Advanced Neural Graph Collaborative Filtering based on time sampling

Kibum Kim
Hanyang University
rlqja1107@gmail.com

Sukwon Yun
Hanyang University
tyryey89@gmail.com

Jungmin Lee
Hanyang University
jungmin9195@gmail.com

Bumjin Cho
Hanyang University
whqjawls96@gmail.com

Huiseung Jo
Hanyang University
whtjddn75@gmail.com

ABSTRACT

In this paper, we propose an advanced version of Neural Graph Collaborative Filtering in the perspective of time sampling. At first, by using side information such as social relationships and item ID, we suggest an advanced version of Heterogeneous Graph Neural Recommender (HGNR). Moreover, we adopted Agglomerative clustering methodology to effectively capture user's group information.

In the case where side information as social data is unobtainable, we instead made use of time information effectively. Existing models did not sufficiently use time information on their model which might cause a big problem when a sequence of its data becomes crucial. Unlike static graphs in a recommender system, we propose Weight Sequence Neural Graph (WSNG) model which is a dynamic graph that utilizes GRU to efficiently capture historical interactions between a user and the item. In Epinion dataset, WSNG outperforms existing methods by a significant margin.

Keywords

Recommender Systems, Collaborative Filtering, Graph Neural Network, Clustering, Dynamic Graph Representation

1. INTRODUCTION

Recommender system has become a basic function in online services such as e-commerce, social media platforms, and advertising. The main purpose of the recommendation system is to predict what users might be interested in among the various items. Most recommend system models utilize latent vectors of users and items. Collaborative filtering is a methodology that analyzes patterns in which users evaluate items and patterns in which items are evaluated to users and then makes user latent vector and item latent vector.

With the success of recommender system, there has been emerging work that makes use of recommender systems in the graph domain[4, 11]. As a deep-learning-based graph, Graph Neural Network has been successfully applied to many fields, recommender system as well takes advantage of its properties of representing user, item node and its topological structure[12]. Especially, it becomes powerful when incorporating external information such as social relationship data. In the perspective of integrating node information with its neighbor node, the observed interaction between user and item is easily expressed in the form of a bipartite graph[1]. Also, such interaction could be interpreted as one-hop neighbors and it has been proven that using multi-hop relations by graph is beneficial to recommender systems.

One of the cases which apply Graph Neural Network in Recommender System, Graph Convolution Network(GCN)[4] would be a representative methodology that makes use of convolution kernel into a graph to aggregate the neighbor node's feature by edges. However, GCN does not consider encoding the message passing procedure into the graph. For tackling the message passing problem, Neural Graph Collaborative Filtering (NGCF)[1] methodology was developed to encode such messages. NGCF makes use of the different shape of adjacency matrix with GCN.

$$GCNAdjacency = A,\ NGCFAdjacency = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \quad (1)$$

The left-top and right-bottom section of NGCF adjacency matrix was not utilized to message pass or aggregate the information among neighbor nodes. That is, only user-item interaction was used to aggregate it, not user by user or item by item. HGNR[2] was developed to tackle the user-user and item-item message passing. HGNR methodology enables the aggregation of information among similar users or items. It assumed that one user and his friends are similar. The information of friends is inserted into the left-top section and the information of items which is captured by S-BERT on review data is inserted into the right-bottom section.

Methodologies spoken above are static graphs that do not consider the time information. It may be important to figure out when user-item interaction has a trend over time. To tackle this problem, We will propose a new methodology which is called Weight Sequence Neural Graph (WSNG). The remainder content of this paper is as follows. In Section 2, we propose our advanced NGCF model from

three perspectives. In Section 3, we describe how we conducted our experiment with Epinion dataset. Following by Section 4, we conclude our proposed model with its significant contribution and remain our future work. The source code of advanced NGCF can be found in https://github.com/rlqja1107/Graduation_Paper

2. MODEL

In this section, we present advanced Neural Graph Collaborative Filtering model and its structure in three key frameworks: (1) an HGNR and procedure of generating embeddings in terms of their item name, illustrated in Figure 1; (2) Agglomerative Clustering which groups each user as a cluster and combine each cluster conditioned on their variance based on their social relations, illustrated in Figure 2; (3) Weight Sequence Learning which is a dynamic graph that reflects and utilizes time information on GCN using GRU, illustrated in Figure 3.

2.1 HGNR on name

Following the conventional usage of graph structure in prospective of recommender system[1, 2], we first need to define matrix which would be input to the Graph Neural Network such as GCN[4]. Three key components in our proposed model, HGNR name, would be social relations matrix(S), rating matrix(R), semantic matrix(C) with their user set $U(|U| = M)$ and item set $I(|I| = N)$.

