HETEROGENEOUS GRAPH NODE CLASSIFICATION WITH MULTI-HOPS RELATION FEATURES

Xiaolong Xu[*] *Lingjuan Lyu*[†] *Hong Jin* *Weiqiang Wang* *Shuo Jia*

Ant Group
No. 569 Xixi Road, Hangzhou 310013, China
$\{yiyin.xxl, jinhong.jh, weiqiang.wwq, xuran.js\}$ @$antgroup.com$, $Lingjuan.Lv$@$sony.com$.

ABSTRACT

In recent years, knowledge graph (KG) has obtained many achievements in both research and industrial fields. However, most KG algorithms consider node embedding with only structure and node features, but not relation features. In this paper, we propose a novel Heterogeneous Attention (HAT) algorithm and use both node-based and path-based attention mechanisms to learn various types of nodes and edges on the KG. To better capture representations, multi-hop relation features are involved to generate edge embeddings and help the model obtain more semantic information. To capture a more complex representation, we design different encoder parameters for different types of nodes and edges in HAT. Extensive experiments validate that our HAT significantly outperforms the state-of-the-art methods on both the public datasets and a large-scale real-world fintech dataset [1].

***Index Terms*—** Graph Neural Networks, Heterogeneous graph, Node classification

1. INTRODUCTION

Knowledge graph (KG) is typically a multi-relational graph containing entities as nodes and relations as edges. Knowledge graphs (KGs) are important resources for a wide range of AI tasks. In recent years, large-scale knowledge graphs such as FreeBase [1], WordNet [2] and YAGO [3] have provided an effective basis for many applications such as semantic search, recommendation, and question answering.

KG is composed of triplets (the smallest group in KG). By aggregating information from triplets, KG could be used for node classification, link prediction, graph classification, and graph clustering. Many popular algorithms, such as Graph convolutional network (GCN) [4], Graph attention networks (GAT) [5], and GraphSAGE [6]), typically perform well in homogeneous graphs. However, they are not good at learning KG related tasks [7]. The main reason is that the KG consists of multiple node types and relation types. To address this issue, various methods have been proposed to learn the representation of KG [8]. Among them, most algorithms are designed to aggregate nodes based on the structure and node features. Some of them (HAN [7], HGT [9], RGCN [10]) ignore the relation feature, and others fail to learn the metapath embedding (the embedding represents the metapath with features) (WGCN [11], CompGCN [12]) or just learn the fixed embedding for metapaths (MAGNN [13]), instead of considering features on metapaths. Typically, in the real world, many features are lying on edges ($e.g.$ view frequency, watch duration, and public year, $et\ al.$).

To close this gap, we design a novel method based on metapath to aggregate neighbors' information. In our method, features lying on relations (metapaths) will be considered to generate KG node embedding. In summary, the contributions of this paper are as follows: 1) We propose a metapath sampling process to generate latent embedding from neighbor nodes, in order to enrich neighbor information; 2) We experiment with the node classification task on both the standard public datasets and a large-scale real-world fintech dataset. Extensive experimental results validate the effectiveness of our proposed HAT.

2. RELATED WORK

For KG node representation, there is a range of methods with or without metapaths. For node representation with metapaths, Metapath2Vec [14] is used to generate node embeddings which can be treated as features for the next step. Heterogeneous information networks to vector (HIN2vec) [15] conducts multiple prediction training tasks to learn representations of nodes and metapaths of a heterogeneous graph. Given a metapath, HERec [16] converts a heterogeneous graph into a homogeneous graph. Based on metapaths and neighbors, HERec applies DeepWalk to learn the node embedding of the target type. Moreover, the metapath is used to generate

[*]Corresponding author

[†]This work is completed during the tenure of Ant Group

[1]1. The data set does not contain any Personal Identifiable Information (PII) 2. The data set is desensitized and encrypted 3. Adequate data protection was carried out during the experiment to prevent the risk of data copy leakage, and the data set was destroyed after the experiment 4. The data set is only used for academic research, it does not represent any real business situation)

subgraphs for graph neural networks. Wang $et\ al.$ introduce heterogeneous graph attention network (HAN), which uses the metapath to sample neighbors and all input nodes will be of the same type. On the other hand, there are many methods without metapaths. For example, in RGCN, every relation (including self-loop) is aggregated in one block which will be added together to get the final embedding. Heterogeneous graph neural network (HetGNN) [17] uses the random walk to sample neighbors. HetGNN then aggregates messages by considering the effects of different node types. Ziniu $et\ al.$ propose heterogeneous graph transformer (HGT) [9], which designs node- and edge-type dependent parameters to characterize the heterogeneous attention over each edge.

