix factorization [11,26] is the mostly used one. However, these matrix

factorization based methods are regarded as the linear methods, which are difficult to model the user-item interactions. Then following works use the deep neural networks to model the user-item relationships [12]. Despite of their success, these CF-based methods only aim to model the direct links, i.e. the first-order relationship between the users and items. However, for a graph, only preserving the first-order relationships between the nodes is not enough to characterize the network structure and thus cannot achieve good performance [5,2].

To preserve the second-order local structure in the networks, network embedding is an effective way [5]. Network embedding aims to embed nodes into a low-dimensional vector space with the goal of capturing the low-order and high-order topological characteristics in graphs [17,20,10,21,25]. Although network embedding is able to incorporate local structures, they mainly target on tasks of common link prediction and node classification. Few of them deal with the task of recommendation and thus they seldom consider some specific properties of recommendation, which makes them difficult to get a good performance on recommendation. Last but not least, few of these methods can be applied to the billion-scale networks.

To extend network embedding to recommendation, we meet three challenges. (1) Diversity of user interest. User interest is always diverse and the diverse recommendation can help users explore new items of interest. Therefore, diversity has been a very important measure to evaluate the recommendation system [1]. However, existing network embedding methods seldom consider the diversity. (2) Dynamic changes of user interest. User's preference is dynamic and how to model such a dynamic property is another challenge. (3) Scalability of recommendation system. Existing recommendation scenario often has a huge number of users and items, which is a serious problem with a scale beyond most of existing network embedding methods.

To address these challenges, we propose RNE, a scalable Recommendation-based Network Embedding method. In our method, when discovering the local structure of a user, we will not model all the items the user clicked. Instead, we propose a sampling method, which considers the diversity and dynamics of the user interest, to sample a portion of the items the user has clicked as the user's neighbors. In this way, the sampling method not only can incorporate the important properties of recommendation, i.e. the diversity and dynamics, to improve the recommendation accuracy, but also reduce the computational complexity of the algorithm. Furthermore, we deploy the algorithm on a recommendation system based on the Parameter Server to do distributed and parallel computing, which further facilitates the large-scale training available.

In summary, the contributions of the paper can be listed as follows:

- We propose a network-embedding-based recommendation method, named RNE. When modeling the local structures between the users and items, our method is able to incorporate the dynamics and diversity of the user interest to produce more accurate and diverse recommendation results.
- We implement our recommendation algorithm in a distributed way based on parameter server, which jointly makes the system available for billion-scale recommendation.
- Experimentally, we deploy the whole system on a recommendation scenario of Taobao. Online A/B tests demonstrate that our method is able to achieve more

accurate results compared with CF and greatly improve the diversity of the recommendation results. Experiments on offline dataset Pinterest also demonstrate the quality of our method.

Table 1. Multifaceted Comparisons between different methods

Method	Local-structure Preserving	Diversity	Billion-scale	Complexity
GMF-CF/MLP-CF/NCF	$\times$	$\times$	$\times$	$O(\|E\|)$
LINE/node2vec	$\surd$	$\times$	$\times$	$O(\|E\|)$
RNE	$\surd$	$\surd$	$\surd$	$O(\|V\|)$

2 Related Work

2.1 Collaborative Filtering

Recommendation algorithms and systems are well-investigated research fields. In our work, we are only given the user-item interaction data. Therefore, we mainly introduce the CF-based recommendation methods and omit the discussions of content-based recommendation methods and the hybrid recommendation methods.

Collaborative Filtering exploits the interaction graph between the users and items to give the recommendation lists to users. Its basic idea is to match the users which have similar item preferences. Earlier CF methods mainly use the matrix factorization on the user-item matrices to obtain the latent user factors and item factors [6,11,18]. The user factors and item factors together aim to reconstruct the original user-item matrices. However, the matrix factorization is just the linear-based methods, which is difficult to capture the user-item relationships. To overcome such a drawback, following works use the deep neural networks to perform collaborative filtering [22,19]. However, most of the CF-based methods only aim to model the pairwise relationships between the user and item but omit their local structures. And many graph-based works have demonstrate that local structures like second-order relationships are very important for capturing graph structures [20]. In this way, existing CF-based methods are sub-optimal for capturing the relationships between user and items.