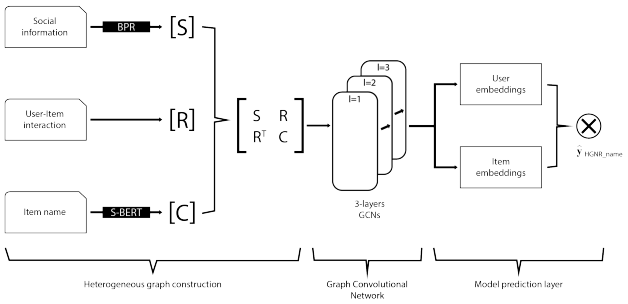

Fig. 1: An illustration of HGNR architecture. The social information [S], user-item interaction [R], item name [C] generates heterogeneous graph which would be input to the GCN layers. By the output with embeddings, it predicts the likelihood of user and item interaction.

2.1.1 S (Social Relations Matrix).

Based on the dataset that provides social relationships such as a trust or follows between users, we link two users when one user trusts the other user. We denote their relationships as 1(when there exists linkage) and 0(with no link) as equation (2). In many cases, the problem occurs with their sparseness in the social dataset. To deal with this sparsity, we made use of BPR [7] which is an effective optimization method that is ranking-based, and use the pairwise loss to predict further linkage. With additional 20 links for each user, the elements of S matrix $S \in \mathbb{R}^{M \times M}$ would be:

$$s_{ij} = \begin{cases} 1 & \text{if user } i \text{ trust user } j \\ 0 & \text{Otherwise} \end{cases} \tag{2}$$

2.1.2 R (Rating Matrix).

Following the mainstream of the recommender system[6], we denote 1(when there is an interaction between a user and certain items) as equation (3). We consider implicit feedback here thanks to their availability against explicit feedback as numerical ratings. We also use transpose of R matrix when combining all matrix together to ensure the entire matrix size. The elements of R matrix $R \in \mathbb{R}^{M \times N}$ would be:

$$r_{ij} = \begin{cases} 1 & \text{if user } i \text{ interacted with item } j \\ 0 & \text{Otherwise} \end{cases} \tag{3}$$

2.1.3 C (Semantic Matrix).

To link item with each other, we used cosine similarity between two items based on their names as equation (4). In contrast to the earlier model HGNR[2], we made use of the property of each item's ID which is intuitive to match similar items. As in S matrix, we made 20 virtual links considering their cosine similarities by use of S-BERT[8] which effectively extracts semantic information from each items' name. The elements of C matrix $C \in \mathbb{R}^{N \times N}$ would be:

$$c_{ij} = \begin{cases} 1 & \text{if item } i\text{'s ID is close to item } j\text{'s ID} \\ 0 & \text{Otherwise} \end{cases} \tag{4}$$

With S, R, C, we finally combine three matrices in one matrix $H \in \mathbb{R}^{(M+N) \times (M+N)}$ as,

$$H = \begin{bmatrix} S & R \\ R^T & C \end{bmatrix}$$

Now, we regard this H matrix in terms of graph Laplacian[13] since Laplacian in graph structure effectively utilizes neighbor node's information which has its impact on generating node embedding. Regarding H matrix as degree matrix, and its diagonal matrix as D, Laplacian matrix would be:

$$L^c = D^{-\frac{1}{2}} H D^{\frac{1}{2}} \tag{5}$$

We now present its updating function with graph Laplacian matrix as input of GCN layer on HGNR on name model:

$$\begin{aligned} &HGNR_{name}(H)^{(l)} = E^{(l)} \\ &= \sigma((L^c + I)E^{(l-1)}W_0^{(l)} + L^c E^{(l-1)} \odot E^{(l-1)}W_n^{(l)}) \end{aligned} \tag{6}$$

Where $\boldsymbol{E}$ is the concatenation of all users and items embedding in l GCN layers, $\boldsymbol{I}$ is an identity matrix, and $\boldsymbol{W_0}$ is a weight that propagates the original node's information as in form of embedding while $\boldsymbol{W_n}$ is a weight that aggregates original node's neighbor nodes. Both weights are trainable and the $\sigma(\cdot)$ is an activation function such as LeakyReLU[9].

