GNN with Edge Enhancement for Recommendation

Kaijiang Deng[* 1] **Kaiwen Zhu**[* 1]

Abstract

Nowadays recommendation is an important task intended to filter out relevant information from massive data. Due to the nature graph structure of the problem, Graph Neural Network (GNN) is a powerful tool for recommendation. In this work we propose a method termed edge enhancement to improve information propagation in GNN. This could also be a promising direction for mitigating the cold start problem and self-evolution.

1. Background and Related Work

Recently, deep learning methods have been widely adopted in recommendation systems to effectively capture non-linear and non-trivial user-item relations (Wu et al., 2022). Among them, Graph Neural Network draws extraordinary attention because of its excellent performance on homogeneous and heterogeneous graphs.

The main idea of GNN is to compute node features by aggregating information from neighbors based on a suitable graph. GNN often includes multiple propagation layers, which consist of aggregation and update operations. Most of the related works, such as GCN (Graph Convolutional Network) (Kipf & Welling, 2017) and GAT (Graph Attention Network) (Veličković et al., 2018), focus on different ways of these two operations in order to effectively and efficiently propagate information among entities (Wang et al., 2021).

According to the type of information used, existing works for non-sequential recommendations using GNN could be classified into three categories: user-item collaborative filtering, social recommendation, and knowledge graph based recommendation.

User-Item Collaborative Filtering: Intuitively, representations of a user would be enhanced by its closely interacted items, and users' representations would influence those interacted items in turn. This is what user-item collaborative filtering utilizes. Such a process may help find close relationships between users and items, which supports accurate edge prediction. As for the construction of the graph, one way is to directly apply GNN on the original user-item bipartite graph (van den Berg et al., 2017), while another way is to enrich the original graph by introducing new edges (Sun et al., 2019) or nodes (Wang et al., 2020a).

Social Recommendation: The basic idea of social recommendation is that users with strong relationships may share similar preferences on items. Considering the high performance of GNN on social relationship problems, many works try to incorporate social relationships into representations. For example, DGRec dynamically infers influences of neighbors during learning to differentiate friends (Song et al., 2019), and ESRP uses the autoencoder to modify observed social relationships by filtering irrelevant relationships (Yu et al., 2022).

Knowledge Graph Based Recommendation: Knowledge of items is helpful to enhance item representation. Besides, the results could be more interpretable provided with such knowledge. For constructing the graph, one popular direction is to incorporate user nodes into the knowledge graph (Wang et al., 2019). For learning, many works apply attention mechanism to distinguish different relations in the knowledge graph (Wang et al., 2020b).

2. Model

In this work we focus on recommendation problems in the academic scenario. That is, given the cooperation relationships among authors, the reference relationships between authors and papers, and the citation relationships among papers, the task is to recommend papers to authors.

We build a heterogeneous graph to model the problem. There are two types of nodes in the graph, representing authors and papers respectively. The edges among authors represent cooperation relationships, the edges among papers represent citation relationships, and the edges from authors to papers represent reference relationships. Besides, inversed edges and self-loops are added to enrich the graph.

The basic idea of our algorithm is learning the feature of nodes by doing convolutions on this heterogeneous graph. Furthermore, we propose *edge enhancement* in the hope of broadening the channel of information propagation and fully exploiting the close relations among nodes.

[*]Equal contribution [1]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China.

2.1. Relational Graph Convolution

Based on GCN for homogeneous graphs, Relational Graph Convolution Network (RGCN) (Schlichtkrull et al., 2017) is a message-passing framework for learning valuable latent features of heterogeneous graphs, excelling in link prediction tasks (Thanapalasingam et al., 2022). RGCN would alter the graph to handle message passing for different relations separately to extend the basic GCN framework to heterogeneous graphs. This is shown in Equation (1).

$$H = \sigma\left(\sum_{r=1}^{R} A_r X W_r\right), \tag{1}$$

where R is the number of relations, A_r is an adjacency matrix describing the edge connection for a given relation r and W_r is a relation-specific weight matrix. In heterogeneous graphs, nodes and edges of various types would naturally form multiple kinds of relations, so that information passed by different edges would be treated differently and then aggregated.

As for the convolution of one relation, fed with features $h_j^{(l)}$ of node j, a convolution layer updates the feature of node i as

$$h_i^{(l+1)} = \sigma\left(b^{(l)} + \sum_{j\in\mathcal{N}(i)} \frac{1}{|\mathcal{N}(i)|} h_j^{(l)} W^{(l)}\right), \tag{2}$$

where $\mathcal{N}(i)$ is the set of neighbors of node i (including i itself) and σ is an activation function.