2.2 Network Representation Learning

Network embedding has been demonstrated as an effective methods for modeling local and global structures of a graph. It aims to learn a low-dimensional vector-representation for each node. DeepWalk [17] and Node2vec [10] propose to use the random walk and skip-gram to learn the node representations. LINE [20] and SDNE [21] propose explicit objective functions for preserving first- and second-order proximity. Some further works [16,2] use the matrix factorization to factorize high-order relation matrix. Aforementioned methods are designed for homogeneous networks. Then some following embedding methods for heterogeneous networks are proposed, like Metapath2vec [7] , HNE [4], BiNE [8] and EOE [24]. Some works further focuses on knowledge graph embedding [23]. Although these network embedding methods are able to preserve the local structures of the vertices, most of them are not specifically designed for the task of recommendation. They do not consider some specific properties of the recommendation tasks like the diversity and dynamic changes of user interest, the scalability issues of large-scale recommendation tasks. Therefore, how to propose an effective network embedding method for billion-scale recommendation is still an open problem.

In summary, we compare our method and the related works in Table 1. Our method is specifically designed for the recommendation scenario and thus consider some specific properties. Furthermore, the proposed method is very scalable and thus can apply to billion-scale recommendations.

3 The Methodology

In our scenario, we have a large number of users and items. Each user may have different ways to interact with the items. For example, the user may view the items, collect the items or buy the items. In this way, we can build the user-item interaction graph, formally formulated as $G = (\mathcal{U}, \mathcal{I}, E)$. Here $\mathcal{U}$ denotes the total of n users and $\mathcal{I}$ denotes the total of T items. $\mathcal{U} \cup \mathcal{I}$ denotes the set of nodes in G. If a user $u \in \mathcal{U}$ views, collects or buys an item $i \in \mathcal{I}$, there is an edge E_{ui} between u and i. We use $E(v), v \in \mathcal{U} \cup \mathcal{I}$ to denote the edges connected to the node v. We assume that G is connected. The recommendation problem is that given a user u, we hope to recommend some personalized items to the user based on his previous behavior.

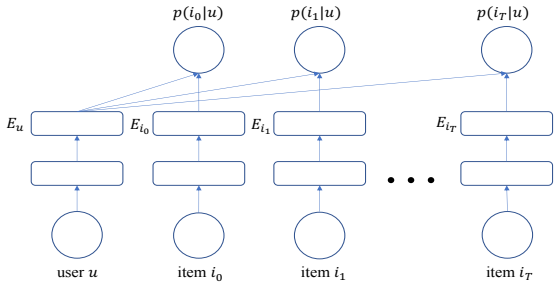

Fig. 1. The framework of RNE.

3.1 Network Embedding for Recommendation

Given the user-item interaction graph $G = (\mathcal{U}, \mathcal{I}, E)$, we aim to map each user and item to a common low-dimensional latent space, where user u can be embedded as $\mathbf{E}^u_{\mathcal{U}} \in R^d$ and item i can be embedded as $\mathbf{E}^i_{\mathcal{I}} \in R^d$. Then with the embeddings for each user and item, we can retrieve the similar items for the user as his recommendation results.

To achieve this, we propose our method, whose framework can be shown in Figure 1. It consists of the embedding-lookup layer, embedding layer and softmax layer. The embedding-lookup layer helps us obtain the embeddings for the users and items. The embedding layer and softmax layer together model the interactions between the users and items to update the embedding-lookup layer. Then we introduce the designed loss functions to update the embeddings.

We first consider how to model the local structure of a user in the given user-item graph. In the original space, the empirical distributions given a user can be defined as:

$$\hat{p}(i|u) = \frac{w_{ui}}{d_u}, \tag{1}$$

where w_{ui} is the weight between user u and item i and d_u is the degree of user u.

Then we hope to estimate the local structure of a user in the embedding space. Word2vec [15] inspires us to use the inner product between two vertices to model their

interactions. Then in our work, given a user u, we define the probability of item i generated by user u as:

$$p(i|u) = \frac{exp(\mathbf{E}_{\mathcal{U}}^{u^T}\mathbf{E}_{\mathcal{I}}^{i})}{\sum_{j=1}^{|I|} exp(\mathbf{E}_{\mathcal{U}}^{u^T}\mathbf{E}_{\mathcal{I}}^{j})}, \tag{2}$$

where T means the transpose of a matrix.