After the sequence of layers, we obtain embeddings from each GCN layer and concatenate them. With concatenated embeddings $\boldsymbol{e_u}$ & $\boldsymbol{e_i}$, we finally predict the likelihood of a user u interacts with item i as a product form of:

$$\widehat{y_{ui}} = \boldsymbol{e_u^T e_i} \tag{7}$$

2.2 Agglomerative Clustering

The existing HGNR model applied BPR optimization in the process of creating S matrix. We would like to increase the performance with different models defining social relationships among users. We defined each user's binary vector from the user by user matrix we created based on existing social relationships among users to the user's input vector. Based on these input vectors, we proceeded our experiments using cluster analysis. Because cluster analysis is a analysis that makes similar elements in the same cluster and other clusters when elements are different from each other, we expect to utilize social relationships among users. We used agglomerative cluster analysis among many cluster analysis methods because it can overcome some weaknesses of the K-means method[10] and the problem of time complexity.

Agglomerative cluster analysis is a model that designates each input as a cluster and combines two similar clusters until the number of clusters set by the analyst remains. We proceeded the experiments with the ward method which is combining the two clusters that increase the variance in all clusters smallest.

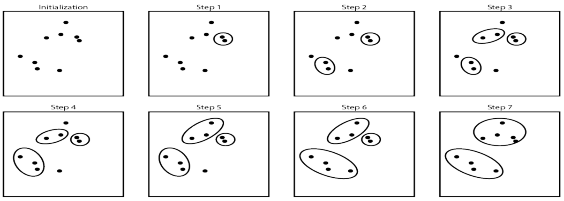

Fig. 2: An illustration of Agglomerative Clustering

At first, each input vector forms a cluster. Each cluster is combined with the nearest cluster, following the designated method. When the number of clusters set by the user is reached, the model stops. We have created a new S matrix as follows.

$$group(u_i) = \begin{cases} 0 & \text{if } u_i \text{ in group 0} \\ 1 & \text{if } u_i \text{ in group 1} \\ 2 & \text{if } u_i \text{ in group 2} \\ 3 & \text{if } u_i \text{ in group 3} \end{cases} \tag{8}$$

$$s_{ij} = \begin{cases} 1 & \text{if } group(u_i) = group(u_j) \\ 0 & \text{if } group(u_i) \neq group(u_j) \end{cases} \tag{9}$$

2.3 Weight Sequence Learning

For Neural Graph Collaborative Filtering[1], the model only encodes the message passing into the graph, but not the time information since it is a static graph. In most cases, the interactions of items or events are followed by the time. They may be strongly or weakly correlated over time. For example, a user could change his or her tendency or preference of certain items over time. A user who preferred a comedy or action genre in the past may enjoy watching the romance now. It may be more accurate or practical to put time information on graph representation, which is called dynamic graph representation. Thus, We try to reflect time information on NGCF[1] model by using EvolveGCN[3] concept. We call the new derived model **WSNG**.

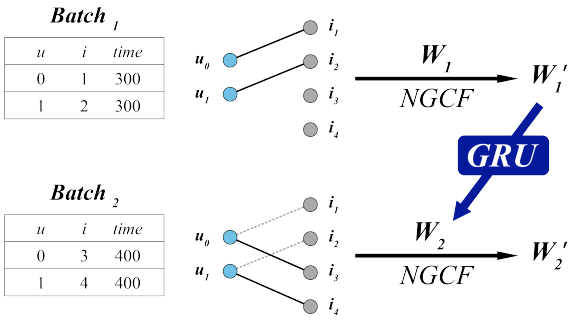

Fig. 3: The example of WSNG. W_1 is used for training the matrix while $W_1{}'$ is trained matrix after several iteration of NGCF. The solid line is edge included in batch. The dotted line is a past interaction which is occurred before corresponding batch

EvolveGCN adapts recurrent neural network to weight matrix for representing the dynamic graph on GCN[4]. Previously, most of the model directly encoded the time information on the node or edge embedding. DGCF[5] directly encodes the vector called time context vector to node embedding for considering the time. EvolveGCN put the time information on GCN weight matrix rather than node embedding. Each timestamp constructs the learned weight matrix W_t^l, which l and t indicates the layer and timestamp respectively. We can briefly write it using GRU

$$W_{t+1}^l = GRU(W_t^l, W_t^l) \tag{10}$$

GRU (Previous hidden : h^{t-1}, Input : x^t):
$r^t = \sigma(\mathrm{W}_r h^{t-1} + U_r x^t)$
$u = \sigma(\mathrm{W}_u h^{t-1} + U_u x^t)$
$\tilde{h^t} = \tau(\mathrm{W}h^{t-1} * r^t + U x^t)$
$h^t = (1-u) * h^{t-1} + u^t * \tilde{h^t}$
return h^t

Using recurrent type network, W_{t+1}^l matrix is constructed to keep the time information. It is possible to replace the GRU with LSTM. When RNN does not have input information at time t which situation is appropriate for now, W_t^l is directly utilized on both hidden state and input.The only difference with standard GRU is hidden state's shape that standard LSTM is a vector unlike this GRU is a matrix shape. We would use this framework in which W_{t+1}^l would contain the previous graph information and trend of time flow.

