Tripartite Heterogeneous Graph Propagation for Large-scale Social Recommendation

Kyung-Min Kim[1,*], **Donghyun Kwak**[2,*], **Hanock Kwak**[3,*], **Young-Jin Park**[4,*] **Sangkwon Sim**[1], **Jae-Han Cho**[1], **Minkyu Kim**[1], **Jihun Kwon**[4], **Nako Sung**[1], **Jung-Woo Ha**[1*]
{kyungmin.kim.ml,donghyun.kwak}@navercorp.com
hanock.kwak2@linecorp.com
{young.j.park,jaehan.cho,min.kyu.kim,jihun.kwon,andy.sangkwon,nako.sung,jungwoo.ha}@navercorp.com
[1]Clova AI Research, NAVER Corp., [2]Search Solution Inc., [3]LINE Plus Corp. and [4]Naver R&D Center, NAVER Corp.

*Authors contributed equally to this research.

ABSTRACT

Graph Neural Networks (GNNs) have been emerging as a promising method for relational representation including recommender systems. However, various challenging issues of social graphs hinder the practical usage of GNNs for social recommendation, such as their complex noisy connections and high heterogeneity. The oversmoothing of GNNs is a obstacle of GNN-based social recommendation as well. Here we propose a new graph embedding method Heterogeneous Graph Propagation (HGP) to tackle these issues. HGP uses a group-user-item tripartite graph as input to reduce the number of edges and the complexity of paths in a social graph. To solve the oversmoothing issue, HGP embeds nodes under a personalized PageRank based propagation scheme, separately for group-user graph and user-item graph. Node embeddings from each graph are integrated using an attention mechanism. We evaluate our HGP on a large-scale real-world dataset consisting of 1,645,279 nodes and 4,711,208 edges. The experimental results show that HGP outperforms several baselines in terms of AUC and F1-score metrics.

ACM RecSys 2019 Late-breaking Results, 16th-20th September 2019, Copenhagen, Denmark

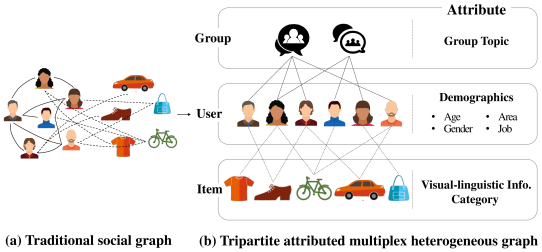

Figure 1: Social graph representation. (a) Traditional user-item graph for social recommendation. (b) Group-user-item tripartite attributed multiplex heterogeneous graph used in our task.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Neural networks**; *Learning latent representations.*

KEYWORDS

Social recommendation, Graph neural networks, Heterogeneous graph embedding, User profiling

INTRODUCTION

Recently, deep learning-based methods have shown promising results in recommender systems. The GNNs are becoming increasingly popular methods to leverage relational information, successfully applied in IT industries. They represent user-item interactions as a user-item graph and compute the similarity between nodes to recommend items to a user. Besides, it is beneficial to use additional user-user interaction information in social recommender systems to alleviate cold-spots in a user-item graph. However, it has not been straightforward to apply GNNs to social recommendation tasks due to the following challenging issues: 1) As the number of user node increases, the number of edges in the user-user graph grows exponentially in general. 2) Tractable social recommendation using GNNs requires proper computational tricks and sampling methods to handle large-scale social graphs. 3) Previous GNNs suffer from the oversmoothing problem, which hinders GNNs from stacking two or more layers. 4) There are two inherently different graphs, i.e., user-user graph and user-item graph. A model has to combine these two graphs coherently.

In this paper, we propose a novel graph configuration and embedding method, Heterogeneous Graph Propagation (HGP), to tackle these four problems. We introduce the concept of a group node that connects groups of related users defined by common social properties to mitigate the complexity of user connections. A user node would belong to multiple group nodes, and there is no edge between user nodes. This configuration reduces computing time and memory by the square of the number of nodes while preserving social attributes and structures. This graph can be formulated as a tripartite attributed multiplex heterogeneous network as illustrated in Figure 1.

At first, HGP builds node embeddings separately for user-item graph and user-group graph. To prevent the oversmoothing problem, it uses personalized PageRank scheme [3] when propagating node embedding through the whole graph structure. The HGP handles the heterogeneity of graph in two ways; it applies different predicting functions for each node type and combines two types of the node embeddings from each graph with an attention-based function. Finally, the HGP recommends items to a user by retrieving the nearest items to the user in user-item joint embedding space.

We evaluate our HGP on a large-scale real-world dataset collected from a social application for our experiments. The dataset includes 1.7M nodes consisting of 456K groups, 1.1M users, and 135K items.