Eqn. 2 is a softmax-like loss function, which defines the conditional distributions $p(\cdot|u)$ of user u over its neighborhoods, i.e. the entire item set, in the embedding space.

With the empirical distributions on the original network and reconstructed distributions on the embedding space, we can learn the embedding by making the defined probability $p(\cdot|u)$ specified by the low-dimensional representations be close to the empirical distributions $\hat{p}(\cdot|u)$. We use the KL-divergence to measure the distance between the distributions. Then the loss functions can be defined as:

$$L = \sum_{u\in\mathcal{U}} \lambda_u KL(\hat{p}(\cdot|u), p(\cdot|u)) \propto -\sum_{(u,i)\in E} w_{ui} logp(i|u), \tag{3}$$

where λ_u denotes the prestige of user u and we set $\lambda_u = d_u$.

Minimizing Eq. 3 will make the vertices with similar neighbors similar to each other. Therefore, it can not only model the observed links on the graph, but also preserve the local structures for each node.

3.2 Recommendation-based Sub-sampling

However, aforementioned network embedding meets two challenges for large-scale recommendation: (1) Minimizing Eqn. 3 is time-consuming since for each edge it needs to run over the entire set of the items when evaluating $p(i|u)$. In this way, the whole complexity is $O(|E||\mathcal{I}|)$, which is unbearable for real recommendation systems. (2) Minimizing Eqn. 3 only considers the topology of the graph. It does not consider the diversity and the time decay of the user interest, which are very important properties for recommendation systems.

To reduce the complexity, we first adopt negative sampling as many methods do [20]. For each positive edge (u, i), we will sample some negative edges according to predefined distributions P_{ui}. By performing negative sampling, the objective function for each edge (u, i) can be reformulated as:

$$L_{ui} = log(\sigma(\mathbf{E}_{\mathcal{U}}^{u^T}\mathbf{E}_{\mathcal{I}}^{i})) + \sum_{j=1}^{k} E_{i_j\sim P_u}(log(\sigma(\mathbf{E}_{\mathcal{U}}^{u^T}E_{\mathcal{I}}^{i_j}))), \tag{4}$$

where k is the number of negative samples for each user-item pair, $P_{ui} \propto d_i^{3/4}$.

Although negative sampling can reduce the time complexity from $O(|E||\mathcal{I}|)$ to $O(k|E|)$, for billion-scale recommendation, a complexity linear to the number of edges is still a great challenge.

To further reduce the complexity, we only select a portion of the items the user has clicked to obtain his behavior sequence. Then the question comes to how to select the items to effectively represent the user's interest. Here, we mainly consider two properties specified for recommendation. (1) The diversity of user interest: User interest is always diverse. A user will always focus on the items of more than one cluster. (2) The

time decay of user interest: User interest is always dynamic. More recent user behavior is more reliable to reflect the recent user interest. Therefore, we should more focus on recent user behavior. Based on these two considerations, we define the selection probability for each user-item pair (u, i) as follows:

$$p(u, i) = 0.999^{t_i} * click(u, c_i)^{\gamma}, \tag{5}$$

where t_i is the hours of the item i from the most recent item, c_i is the cluster index of item i and $click(u, c_i) = \sum_{j \in c_i} w_{uj}$, γ is set to -0.2. Then for each user, we will sample m samples according to the defined probability in Eqn. 5 to represent his behavior sequence. Then in this way, the complexity can be reduced from $O(k|E|)$ to $O(km|\mathcal{U}|)$, which is linear to the number of nodes.

In summary, on the one hand, if a user more recently shows the interest to an item, the item should have a larger probability to be sampled. On the other hand, the method is prone to sample the items of the clusters clicked less times by the user. In this way, our method may cover more clusters to ensure the diversity. Therefore, such a sampling strategy can simultaneously model the diversity and time decay of the user interest Furthermore, with the sampling strategy, we do not need to model all the edges in one iteration but instead for each user we only model a portion of its preferred items as the user's behavior sequence. It significantly reduce the time complexity.

3.3 Implementation

In this section, we will introduce the technical implementation of the proposed RNE. The whole process can be divided into two phases: offline model training and online retrieval. The proposed RNE is implemented and deployed on KunPeng platform [3]. This section will describe them in detail.