2.2. MetaPath2Vec

Instead of random initialization, authors' features are initialized from MetaPath2Vec model to represent authors' characteristics better. MetaPath2Vec is a framework dedicated to representation learning in heterogeneous graphs (Dong et al., 2017). Since identically treating different types of nodes and relations would lead to the production of indistinguishable representations for heterogeneous nodes, MetaPath2Vec introduced Heterogeneous Skip-Gram and Meta-Path-Based Random Walks to learn desirable node representations in heterogeneous graphs.

Heterogeneous Skip-Gram: By altering the original Skip-Gram model, we can enable the model to learn from different types of relationships. In a heterogeneous graph $G = (V, E, T)$, where $T = T_V \cup T_E$ denotes the types of nodes and edges, if $|T_V| > 1$, the model would maximize the probability of having the heterogeneous context $N_t(v)$, $t \in T_V$ given a node v:

$$\arg\max_\theta \sum_{v\in V}\sum_{t\in T_V}\sum_{c_t\in N_t(v)} \log p(c_t|v;\theta) \tag{3}$$

where $N_t(v)$ denotes v's neighborhood with the t^{th} type of nodes and $p(c_t|v;\theta)$, commonly defined as a softmax function, is the conditional probability of having a context node c_t given a node v. Negative Sampling, which means sampling a relatively small set of nodes from the corpus for the construction of softmax, is introduced for optimization, and the Equation (3) would be updated as:

$$\log\sigma(X_{c_t}\cdot X_v) + \sum_{m=1}^{M}\left(\mathbb{E}_{u^m\sim P(u)}[\log\sigma(-X_{u^m}\cdot X_v)]\right)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$, X_i represents the embedding vector for node i and $P(u)$ is the pre-defined distribution from which a negative node u^m is drew for M times.

Meta-Path-Based Random Walks: Due to the fact that conventional random walkers would cause biased treatment to different types of nodes, Meta-Path-Based random walks are designed to generate paths capturing both the semantic and structural correlations between them. For a heterogeneous graph $G = (V, E, T)$ and a meta-path scheme $\mathcal{P}$: $V_1 \xrightarrow{R_1} V_2 \xrightarrow{R_2} \cdots V_t \xrightarrow{R_t} V_{t+1}\cdots \xrightarrow{R_{l-1}} V_l$, wherein $R = R_1 \circ R_2 \circ \cdots \circ R_{l-1}$ defines the composite relations between node types V_1 and V_l, the transition probability at step i is defined as follows:

$$p(v^{i+1}|v_t^i,\mathcal{P}) = \begin{cases} \frac{1}{|N_{t+1}(v_t^i)|} & (v^{i+1}, v_t^i)\in E, \phi(v^{i+1}) = t+1 \\ 0 & (v^{i+1}, v_t^i)\in E, \phi(v^{i+1}) \neq t+1 \\ 0 & (v^{i+1}, v_t^i)\notin E \end{cases} \tag{4}$$

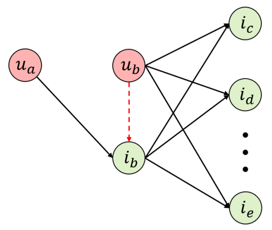
where $v_t^i \in V_t$, $N_{t+1}(v_t^i)$ denote the V_{t+1} type of neighborhood of node v_t^i and $\phi(v): V \to T_V$ is the mapping from nodes to types. In practice, V_1 is often the same as V_l to facilitate recursive guidance for random walkers. The Meta-Path-Based Random Walks strategy ensures that various relations between nodes of different types are appropriately incorporated into the skip-gram model.

2.3. Edge Enhancement

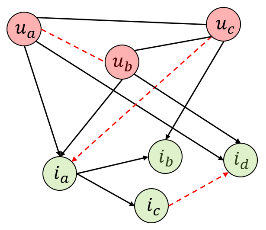
One of the principles of graph neural networks is that information propagates through edges, while edges connect close-knit nodes. Based on this, we could safely assume that if there are more edges between close-knit nodes, then information will propagate more efficiently, so that the model could learn better node embeddings.

Motivated by this assumption, we propose an idea termed edge enhancement: find nodes with close relationships and then add edges between them to build a new graph. In this graph, relationships among nodes could be utilized more adequately, thus enabling the model to capture more abundant node features.

To accomplish edge enhancement, we propose two attempts, which are depicted in Figure 1.



(a) Simple solution: add edges representing writing relationships



(b) Further solution: add edges predicted by the model and train the model iteratively

Figure 1. Visualization of two attempts for edge enhancement.