Although all the above methods can learn node embedding, they have overlooked the relation importance in heterogeneous graphs. This issue can however be tackled by other methods. For example, Weighted-GCN (WGCN) [11] assigns a learnable scalar weight to each relation and multiplies an incoming "message" by this weight. However, WGCN only learns a scalar, rather than relation embeddings. Composition-based multi-relational graph convolutional network (CompGCN) [12] leverages a variety of entity-relation composition operations knowledge graph embedding (KGE) techniques and scales with the number of relations, which however fails to learn the metapath embedding. Metapath aggregated GNN for heterogeneous graph embedding (MAGNN) [13] investigates how to use middle nodes information lying on metapath to improve model performance. However, this algorithm only learns fixed relation embeddings without features on metapaths.

3. HETEROGENEOUS ATTENTION ALGORITHM

3.1. Metapath Sampling With Relation Features

Metapath neighbors and middle neighbors [7]. Given a specific metapath ρ and a node i in a heterogeneous graph, the metapath neighbor $\mathcal{N}_i^\rho$ of metapath ρ for node i is defined by the set of last nodes connected with i by metapath ρ. Let S_i^ρ denote the set of nodes lying on the metapath ρ of i. The **middle neighbor** is defined by $\mathcal{P}_i^\rho = S_i^\rho - \mathcal{N}_i^\rho - \{i\}$.

In Heterogeneous Attention (HAT), both pre-defined metapaths and original relations are taken to generate input subgraphs. For instance, in IMDB dataset, original relations ($e.g.$ actor, director) and metapaths ($e.g.$ MDM, MAM) are adopted to sample subgraphs as inputs. An example is shown in Figure 1. For the graph sampling process, all the original relations of neighbors will be considered to generate input subgraphs. By contrast, random sampling methods will miss some important information [18], leading to weak model performance.

3.2. Feature Transformation

For better representation, we project the original node features and edge features to a low dimension. Let D_n, D_e be the node and edge feature dimensions, d be the dimension of latent representations, for node i, we use the following equations for the feature transformation.

$$\begin{aligned} h_i^l &= W^{\tau,l}\mathbf{f}_i^l + b^{\tau,l} \\ r_{i,j}^{\rho,l} &= W^{\rho,l}\mathbf{e}_i^l + b^{\rho,l} \\ h_i^{\rho,l} &= \Phi(h_i^l, r_{i,j}^{\rho,l}) \end{aligned} \tag{1}$$

where $\mathbf{f}_i^l$ and $\mathbf{e}_i^l$ refer to node features and edge features, h_i^l and $r_{i,j}^{\rho,l}$ are latent vectors of the i-th node and the edge – the relation between the target node and i-th node in the l-th layer of the ρ relation. $W^{\tau,l} \in \mathbb{R}^{D_n \times d}$ and $W^{\rho,l} \in \mathbb{R}^{D_e \times d}$ stand for transformation weights of the node (type τ) and edge (relation ρ) in the l-th layer respectively.

The function Φ represents the concatenation function used to combine node and edge latent vectors. In fact, other functions ($e.g.$ linear transformation, element-wise addition) can be adopted as well. $h_i^{\rho,l}$ denotes the node hidden status of the l-th layer. The feature transformation stage will be processed in each subgraph for $\mathcal{N}_i^\rho$. But for the target node, $h_i^{\rho,l} = h_i^l$. In addition, when using pre-defined metapaths, for aggregating middle neighbors ($\mathcal{P}_i^\rho$), the calculation obeys the metapath sampling process shown in Figure 1.

3.3. Node Attention and Fusion

This process aims to find a vector that can represent the aggregated status of all neighbors (middle neighbors) in the same relation.

$$\mathbf{z}_i^{\rho,l} = \mathbf{F}_\rho(h_i^{\rho,l}, h_j^{\rho,l}) \tag{2}$$

Where $\mathbf{F}_\rho$ represents the aggregation function in the ρ relation (or metapath). $\mathbf{z}$ denotes the node's hidden status embedding after node fusion. After this step, for one specific relation, all nodes' information is treated as one vector which can represent all information of the target node, neighbor (middle neighbor) nodes, and relations between them. Here, we adopt the attention mechanism to integrate node embeddings into one vector.

$$\mathbf{z}_i^{\rho,l} = \sigma(\Big\|_{m=1}^{M} \sum_{j \in \mathcal{N}_i^\rho} \alpha_{i,j}^{\rho,l,m} h_j^{\rho,l}) \tag{3}$$