First, We should define the notation for describing the algorithm. We can arrange the notation using the following table.

Notation	Explanation
I	All interaction data
I_{train}	Interaction train data
I_{test}	Interaction test data
B_i	i-th batch
W_i^l	NGCF Weight matrix before learning at i-th batch
$W_i^{l'}$	NGCF Weight matrix after learing at i-th batch
A_i	Adjacency Matrix at i-th batch

Table 1. : Notation

Using the above notation, I representing the graph is randomly organized, ignores the occurrence order. For detecting the time trend, we need to sort I. With NGCF[1] considering I occurs at the same time, we will combine this approach with another new approach. We assume that the interactions in the same batch B_i, occur simultaneously but two interactions included in B_i and B_j, $i \neq j$ occur at different time. If i is smaller than j, the interaction in B_i occurs earlier than B_j. Time information can be obtained from the time difference between batches. It is efficient that reflecting time trend and learning NGCF weight matrix is much faster than learning one by one. This approach has a trade-off that when batch size is small, we can precisely detect the time trend, but training time would be slow. Inverse situations when batch size is big can be considered as well.

Using the interaction in B_i, we can make an adjacency matrix same as NGCF's thing.

$$A_i = \begin{bmatrix} 0 & B_i \\ B_i^T & 0 \end{bmatrix} \tag{11}$$

W_i^l can be trained as representing the graph and message passing to the neighbor node. NGCF training iteration on each batch is restricted by early stop using loss value. After sufficiently training the weight matrix is equal to the standard NGCF training step, $W_i^{l'}$ is thought of as containing the node feature and structure information. The weight Sequence learning approach is introduced to put the time information on the weight matrix. In this step, we use the EvolveGCN[3] idea.

However, Weight matrix information could be transformed into a completely different matrix after encoding to GRU, which loses the previous structure and node feature information. So, We should define the regularization loss to prevent information loss.

$$L_{reg} = \sum \left| {W_{i+1}^l}' - W_i^{l'} \right| \tag{12}$$

Regularization takes two roles that prevent information loss and predict the future structure and node feature. Using recurrent networks with regularization loss save structure, node feature, and time information into the weight matrix simultaneously. To keep each layer's information, we use recurrent networks on each NGCF layer separately. Below **Algorithm 1** summarizes the whole step.

Algorithm 1 WSNG Algorithm

Input: I

Sort I by timestamp
I_{train}, I_{test} = Split the I by 0.8 for training and 0.2 for testing
Split I_train into equal size of batch and construct A_i, i=1,2....
Initialize the NGCF Weight Matrix, $W_0^l, l = 1, 2, 3$

for $B_i, i = 1, 2, ...n,$ **do**
    W_i^l = NGCF.parameters(), l=1,2,3
    Learning the Matrix using A_i by NGCF
    $W_i^{l'}$ = NGCF.parameters(), l=1,2,3
    W_{i+1}^l = GRU($W_i^{l'}$, $W_i^{l'}$)
    NGCF.parameters() = W_{i+1}^l, l =1,2,3
end for
Test the I_{test}

Detailed code is uploaded on github, https://github.com/rlqja1107

3. EXPERIMENT

To evaluate the performance of the model, Epinion dataset has been chosen as our experimental data. Epinion is a review dataset for online shopping malls. To evaluate how the performance of the model depends on where the time axis is taken, we conducted experiments on both kinds of time axes so that the ratio of train data with test data can be 8:2, 9:1. Train data only exists as value before the time axis and test data only exists as value after the time axis. Through preprocessing, we limited the existence of evaluation data only to customers who are in learning data. For the item, this rule was not imposed.

Information on the dataset is provided in table 2 and parameter setting is provided in table 3 as well.