Off-line Model Training To train the proposed RNE, we utilize the Stochastic Gradient Descent (SGD) on the loss function of Eqn. 4 to update the node embeddings. In detail, we use E_{pos} to denote all the positive edges sampled by the method we proposed before. Then for each $(u, i) \in E_{pos}$, we can update their embeddings as follows:

$$\mathbf{E}^u_{\mathcal{U}} = \mathbf{E}^u_{\mathcal{U}} + \lambda\{ \sum_{z \in \{i\} \cup N^k_{neg}(u)} [I(z, u) - \sigma({\mathbf{E}^u_{\mathcal{U}}}^T \mathbf{E}^z_{\mathcal{U}})] \cdot \mathbf{E}^z_{\mathcal{U}}\}, \tag{6}$$

where $I(a, b)$ is the indicator function that if $a = b$, $I(a, b) = 1$, otherwise $I(a, b) = 0$. $N^k_{neg}(i)$ is the negative neighborhoods of vertex i. λ denotes the learning rate. Similarly, we can update embedding $E^i_{\mathcal{I}}$ for an item i in a similar way, which we will not discussed more.

From Eqn. 6, when given a positive edge, we can update their embeddings. Then we will go over all the pair of positive edges for several iterations to update their embeddings. The whole algorithm can be summarized in Alg. 1.

From Alg. 1, we find that the learning process from Line 4 to Line 6 is independent for different edges, which inspires us to use some parallelization mechanism to implement it. Then we deploy the whole algorithm on the parameter server, which implements a data-parallelization mechanism. In detail, from Eqn. 6 we find that to update a node's embedding, we only need to know the node's previous embeddings, the node's neighborhoods and their embeddings. Therefore, we can resort to parameter server to

Algorithm 1 Training Algorithm for RNE

Input: $G = (\mathcal{U}, \mathcal{I}, E)$
Output: $\mathbf{E}_u$, $\mathbf{E}_I$

```
Initializing E_u and E_I.
while not converged do
    Construct the positive edge set S_pos according to G = (U, I, E) and Eq. 5.
    for all (u, i) ∈ S_pos do
        Construct the negative set N^k_neg(u).
        Update E_u and E_i according to Eq. 6.
    end for
end while
```

implement such a process in a parallelized way. The main workflow of the system is built as follows: (1) In each iteration, the server will assign each worker a subset of the vertices of the graph G. (2) Each worker will pull the assigned vertices from the server and calculate the positive and negative neighborhoods for the assigned vertices. Then with positive and negative sets, each worker can update the embeddings of the assigned vertices according to Eqn 6. (3) After updating, each worker will push his assigned vertices' embddings to the server. Such a training process will be iterated several times.

Online Efficient Nearest Neighbor Search For online recommendation, we use the nearest neighbor search on the learned embedding space to make recommendations. That is, given a query user u, we can recommend items whose embeddings are the K-nearest-neighbors (K-nn) of the query user's embedding E_u. To achieve the K-nn search, we use the Faiss library [14] which is an efficient implementation for state-of-the-art product-quantization methods. Given that RNE is trained offline and all the user and item embeddings are computed via Parameter Server and saved in database, the efficient K-nn search enables the system to recommend items online.

4 Experiments

The goal of RNE is to produce high-quality and scalable recommendations for real-world systems. Therefore, we conduct comprehensive experiments in two ways: Online A/B tests and Offline experiments.

4.1 Datasets

We use two real-world datasets, i.e. Ali-mobile taobao and Pinterest in this paper.

- Ali-mobile taobao: It is a mobile recommendation scenario deployed on Taobao, the largest E-commerce platform in China. The dataset is extremely large. It has about 1 billion users, tens of million items and a total of about one hundred billion edges. Each edge denotes whether the user has clicked the products. We deploy our algorithm on the service to do online A/B test to evaluate our method.
- Pinterest: The dataset is an image recommendation dataset constructed by [9]. We filter the users which have very few interactions with the items and only retain the users which have more than 20 interactions. After the pre-processing, the dataset consists of 50 thousand users, 10 thousand items and 1.5 million user-item edges. Each edge denotes whether the user has pinned the items.

4.2 Online A/B tests

The ultimate goal of the recommendation system is to lift the user's interest in the items. Therefore, we perform random A/B experiments on Ali-mobile taobao to demonstrate this, where a random set of users obtain the recommendation results of RNE and another obtain the results of CF-based methods. Any difference in the engagement of the items between the two groups can truly reflect the recommendation quality of two methods. Note that here we only use one baseline because deploying many methods online to do A/B tests will cost a lot of resources. And the reason why we choose CF is that it is well investigated for recommendation and existing network embedding methods cannot scale to billion-scale dataset. For more comparisons with state-of-the-art methods, we do offline experiment, which we will introduce in detail later.