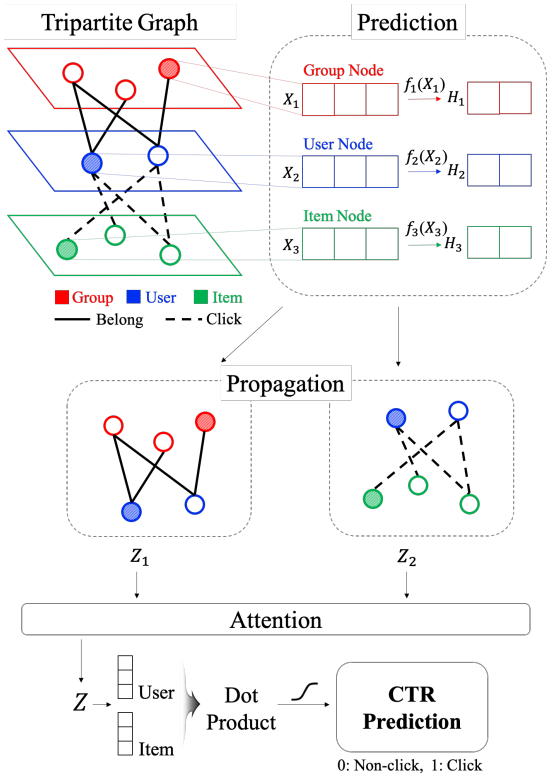

Figure 2: Schematic architecture of Heterogeneous Graph Propagation (HGP) as a social recommender system. HGP propagates neighborhoods independently for each edge type and then combines the final node representations with an attention model. We compute dot-product similarity between user and item representations to predict CTR.

The nodes have multiple attributes such as group topic (group node), demographic properties (user node), visual-linguistic information, and category (item node). The total number of edges is 4.7M. The experimental results show that our HGP outperforms competitive graph embedding methods. Moreover, as the number of layers increases, the HGP can achieve better performance. It implies that propagating item preference of friends indeed help improve the performance of recommendation.

HETEROGENEOUS GRAPH PROPAGATION

Items, users, and social relationships build a complex graph with multiple types of nodes and edges. To effectively handle different edge types, Heterogeneous Graph Propagation (HGP) propagates neighborhoods for each edge type independently and then combines the final node representations with an attention model. Also, HGP uses the predicting neural networks separated for each node type considering heterogeneity of node attributes.

In a heterogeneous graph $G = (V, E)$, there is a node type mapping function $\phi : V \rightarrow O$ and an edge type mapping function $\psi : E \rightarrow R$. We denote A_r as an adjacency matrix that only includes edges of type $r \in R$. To propagate self information of nodes to itself, the graph networks add self loops to the adjacency matrix: $\tilde{A}_r = A_r + I$. It is then symmetrically normalized as: $\hat{A}_r = \tilde{D}_r^{-1/2} \tilde{A}_r \tilde{D}_r^{-1/2}$, where $\tilde{D}_r$ is the diagonal degree matrix of $\tilde{A}$.

The nodes are initially represented by the feature matrix $X \in \mathbb{R}^{|V| \times n}$ where n is the number of features. We split the nodes by types: $X_1, X_2, ..., X_{|O|}$, apply each predicting neural network: $H_i = f_i(X_i)$ and concatenate the results: $H = [H_1, H_2, ..., H_{|O|}]$. Starting with $Z_r^{(0)} = H \in \mathbb{R}^{|V| \times m}$ for each edge type $r \in R$, HGP uses similar scheme as that of APPNP [3] which avoids the oversmoothing problem. The purpose of APPNP is not to learn deep node embedding, but to learn a transformation from attributes to class labels in the semi-supervised setting. HGP instead uses non-linear propagation with additional learnable weights to learn deep node representations:

$$Z_r^{(k+1)} = (1 - \alpha) ReLU(\hat{A}_r Z_r^{(k)} W_H^{(k)}) + \alpha H. \quad (1)$$

HGP combines the final node representation matrices with an attention model. Without loss of generality, we select i-th node (row) from $Z_r^{(K)}$ for each edge type. We stack these vectors building a matrix $Y_i \in \mathbb{R}^{|R| \times m}$. Using a single layer Transformer, the HGP performs self attention to Y_i:

$$Y_i' = Attention(Y_i W_Q, Y_i W_K, Y_i W_V), \quad (2)$$

where query, key and value are same, except that different weight matrices are used. Then, the HGP concatenates all rows of Y_i' and pass it to a linear layer, generating representation z_i for i-th node.

We compute dot-product similarity between the user and item representations to predict CTR. We optimize the model by reducing the cross-entropy loss. We sample subset of nodes for every batch

Table 1: Statistics of the datasets.

Node or edge types	Numbers
User	1,105,921
Group	456,483
Item	82,875
Item-User	3,746,650
Group-User	964,548

Table 2: Performance comparison of competing models. The hyphen '-' implies that we can not measure the stable performance due to the high variance of the results.