Where M is the multi-head number, and σ is an activation function. In our work, we use Eq. (3) to aggregate node embeddings when processing the fusion stage. To consider the impact of different neighbor (middle neighbor) nodes on the target node, all neighbors have calculated attention coefficients in a given relation ρ.

$$\alpha_{i,j}^{\rho,l,m} = \frac{exp(h_i^{\rho,l} \cdot h_j^{\rho,l})}{\sum_{k \in \mathcal{N}_i^\rho} exp(h_i^{\rho,l} \cdot h_k^{\rho,l})} \tag{4}$$

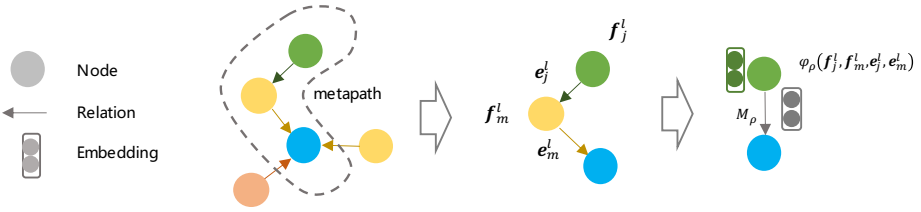

Fig. 1. Graph sampling: generating latent embedding for neighbor node. Different colors represent different node and relation types. Nodes and relations within dash line is the pre-defined metapath. $\mathbf{f}_j^l$ and $\mathbf{e}_j^l$ stand for node and edge features in l-th layer. In this method, $h_j^l = \varphi_\tau(\mathbf{f}_j^l, \mathbf{f}_m^l)$ and $r_{j,m}^{\rho,l} = \varphi_\rho(\mathbf{e}_j^l, \mathbf{e}_m^l)$. Here, h_j^l and $r_{j,m}^{\rho,l}$ are the same as in Eq. (1). $\varphi_\tau(\mathbf{f}_j^l, \mathbf{f}_m^l) = W^{\tau,l}[\mathbf{f}_j^l, \mathbf{f}_m^l] + b^{\tau,l}$ and $\varphi_\rho(\mathbf{e}_j^l, \mathbf{e}_m^l) = W^{\rho,l}[\mathbf{e}_j^l, \mathbf{e}_m^l] + b^{\rho,l}$.

Algorithm 1 Heterogeneous Attention (HAT)

Input: the graph $\mathbf{G} = (\mathcal{V}, \mathcal{E})$, Node features $\mathbf{f}_i$ and Edge features $\mathbf{e}_i$, the set of pre-defined metapaths and all original relations $\{\rho_1, \rho_2, ..., \rho_n\}$, Hops number N and multi-heads number M

Output: the result of node classification ($\hat{y}$), node embedding, relation embedding

while $n = 1, 2, ..., N$ **do**
  while ρ_i in $\{\rho_0, \rho_1, ..., \rho_n\}$ **do**
    Feature Transformation $\mathbf{f}_i^l$, $\mathbf{e}_i^l$;
    Feature Combination $h_i^{\rho,l}$ according to Eq. (1);
    while m=1,2,...,M **do**
      Computing node attention coefficients $\alpha_{i,j}^{\rho,l,m}$ adopting Eq. (4);
      Node embedding Fusion to get $\mathbf{z}_i^{\rho,l}$ as Eq. (3);
    end while
  end while
  while m=1,2,...,M **do**
    Calculating path attention coefficients $\beta_i^{\rho,l,m}$ based on Eq. (6).
    Getting $\mathbf{f}i^{l+1}$ as node embedding in this layer by using Eq. (5).
  end while
end while
MLP to calculate probability $\hat{y}_i$;
return probability $\hat{y}_i$, node- and relation embedding.

3.4. Path Attention

In the heterogeneous graph, there may exist multi-relations between two nodes. Based on this, the path-level attention is needed, which can reflect how the target node is impacted by different relations.

$$\mathbf{f}_i^{l+1} = \sigma(\Big\|_{m=1}^{M} \sum_{\rho} \beta_i^{\rho,l,m} \mathbf{z_i}^{\rho,l}) \tag{5}$$

$$\beta_i^{\rho,l,m} = \frac{exp(\mathbf{q}^{\rho T} \cdot \mathbf{z}_i^{C,l})}{\sum_{\rho'} exp(\mathbf{q}^{\rho' T} \cdot \mathbf{z}_i^{C,l})} \tag{6}$$

where $\beta_i^{\rho,l,m}$, $\mathbf{q}^{\rho T}$ are the path-attention coefficient and the attention vector for the ρ relation in the l-th layer, respectively. $\mathbf{z}_i^{C,l}$ is the concatenation of all $\mathbf{z}_i^{\rho,l}$. After this process, we can obtain the target node embedding $\mathbf{f}_i^{l+1}$.