Table 2. : Statistics of the dataset

train:test	epinion82	epinion91
users	16,626 : 5,228	23,082 : 3,965
items	16,868 : 8,977	26,168 : 6,759
average interaction	2.999 : 2.488	3.240 : 2.291

Table 3. : Parameter Settings

parameters	settings
epoch	2,000
learning_rate	16,626 : 5,2280.01, 0.001, 0.0001
batch_size	[64, 1,024]
reg	0.0001
dropout	0.3
embedding_dim	64
GCN_layer	3
HGNR: (alpha, beta)	[(0,0), (1,1)]

4. CONCLUSION AND FUTURE WORK

We have proposed a variety of methodologies that better utilize NGCF in the problem of recommendation using implicit feedback data. It suggested how to make S and C matrix differently from conventional HGNR using agglomerative clustering and how to apply Weight Sequence Learning to NGCF. The performance measurement of the model has been calculated by NDCG and HR with Top 10 recommendations. We can mention that some methodologies perform better than other conventional models. For instance, HGNR_name is **12%** higher than conventional HGNR and WSNG is **23%** higher than HGNR in Epinion82. Also, WSNG is **12%** higher than NGCF in Epinion91.

Table 4. : Metric Report

Dataset	Epinion82		Epinion91	
	NDCG@10	HR@10	NDCG@10	HR@10
BPR	0.000094	0.000956	0.000219	0.001513
NGCF	0.000389	0.004399	0.000639	0.006305
HGNR	0.000505	0.004973	0.000579	0.005801
HGNR_name	0.000537	0.005574	0.000349	0.003531
HGNR_group3	0.000426	0.004782	0.000472	0.004540
HGNR_name_group3	0.000407	0.004017	0.000470	0.005044
HGNR_group4	0.000337	0.005356	0.000641	0.005296
HGNR_name_group4	0.000329	0.005164	0.000426	0.004035
WSNG	0.000666	0.006121	0.000888	0.007062

Future works could be progressed with the following three viewpoints. At first, our models will need to be applied to more datasets such as Yelp. Second, when we perform experiment with agglomerative clustering analysis, further experiments have to be made with various number of clusters. Third, with Weight Sequence Learning model, the robustness and stability of model has to be obtained with fine tuning of various hyperparameters.

Acknowledgement: This work was supported by the ICT RD program of MSIP. [R7518-16-1001, Innovation hub for High Performance Computing]

5. REFERENCES

[1] Wang, X., He, X., Wang, M., Feng, F., Chua, T. S. (2019, July). [*Neural graph collaborative filtering*]. In Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval (pp. 165-174).

[2] Liu, S., Ounis, I., Macdonald, C., Meng, Z. (2020, July). [*A Heterogeneous Graph Neural Model for Cold-Start Recommendation*]. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 2029-2032).

[3] Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., ... Leiserson, C. (2020, April). [*Evolvegcn: Evolving graph convolutional networks for dynamic graphs*]. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 04, pp. 5363-5370).

[4] Kipf, T. N., Welling, M. (2016). [*Semi-supervised classification with graph convolutional networks*]. arXiv preprint arXiv:1609.02907.

[5] Li, X., Zhang, M., Wu, S., Liu, Z., Wang, L., Yu, P. S. (2020). [*Dynamic Graph Collaborative Filtering*]. arXiv preprint arXiv:2101.02844.

[6] Koren, Y., Bell, R., Volinsky, C. (2009). [*Matrix factorization techniques for recommender systems*]. Computer, 42(8), 30-37.

[7] Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L. (2012). [*BPR: Bayesian personalized ranking from implicit feedback*]. arXiv preprint arXiv:1205.2618.

[8] Reimers, N., Gurevych, I. (2019). [*Sentence-bert: Sentence embeddings using siamese bert-networks*]. arXiv preprint arXiv:1908.10084.

[9] Xu, B., Wang, N., Chen, T., Li, M. (2015). [*Empirical evaluation of rectified activations in convolutional network*]. arXiv preprint arXiv:1505.00853.

[10] Likas, A., Vlassis, N., Verbeek, J. J. (2003). [*The global k-means clustering algorithm*]. Pattern recognition, 36(2), 451-461.

[11] Berg, R. V. D., Kipf, T. N., Welling, M. (2017). [*Graph convolutional matrix completion*]. arXiv preprint arXiv:1706.02263.

[12] Wu, S., Sun, F., Zhang, W., Cui, B. (2020). [*Graph neural networks in recommender systems: a survey*]. arXiv preprint arXiv:2011.02260.

[13] Mohar, B., Alavi, Y., Chartrand, G., Oellermann, O. R. (1991). [*The Laplacian spectrum of graphs*]. Graph theory, combinatorics, and applications, 2(871-898), 12.