Models	ROC-AUC	PR-AUC	F1
FM	0.5725	0.5654	0.5400
metapath2vec	0.5	-	-
metapath2vec+EGES	0.6136	0.6290	0.5604
MNE+EGES	0.6158	0.6307	0.5660
FastGCN	0.6010	0.5937	0.5417
HGP (ours)	**0.6365**	**0.6378**	**0.5967**

to reduce memory size. Specifically, we adjust the sampling probability to be proportional to the number of nodes for each type. To reduce approximation variance, the sampling probability is also proportional to the degree of node.

EXPERIMENTS

Datasets

We use a dataset collected from a large-scale social network service including 1,645,279 nodes connected with 4,711,208 edges. *Group*, *User* and *Item* are three node types in the dataset. The group and user nodes are connected if the user belongs to the group. The item and user nodes are connected when the user positively interacts with the item. Table 1 shows the overall statistics of the dataset. To enhance accuracy and generality, the nodes contain various attributes such as group topic of group node, demographic properties of user node, and visual-linguistic information and category of item node. We extract high-level features with BERT and VGG16 for visual-linguistic attributes. We transform categorical attributes into dense features with linear embedding layers. Finally, we aggregate all features to represent the nodes. We use the first eleven days as a training set, the subsequent two days as a validation set, and the last four days as a test set.

Comparable Models

We compare our model with several models proposed for graph structures as well as a traditional model, i.e., Factorization Machine (FM).

metapath2vec [2]: It performs meta-path based random walk and leverages heterogeneous skip-gram model for node embedding. It only uses the identification information of nodes and does not cover attributes. In our datasets, we choose *G-U-I-U-G* as a meta-path which considers all node types.

metapath2vec+EGES: We modify metapath2vec to use the attributes. Following the attribute embedding methods from EGES [4], it densely embeds each attribute and fuses them using attention.

MNE+EGES: The nodes in MNE [5] use its distinctive embedding and several additional embeddings for each edge type, which are jointly learned by a unified graph embedding model. The attribute embedding method is same as EGES.

FastGCN [1]: The FastGCN is a homogeneous graph embedding method that directly subsamples the nodes for each layer altogether. It is scalable but does not consider edge types. In our task, it uses the same attribute embedding method as HGP for initial node features.

Results

We report the experimental results of the competitors on Table 2. We found that the values of PR-AUC and F1-score are proportional to that of ROC-AUC. The metapath2vec that does not use node

Table 3: Learning time of our model according to the use of sampling method.

Method	Learning Time Per Epoch
HGP w/o sampling	≈ 45 hrs
HGP	≈ 1.1 hrs

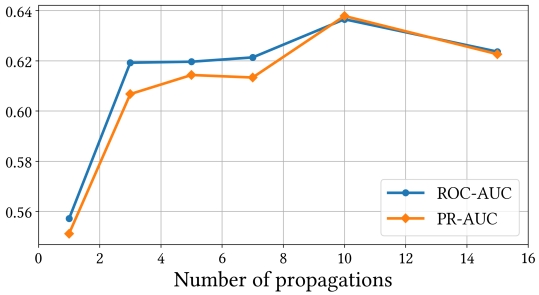

Figure 3: Performance comparison of HGP with different propagation steps.

attributes fails to learn CTR prediction. The HGP outperforms the FastGCN, which is not suitable for heterogeneous graphs and suffers from the oversmoothing problem. It also outperforms other recent heterogeneous graph models (metapath2vec+EGES and MNE+EGES). Moreover, the validation loss of our model converges within a half day by using sampling scheme. In Table 3, we compare the learning time of HGP according to the use of sampling scheme .

Figure 3 shows the performance comparison of HGP with different propagation steps. In our graph, HGP needs at least two propagation steps to know other members in a group (user → group → user). If the number of propagation steps is three, it can approach other preferred items of users who have common preferences (user → item → user → item). The performance of previous GCN architecture degrades as the number of propagation steps k increases, even when the k is two or three. Overall, the HGP achieves the best performance at k=10.

CONCLUSION

In this paper, we proposed a graph configuration, group-user-item tripartite attributed multiplex heterogeneous networks, for a social recommender system. To avoid the oversmoothing problem, the HGP propagates neighborhood information using the personalized PageRank scheme. The HGP can effectively handle the heterogeneity of a graph in two ways: 1) It builds node embeddings separately for each edge type and combines them with the attention function. 2) It uses different predicting functions on each node type. We tested our model on the large-scale real-world dataset and showed that the HGP outperforms competitive graph embedding methods in terms of various metrics.

REFERENCES

[1] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. Fastgcn: fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations (ICLR)*.

[2] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 135–144.

[3] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *International Conference on Learning Representations (ICLR)*.

[4] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 839–848.

[5] Hongming Zhang, Liwei Qiu, Lingling Yi, and Yangqiu Song. 2018. Scalable Multiplex Network Embedding.. In *IJCAI*. 3082–3088.