We use the following six metrics to measure the recommendation quality.

- AVD (Averaged View Depth): The metric denotes how deep a user views the page. It measures the recommendation quality.
- ACN (Averaged Click Number): The metric measures the number of clicks on the items for each user in average. It measures the recommendation quality.
- P-CTR (Page Click-through Rate): for a page p, $pctr = \frac{\#click-throughs(p)}{\#impressions(p)} \times 100\%$. It measures the recommendation quality.
- U-CTR (User Click-through Rate): for a user u, $uctr = \frac{\#click-throughs(u)}{\#impressions(u)} \times 100\%$. It measures the recommendation quality.
- Re-C (Recommended number of clusters): The averaged number of clusters recommended to users. The clusters are obtained by using our clustering algorithm. The metric measures the recommendation diversity.
- CK-C (Clicked number of clusters): The averaged number of clusters clicked by users. It measures both the recommendation quality and the diversity.

Table 2. Performance of online A/B tests on Ali-mobile taobao

Metrics	AVD	ACN	P-CTR	U-CTR	Re-C	CK-C
Ali-mobile taobao	9.54%	13.21%	4.99%	1.12%	20.49%	16.32%

Table 2 summarizes the lift in engagement of items recommended by RNE compared with CF-based methods in controlled A/B experiments. From Table 2, we have the following observations and analysis:

- We find that RNE can achieve a significant improvement in terms of AVD and ACN over the CF. It indicates by using the results of RNE, users are more willing to go deeper to view more items and click more items, which indirectly demonstrates the ranking quality of RNE.
- In terms of the two CTR metrics, a popular and well-accepted metric to evaluate the recommendation quality, our proposed method also achieves a better result than CF. It further demonstrates that RNE is able to produce personalized items for users. The reason for a better recommendation quality is twofold. (1) Our method is able to capture the local structures between the users and items. (2) Our method considers the dynamic change of the user's interest.

– We find that RNE achieves a higher Re-C compared with CF, which indicates that our recommended results are from more clusters. The reason is that our proposed method incorporates the diversity issue into the model design.
– More importantly, our method achieves a higher CK-C than CF, which demonstrates that not only our method can produce more diverse recommendations, but also the users are willing to click these diverse recommended items. It indicates that our method is able to improve the recommendation quality while improving the recommendation diversity.
– Under the billion-scale scenario, RNE can be deployed online and still obtain good results, which demonstrates the superiority of our method.

4.3 Showcase

(a) (b)

Fig. 2. Real showcase on Ali-mobile taobao: Given an item (in red box), searching for the nearest 8 items (in blue box) using the embeddings learned by RNE.

In this section, we give some showcase to see some intuitions regarding the embeddings we learn. After the learning process of RNE, all the items will have embeddings. Then in this experiment, given a query item's embedding, we aim to find the most similar 8 items whose embeddings have the smallest distance with the query. Then we display both the query image and the recommended images in Figure 2.

In Figure 2(a), the query item is a princess-style educational toy for girls. When we look at the returned results, these images belong to different categories with the query, like plasticine and origami. But all of them are for fun and a majority of them are also princess-style. It demonstrates that our method is able to find more categories of items but retain the primary style of the item. In Figure 2(b), the query item is a woman sweatshirt of the brand of Peacebird. The returned images are all coat, sweater or sweatshirt of the brand of Peacebird or Only. Similarly, the returned images and the query image are all casual style but belong to different fine-grained categories. Moreover, actually the brand of Peacebird and Only have very similar styles and our method can learn their inherit relationships.

In summary, in our method, we do not have the item features and the direct relationships between items. We are only given the user-item interaction graph. Although in this case, our method still can model the item relationships by using the user behaviors

as the bridge. It demonstrates that by using the network embedding method we propose, the learned embeddings can capture the local relationships between the entities.

4.4 Offline Experiment

Table 3. Recommendation Performance on Pinterest.

