Chapter 4
Attribute-assisted Heterogeneous Graph Representation

Abstract The previous heterogeneous graph representation methods mainly focus on preserving the complex interactions and rich semantics into node representation. As a matter of fact, diverse types of nodes in heterogeneous graph are assisted with different attributes, providing valuable side information for depicting the characteristics of nodes. Integrating attribution information is also desired for heterogeneous graph representation in a real world application. Fortunately, heterogeneous graph neural networks naturally provide an alternative way to achieve this, meanwhile, have powerful representation ability. In this chapter, we introduce three attribute-assisted heterogeneous graph representation models including heterogeneous graph attention network (HAN), heterogeneous graph propagation network (HPN), and heterogeneous graph structure learning (HGSL), which simultaneously utilize both complex structural information and rich attribute information to learn node representation.

4.1 Introduction

Besides complex structures and rich semantics, real-world heterogeneous graphs (HGs) are usually associated with diverse types of attributes, called attribute-assisted HGs, providing valuable side information for depicting the characteristics of nodes. For example, in the ACM graph, the user's attribute mainly consists of name, age, and gender, while the paper's attribute usually involves a set of keywords. Ignoring such attributes may lead to sub-optimal heterogeneous graph representation. How to integrate attribution information into node representation is a realistic demand of real-world HG representation.

Heterogeneous graph neural network (HGNN), as a powerful deep learning based technology, naturally provides an elegant way to achieve this. Specifically, HGNN takes node attributes to initial node representation, and then aggregates diverse types of neighbors to update node representation, which simultaneously utilizes both complex structural information and rich attribute information to learn node

representations. In this chapter, we introduce three attribute-assisted heterogeneous graph representation models including **H**eterogeneous graph **A**ttention **N**etwork (named **HAN**), **H**eterogeneous graph **P**ropagation **N**etwork (named **HPN**), and **H**eterogeneous **G**raph **S**tructure **L**earning (named **HGSL**). HAN is a classical HGNN which leverages both node- and semantic-level attention to learn node representation in a hierarchical manner. To alleviate the deep degradation phenomenon (a.k.a, semantic confusion), HPN improves the aggregating process of HGNN via emphasizing the characteristic of each node. After that, to further discover and learn graph structures, HGSL jointly learns the heterogeneous graph and the GNN parameters for downstream tasks.

4.2 Heterogeneous Graph Attention Network

4.2.1 Overview

A recent research trend in deep learning is the attention mechanism, which deals with variable sized data and encourages the model to focus on the most salient parts of data. It has demonstrated the effectiveness in deep neural network framework and is widely applied to various applications, such as text analysis (1), knowledge graph (28), and image processing (37). Graph Attention Network (GAT) (33), a novel convolution-style graph neural network, leverages attention mechanism for the homogeneous graph which includes only one type of nodes or links.

Despite the success of attention mechanism in deep learning, it has not been considered in the graph neural network framework for heterogeneous graph. As a matter of fact, the real-world graph usually comes with multi-types of nodes and edges, which we uniformly call heterogeneous graph. Because the heterogeneous graph contains more comprehensive information and rich semantics, it has been widely used in many data mining tasks. Meta-path (31) is a widely used structure to capture the semantics. As can be seen in Fig. 4.1, depending on the meta-paths, the relation between nodes in the heterogeneous graph can have different semantics. Due to the complexity of heterogeneous graph, traditional graph neural network cannot be directly applied to heterogeneous graph.

Based on the above analysis, when designing graph neural network architecture with attention mechanism for heterogeneous graph, we need to address the following new requirements. (1) Heterogeneity of graph. The heterogeneity is an intrinsic property of heterogeneous graph, i.e., various types of nodes and edges. How to handle such complex structural information and preserve the diverse feature information simultaneously is an urgent problem that needs to be solved. (2) Semantic-level attention. Different meaningful and complex semantic information are involved in heterogeneous graph, which is usually reflected by meta-paths (31). Different meta-paths in heterogeneous graph may extract diverse semantic information. How to select the most meaningful meta-paths and fuse the semantic information for the

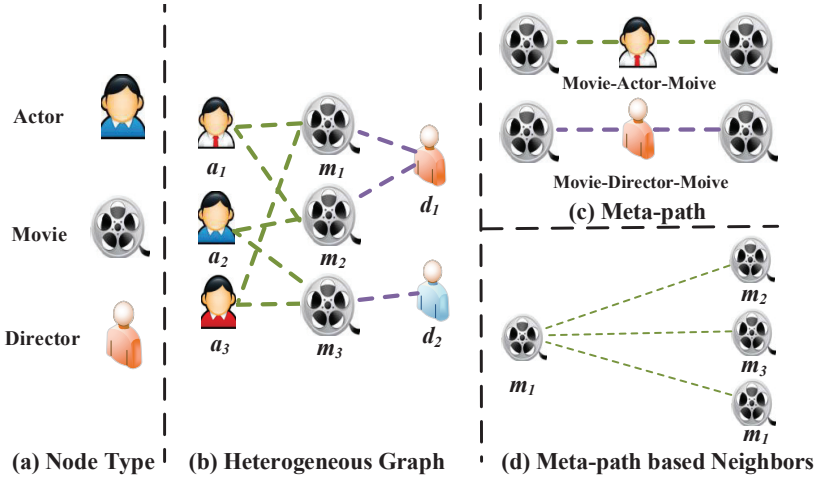


Fig. 4.1 An illustrative example of a heterogenous graph (IMDB). (a) Three types of nodes (i.e., actor, movie, director). (b) A heterogenous graph IMDB consists three types of nodes and two types of connections. (c) Two meta-paths involved in IMDB (i.e., Moive-Actor-Moive and Movie-Director-Movie). (d) Moive m_1 and its meta-path based neighbors (i.e., m_1, m_2 and m_3).

specific task is an open problem (23; 2; 29). (3) Node-level attention. In a heterogeneous graph, nodes can be connected via various types of relation, e.g., meta-path. Given a meta-path, each node has lots of meta-path based neighbors. How to distinguish the subtle difference of their neighbors and select some informative neighbors is required. For each node, node-level attention aims to learn the importance of meta-path based neighbors and assigns different attention values to them. Therefore, how to design a model which can discover the subtle differences of neighbors and learn their weights properly will be desired.