2.3.1. Simple Solution

In this problem, existing edges do not embody the relation that authors write papers, which should be important for associating authors with papers. Hence we may as well add edges representing writing relationships. This could be inferred from the graph structure. For some paper and some author, if most of the papers cited by this paper are also cited by this author, then this paper is likely to be written by this author.

Denote the set of papers cited by author a by C_a, the set of papers cited by paper p by C_p. If

$$w_{ap} = \frac{|C_a \cap C_p|}{|C_p|} \tag{5}$$

is greater than some threshold, then we add an edge from a to p with weight w_{ap} to the graph.

In convolution, when aggregating information from a newly added edge (a, p), we discount the information by w_{ap} to penalize unconfident edges and reward confident edges. Mathematically, we substitute Equation (2) with

$$h_i^{(l+1)} = \sigma\left(b^{(l)} + \sum_{j \in \mathcal{N}(i)} \frac{w_{ji}}{|\mathcal{N}(i)|} h_j^{(l)} W^{(l)}\right) \tag{6}$$

for edges representing writing relationships.

2.3.2. Further Solution

The previous solution utilizes prior knowledge and graph structure to enhance edges, which is limited to a narrow scenario. To get rid of such restrictions, we try to *learn* it. Since this model is trained to predict edges, we add the predicted edges with high confidence into the graph and then train again on the new graph.

In implementation, to find the virtual edges with high confidence, we use locality sensitive hashing to find node pairs with the highest cosine similarities. Then we connect those node pairs and set the edge weight as the cosine similarities of the two endpoints. Again, in convolution node embeddings are updated by Equation (6).

2.4. Network Structure

Figure 2 summarizes the structure of our network.

WeightedGraphConv will implement convolution on node features. If Edge Enhancement is applied, edge weights would be taken in and normalized, serving as w_{ji} in Equation (6).

Basic Block consists of a WeightedGraphConv layer, a Batch Normalization layer and a LeakyReLU layer.

Our network includes three basic blocks and one WeightedGraphConv layer, and we use margin loss as the loss function for node v:

$$\mathcal{L} = \sum_{v_i \sim P_n(v), i=1,2,\cdots,k} \max(0, M - y_{u,v} + y_{u,v_i})$$

where u and v are nodes with an edge connected, v_i are negative samples following the distribution $P_n(v)$, M is a constant hyperparameter and $y_{i,j}$ is the score given by the model indicating the possibility of an edge between i and j.

The inputs of the model are the features of authors and papers, and the outputs are the embeddings of nodes. For each author-paper pair a-p, dot product of their embeddings would serve as the score to decide whether to recommend p to a or not.

3. Results and Discussion

In our experiments, we add inferred edges with weight in Equation (5) no less than 0.5 to the graph in simple solution, and add 4 times more edges for each relation in further solution. Using F1-score to measure the performance of edge prediction, Table 1 shows the results of different methods. The loss curve during training is plotted in Figure 3, from which we could see that the model does capture the features and thus works well in predicting edges (at least on the training set), and the further edge enhancement behaves well at this especially.

To our regret, although edge enhancement increases the F1-score slightly, the improvement is not so significant. Several possible causes may be to blame:

- The original graph is already dense enough, so more

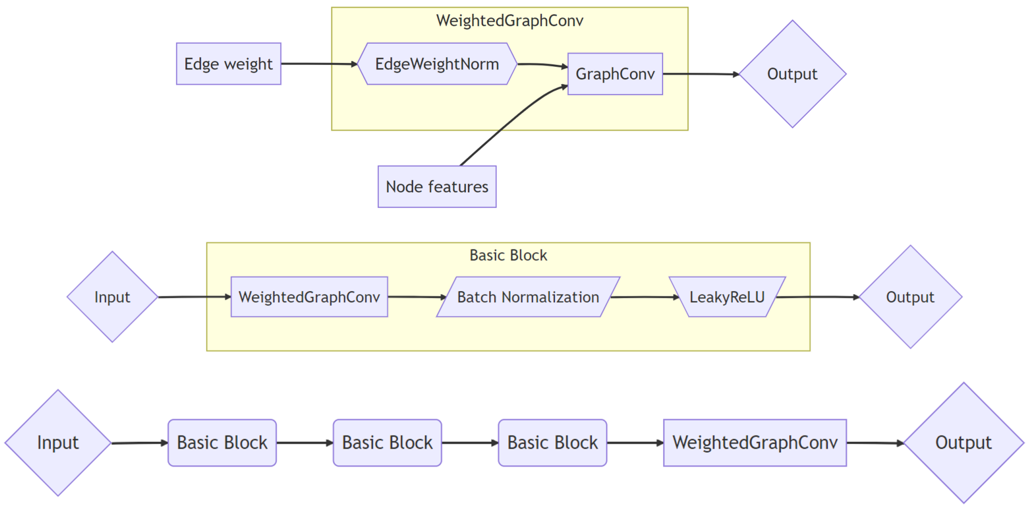


Figure 2. Structure of the Network

Table 1. F1-score of edge prediction

Method	F1-score
without edge enhancement	0.9396
simple edge enhancement	0.9402
further edge enhancement	0.9416