4. EXPERIMENTS

4.1. Datasets

We evaluate our HAT on four datasets as listed in Table 1: DBLP, IMDB, ACM, and a real fintech dataset from Alipay. In financial service, one of the urgent needs is to predict whether one person or company is risky. If one person or company in the fintech KG suffers from credit risk, the service party will allocate fewer loans to this person or company. Based on this situation, we simplify this as a binary classification problem.

Different from other datasets, real fintech data contains edge features. Generally, DBLP, IMDB, and ACM are standard datasets that are used to evaluate GNNs. In this paper, although DBLP, IMDB, and ACM datasets have no edge features, their relation embeddings will be learned as well. Typically, there are two approaches to initialize different types of edge features. The first is to use trainable embeddings and the second is to use the pre-trained BERT (based on its metapath). Empirical results show that these two approaches have similar performance on public datasets, but for the real-world fintech dataset, the second method is better. For convenience, we select the second approach as the default initialization method.

4.2. Tasks and Evaluation

We also take binary- and multi-label to train all models. DBLP dataset is a multi-label classification problem, while IMBD and fintech datasets are binary node classification tasks. In our experiments, we use four classes (database, data mining, machine learning, information retrieval) for the DBLP dataset. The original IMDB dataset is a multi-class problem. Here we simplify it as a binary class (whether a movie belongs to the action class according to their genre). For the ACM dataset, labels are divided into three classes (database, wireless communication, data mining). As mentioned before, the fintech dataset mainly focuses on whether a company is

Table 1. Statistics of the Datasets

Dataset	Nodes	Node Types	Relations	Edge Types	Edge Feature	Node Feature	Training	Validation	Test
DBLP	27194	4	122393	3	-	334	800	400	2857
IMDB	12890	3	19120	2	-	1232	300	300	2687
ACM	8916	2	17549	2	-	1830	600	300	2125
Fintech Data	11352198	2	70732869	4	4	113	2521913	1260956	7569329

risky. Throughout this work, the Micro-F1 score is adopted as the evaluation metric for node classification.

4.3. Baselines and Parameters Setting

We compare our HAT with several state-of-the-art node classification algorithms from different perspectives, namely, homogeneous graph (GCN, GAT), heterogeneous graph (RGCN, HAN, HGT), and relation embedding (CompGCN, MAGNN). In order to compare all algorithms fairly, we keep all baseline models tuned, then choose the best result for comparison. Furthermore, to systematically analyze the effectiveness of HAT, we compare with the original HAT (HAT), HAT with metapath sampling subgraph as input (HAT (meta)), HAT with relation embedding (HAT (edge)), and HAT that adopts metapath and relation embedding (HAT (all)).

To obtain better performance for different methods, we use a grid search technique to select suitable parameters for both baselines and HAT. The range of hidden size, learning rate, and L2 coefficient are [32, 64, 128, 256], [0.0001, 0.0003, 0.001], [0.0001, 0.0003, 0.0005, 0.001], respectively. Some baselines (GAT, HAN, HGT, MAGNN, and HAT) need the multi-heads number, in this paper, we try [1, 3, 6, 8, 10]. We conduct both Rmsprop [19] or Adam [20] as the optimizer and select the best result. According to previous works [21], all methods have three graph hidden layers. We use a two-layer neural network to transfer the learned graph embedding for prediction.

4.4. Experimental Results

Our experimental results mainly aim to answer the following two research questions:

Does HAT perform better than the state-of-the-art methods? As validated in Table 2, we find that HAT with both metapath and edge embedding consistently outperforms all the other methods on all datasets. We summarize the potential reasons as follows. The self-governing transfer weights mechanism is designed to get node and relation information for the HAT algorithm. For the heterogeneous graph, the relation which brings different semantic information is also important for model performance. However, most previous works mainly focus on how to treat the node but neglect the relation, metapath features, or middle neighbors. Our experiments imply that features lying on metapaths and middle neighbors can largely improve the expressive power of the model.

Table 2. Experimental results. HAT (all) achieves the best performance.