Method	HR				NDCG				RR			
	Top5	Top10	Top50	Top100	Top5	Top10	Top50	Top100	Top5	Top10	Top50	Top100
GMF	0.501	0.678	0.9	0.97	0.332	0.386	0.425	0.434	0.276	0.292	0.301	0.307
MLP	0.504	0.679	0.908	**0.99**	0.341	0.385	0.426	0.436	0.275	0.295	0.302	0.309
NCF	0.529	0.688	0.912	**0.99**	0.35	0.405	0.443	0.454	0.298	0.32	0.343	0.341
LINE	**0.536**	**0.7**	0.91	**0.99**	0.353	0.409	0.448	0.455	0.297	0.325	0.334	0.335
node2vec	0.527	0.691	**0.93**	**0.99**	0.355	**0.411**	**0.459**	**0.46**	**0.302**	**0.329**	0.341	0.342
RNE	0.531	0.695	0.925	**0.99**	**0.356**	0.41	0.45	0.457	0.3	0.327	**0.345**	**0.348**

To compare more baselines to get comprehensive results, we conduct the offline experiments on the dataset of Pinterest. We randomly sample 90% user-item pairs as the training set and the rest as the testset. For training set, we use 9-fold cross-validation to tune the parameters for all the methods. Note that in this dataset, we do not have the cluster and time information for the item. So we uniformly sample the items to do training. To evaluate the performance, we use the following three metrics: Normalized Discounted Cumulative Gain (NDCG), Mean Reciprocal Rank (MRR) and Hit Rate (HR). NDCG and MRR will consider the rank of the hit and will assign higher scores to hits at top ranks. While HR will only evaluate whether the test items are hit or not. We calculate all the metrics for the test users and report the average score.

We first use the advanced CF-based methods GMF, MLP and NCF [12] as baseline methods. We perform the same process of parameter search as the work [12] did to select the optimal parameters. For network embedding methods, since we only have the graph topology, in this case LINE [20] and node2vec [10] are state-of-the-art network embedding methods, so we choose them as the baselines. For LINE, we use LINE$_{1st+2nd}$ with the default parameter settings. For node2vec, we also use the default settings except for the bias parameters p, q, which we conduct the grid search from $\{0.5, 1\}$. The embedding dimension of them is all set as 128.

The results are shown in Table 3. From Table. 3, we find that RNE achieves a better performance than all the CF-based methods. The reason is that RNE is able to capture the local structure of each user while CF-based methods only focus on the direct links the user has clicked. It demonstrates that capturing the local structures on the user-item graph is important for recommendation. LINE, node2vec and RNE achieve similar performance in different evaluation metrics and scenarios. But our method runs much faster than node2vec and LINE, which will be discussed later. Therefore, RNE is a better balance between accuracy and efficiency.

Now we discuss the training time of LINE, node2vec and RNE. For a fair comparison, we do not use the distributed strategy for RNE. From Figure 3, we find that RNE can boost the running time over LINE and node2vec. Specifically, when the training edges increase from 0.15 million to 1.5 million, the running time improvement of RNE

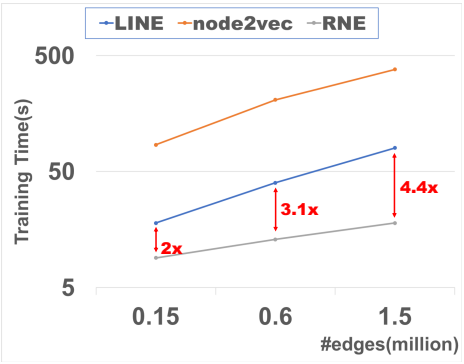

Fig. 3. Time comparisons on Pinterest dataset. We change the number of edges to be trained and report the training time for each network embedding method.

compared with LINE will be larger and larger, from 2x to 4.4x. When the edges continuously increase to the billion-scale dataset like the Ali-mobile taobao dataset, it is difficult for LINE and node2vec to obtain the results. But RNE can still obtain a good result. The reasons why our method can scale to billion-scale dataset are twofold: (1) The proposed sampling method avoids us running over all the edges in the graph. (2) Our method can be deployed on distributed system for parallel computations.

In summary, RNE has a good scalability, which is much more efficiency than baseline methods and can scale to billion-scale recommendation scenario, meanwhile RNE do not sacrifice its recommendation accuracy.