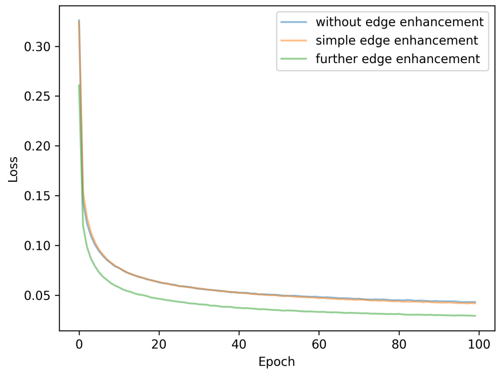


Figure 3. Training loss

edges could hardly help to improve information propagation.

- The predicted edges added to the graph may be incorrect, whereby error will accumulate and propagate.
- Too many edges may cause homogenization of node embeddings, preventing the model from learning distinct features.

4. Future Work

As discussed in the previous section, dense graphs may limit the potential of edge enhancement. On the other hand, if the graph is sparse (*i.e.*, there are few edges in the graph), edge enhancement is likely to show better performance. This is just the scenario of *cold start* problem in recommendation, which requires recommending items to users with few existing interactions between users and items. Therefore, it is promising to apply edge enhancement to cold start problem.

Besides, the schema of our further edge enhancement is like *self-evolution*. That is, the model improves itself iteratively by learning, and each iteration will enhance the next iteration. If the error accumulation and propagation is prevented, then we can imagine that the model equipped with edge enhancement could obtain such an evolution capacity.

References

Dong, Y., Chawla, N. V., and Swami, A. Metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Dis-*

covery and Data Mining, KDD '17, pp. 135–144, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348874. doi: 10.1145/3097983.3098036. URL https://doi.org/10.1145/3097983.3098036.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks, 2017.

Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., and Welling, M. Modeling relational data with graph convolutional networks, 2017.

Song, W., Xiao, Z., Wang, Y., Charlin, L., Zhang, M., and Tang, J. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, pp. 555–563, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359405. doi: 10.1145/3289600.3290989. URL https://doi.org/10.1145/3289600.3290989.

Sun, J., Zhang, Y., Ma, C., Coates, M., Guo, H., Tang, R., and He, X. Multi-graph convolution collaborative filtering. In *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 1306–1311, 2019. doi: 10.1109/ICDM.2019.00165.

Thanapalasingam, T., van Berkel, L., Bloem, P., and Groth, P. Relational graph convolutional networks: a closer look. *PeerJ Computer Science*, 8:e1073, nov 2022. doi: 10.7717/peerj-cs.1073. URL https://doi.org/10.7717%2Fpeerj-cs.1073.

van den Berg, R., Kipf, T. N., and Welling, M. Graph convolutional matrix completion, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks, 2018.

Wang, S., Hu, L., Wang, Y., He, X., Sheng, Q. Z., Orgun, M. A., Cao, L., Ricci, F., and Yu, P. S. Graph learning based recommender systems: A review, 2021.

Wang, X., He, X., Cao, Y., Liu, M., and Chua, T.-S. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pp. 950–958, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330989. URL https://doi.org/10.1145/3292500.3330989.

Wang, X., Jin, H., Zhang, A., He, X., Xu, T., and Chua, T.-S. Disentangled graph collaborative filtering. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pp. 1001–1010, New York, NY, USA, 2020a. Association for Computing Machinery. ISBN 9781450380164. doi: 10.1145/3397271.3401137. URL https://doi.org/10.1145/3397271.3401137.

Wang, Z., Lin, G., Tan, H., Chen, Q., and Liu, X. Ckan: Collaborative knowledge-aware attentive network for recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pp. 219–228, New York, NY, USA, 2020b. Association for Computing Machinery. ISBN 9781450380164. doi: 10.1145/3397271.3401141. URL https://doi.org/10.1145/3397271.3401141.

Wu, S., Sun, F., Zhang, W., Xie, X., and Cui, B. Graph neural networks in recommender systems: A survey, 2022.

Yu, J., Yin, H., Li, J., Gao, M., Huang, Z., and Cui, L. Enhancing social recommendation with adversarial graph convolutional networks. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3727–3739, 2022. doi: 10.1109/TKDE.2020.3033673.