Method	DBLP	IMDB	ACM	Fintech Data
GCN	92.13	57.77	88.31	90.43
GAT	92.05	57.54	87.24	90.26
RGCN	92.58	60.01	88.58	89.56
HAN	92.67	61.26	90.57	91.14
HGT	91.87	60.51	90.39	91.32
CompGCN	93.17	60.23	90.62	91.19
MAGNN	93.19	61.24	90.78	91.18
HAT	93.21	61.25	91.03	91.67
HAT (meta)	94.16	62.34	91.31	92.16
HAT (edge)	94.19	62.27	91.12	92.25
HAT (all)	**94.48**	**62.38**	**91.86**	**92.71**

Impact of metapath sampling and relation embedding. From the results reported by Table 2, on DBLP, IMDB, and ACM datasets, we discover that the HAT (edge) performs worse than HAT (meta) in most cases. The primary reason is that there is no more information lying on relations for those datasets. However, given extra information on relations (fintech dataset), HAT (edge) can outperform HAT (meta). Overall, it can be concluded that HAT (all) achieves the best performance. Not only because metapaths are adopted, but learning relation embedding with features also contributes. For example, as shown in Table 2, MAGNN is a strong baseline, the pure version of HAT (without edge information) is only slightly better than it. This improvement mainly comes from self-governing weights of node and edge types. However, given extra edge information (similar to MAGNN), HAT (edge) will outperform MAGNN with a larger margin, especially on the real fintech dataset.

5. CONCLUSION

In this paper, we propose a novel algorithm named HAT which can handle the relation embedding, no matter whether there are relation features lying on paths. Our method generalizes several existing methods. For example, when only using metapath to sample input subgraphs without considering relation features, HAT would be similar to MAGNN. Moreover, our method alleviates the problem of sparse features for nodes on public KGs. Through extensive experiments, we show the effectiveness of HAT over baseline algorithms and demonstrate its capability in learning multi-hop embeddings.

6. REFERENCES

[1] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. 2008, pp. 1247–1250, ACM.

[2] George A Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[3] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 697–706.

[4] Thomas N. Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," 2017.

[5] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio, "Graph attention networks," 2018.

[6] Hamilton Will, Ying Zhitao, and Leskovec Jure, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., pp. 1024–1034. Curran Associates and Inc., 2017.

[7] Wang, Xiao, Ji, Houye, Shi, Chuan, Wang, Bai, Ye, Yanfang, Cui, Peng, Yu, and Philip S, "Heterogeneous graph attention network," in *The World Wide Web Conference*, New York and NY and USA, 2019, WWW '19, p. 2022–2032, Association for Computing Machinery.

[8] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2021.

[9] Hu Ziniu, Dong Yuxiao, Wang Kuansan, and Sun Yizhou, "Heterogeneous graph transformer," in *Proceedings of The Web Conference 2020*, New York and NY and USA, 2020, WWW '20, p. 2704–2710, Association for Computing Machinery.

[10] Schlichtkrull Michael, Kipf Thomas N., and Bloem Peter, "Modeling relational data with graph convolutional networks," in *The Semantic Web*, Gangemi, Aldo, and Navigli Roberto, Eds., pp. 593–607. Springer International Publishing, Cham, 2018.

[11] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou, "End-to-end structure-aware convolutional networks for knowledge base completion," 2018.

[12] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar, "Composition-based multi-relational graph convolutional networks," 2020.

[13] Fu Xinyu, Zhang Jiani, Meng Ziqiao, and King Irwin, "Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding," *Proceedings of The Web Conference 2020*, Apr 2020.

[14] Dong Yuxiao, Chawla Nitesh V., and Swami Ananthram, "Metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York and NY and USA, 2017, KDD '17, p. 135–144, Association for Computing Machinery.

[15] Fu Tao-yang, Lee Wang-Chien, and Lei Zhen, "Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, New York and NY and USA, 2017, CIKM '17, p. 1797–1806, Association for Computing Machinery.

[16] C. Shi, B. Hu, W. X. Zhao, and P. S. Yu, "Heterogeneous information network embedding for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 2, pp. 357–370, 2019.

[17] Zhang Chuxu and Song Dongjin, "Heterogeneous graph neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery; Data Mining*, New York and NY and USA, 2019, KDD '19, p. 793–803, Association for Computing Machinery.

[18] Jure Leskovec and Christos Faloutsos, "Sampling from large graphs," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2006, KDD '06, p. 631–636, Association for Computing Machinery.

[19] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," 2012.

[20] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," 2017.

[21] Guohao Li, Matthias Müller, Guocheng Qian, Itzel C. Delgadillo, Abdulellah Abualshour, Ali K. Thabet, and Bernard Ghanem, "Deepgcns: Making gcns go as deep as cnns," 2019, vol. abs/1910.06849.