5 Conclusion

In this paper, we propose a novel network embedding method named RNE for scalable recommendation. The proposed network embedding method is able to capture the local structures on the user-item graph to achieve a better recommendation quality. Specifically, to consider the specific properties for recommendation, i.e the diversity and time-decay of user interest, we design a sampling method for embedding process to incorporate these properties. And the sampling method also guarantees the scalability of the proposed method while almost preserving the recommendation quality. We also deploy our algorithm on parameter server to make it available for large-scale recommendation. Experimental results on online A/B tests and offline experiments all demonstrate the superiority of the proposed method.

For the future work, we may consider the user and item features, which can further address the sparsity and cold-start problem. We also want to analyze the role of features and topology structures for recommendation.

6 Acknowledgement

We would like to thank all the colleagues of our team and all the members of our cooperative team: the search engine team in Alibaba. They provide many helpful comments for the paper. We also would like to thank the support of the Initiative Postdocs Supporting Program and the valuable comments provided by all the reviewers.

References

1. Adomavicius, G., Kwon, Y.O.: Improving aggregate recommendation diversity using ranking-based techniques. TKDE **24**(5), 896–911 (2012)
2. Cao, S., Lu, W., Xu, Q.: Grarep: Learning graph representations with global structural information. In: CIKM. pp. 891–900 (2015)
3. J. Zhou, X. Li, P. Zhao, C. Chen, L. Li, X. Yang, Q. Cui, J. Yu, X. Chen, Y. Ding *et al.*, "Kunpeng: Parameter server based distributed learning systems and its applications in alibaba and ant financial," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 1693–1702.
4. Chang, S., Han, W., Tang, J., Qi, G.J., Aggarwal, C.C., Huang, T.S.: Heterogeneous network embedding via deep architectures. In: SIGKDD. pp. 119–128. ACM (2015)
5. Cui, P., Wang, X., Pei, J., Zhu, W.: A survey on network embedding. TKDE (2018)
6. Deshpande, M., Karypis, G.: Item-based top- n recommendation algorithms. Acm Trans.inf.syst **22**(1), 143–177 (2004)
7. Dong, Y., Chawla, N.V., Swami, A.: metapath2vec: Scalable representation learning for heterogeneous networks. In: SIGKDD. pp. 135–144. ACM (2017)
8. Gao, M., Chen, L., He, X., Zhou, A.: Bine: Bipartite network embedding (2018)
9. Geng, X., Zhang, H., Bian, J., Chua, T.S.: Learning image and user features for recommendation in social networks. In: ICCV. pp. 4274–4282 (2015)
10. Grover, A., Leskovec, J.: node2vec:scalable feature learning for networks. In: SIGKDD. pp. 855–864 (2016)
11. Harvey, M., Carman, M.J., Ruthven, I., Crestani, F.: Bayesian latent variable models for collaborative item rating prediction. In: CIKM. pp. 699–708 (2011)
12. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering pp. 173–182 (2017)
13. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. Acm Transactions on Information Systems **22**(1), 5–53 (2004)
14. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with gpus. arXiv preprint arXiv:1702.08734 (2017)
15. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
16. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: SIGKDD. pp. 1105–1114 (2016)
17. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: SIGKDD. pp. 701–710. ACM (2014)
18. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: International Conference on World Wide Web. pp. 285–295 (2001)
19. Strub, F., Mary, J.: Collaborative filtering with stacked denoising autoencoders and sparse inputs. In: NIPS Workshop on Machine Learning for eCommerce (2015)
20. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: WWW. pp. 1067–1077 (2015)
21. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: SIGKDD. pp. 1225–1234 (2016)
22. Wu, Y., Dubois, C., Zheng, A.X., Ester, M.: Collaborative denoising auto-encoders for top-n recommender systems pp. 153–162 (2016)
23. Xiao, H., Huang, M., Zhu, X.: Transg: A generative model for knowledge graph embedding. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 2316–2325 (2016)

24. Xu, L., Wei, X., Cao, J., Yu, P.S.: Embedding of embedding (eoe): Joint embedding for coupled heterogeneous networks. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. pp. 741–749. ACM (2017)
25. Zhang, Z., Cui, P., Li, H., Wang, X., Zhu, W.: Billion-scale network embedding with iterative random projection. arXiv preprint arXiv:1805.02396 (2018)
26. Zhou, X., Chen, L., Zhang, Y., Qin, D., Cao, L., Huang, G., Wang, C.: Enhancing online video recommendation using social user interactions. VLDBJ **26**(5), 637–656 (2017)