In this section, we introduce a novel **H**eterogeneous graph **A**ttention **N**etwork, named **HAN**, which considers both node-level and semantic-level attentions. In particular, given the node features as input, we use the type-specific transformation matrix to project different types of node features into the same space. Then the node-level attention is able to learn the attention values between the nodes and their meta-path based neighbors, while the semantic-level attention aims to learn the attention values of different meta-paths for the specific task in the heterogeneous graph. Based on the learned attention values in terms of the two levels, our model can get the optimal combination of neighbors and multiple meta-paths in a hierarchical manner, which enables the learned node representations to better capture the complex structure and rich semantic information in a heterogeneous graph. After that, the overall model can be optimized via backpropagation in an end-to-end manner.

4.2.2 The HAN Method

In this section, we provide more details about HAN. Specifically, this model follows a hierarchical attention structure: node-level attention → semantic-level attention. Fig. 4.2 presents the whole framework of HAN. First, we propose a node-level attention to learn the weight of meta-path based neighbors and aggregate them to get the semantic-specific node representation. After that, HAN can tell the difference of meta-paths via semantic-level attention and get the optimal weighted combination of the semantic-specific node representation for the specific task.

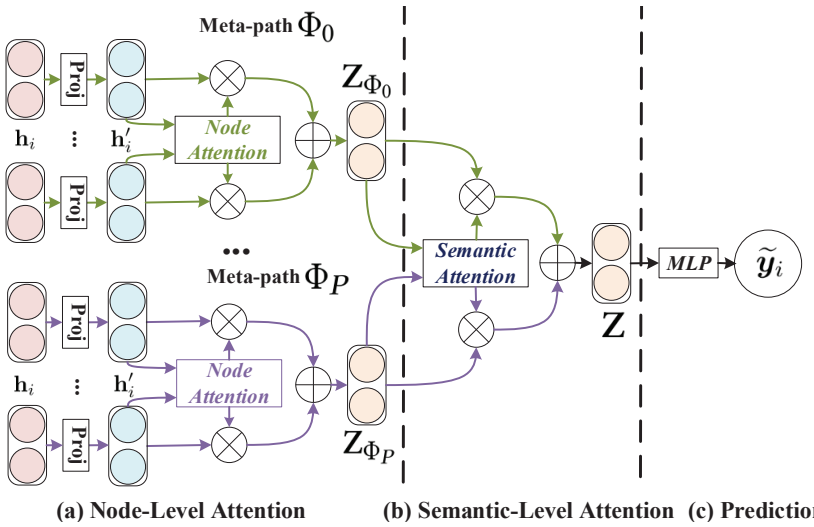


Fig. 4.2 The overall framework of the proposed HAN. (a) All types of nodes are projected into a unified feature space and the weight of meta-path based node pair can be learned via node-level attention. (b) Joint learning the weight of each meta-path and fuse the semantic-specific node representation via semantic-level attention. (c) Calculate the loss and end-to-end optimization for the proposed HAN.

4.2.2.1 Node-level Attention

Before aggregating the information from meta-path neighbors for each node, we should notice that the meta-path based neighbors of each node play a different role and show different importance in learning node representation for the specific task. Here we introduce node-level attention can learn the importance of meta-path based neighbors for each node in a heterogeneous graph and aggregate the representation of these meaningful neighbors to form a node representation.

Due to the heterogeneity of nodes, different types of nodes have different feature spaces. Therefore, for each type of nodes (e.g., node with type ϕ_i), we design the

type-specific transformation matrix M_{ϕ_i} to project the features of different types of nodes into the same feature space. Unlike (9), the type-specific transformation matrix is based on node-type rather than edge-type. The projection process can be shown as follows:

$$h'_i = M_{\phi_i} \cdot h_i, \tag{4.1}$$

where h_i and h'_i are the original and projected feature of node i, respectively. By type-specific projection operation, the node-level attention can handle arbitrary types of nodes.

After that, we leverage self-attention (32) to learn the weight among various kinds of nodes. Given a node pair (i, j) which are connected via meta-path Φ, the node-level attention e^{Φ}_{ij} can learn the importance e^{Φ}_{ij} which means how important node j will be for node i. The importance of meta-path based node pair (i, j) can be formulated as follows:

$$e^{\Phi}_{ij} = att_{node}(h'_i, h'_j; \Phi). \tag{4.2}$$

Here att_{node} denotes the deep neural network which performs the node-level attention. Given meta-path Φ, att_{node} is shared for all meta-path based node pairs. It is because there are some similar connection patterns under one meta-path. The above Eq. (4.2) shows that given meta-path Φ, the weight of meta-path based node pair (i, j) depends on their features. Please note that, e^{Φ}_{ij} is asymmetric, i.e., the importance of node i to node j and the importance of node j to node i can be quite difference. It shows node-level attention can preserve the asymmetry which is a critical property of heterogenous graph.

Then we inject the structural information into the model via masked attention which means we only calculate the e^{Φ}_{ij} for nodes $j \in \mathcal{N}^{\Phi}_i$, where $\mathcal{N}^{\Phi}_i$ denotes the meta-path based neighbors of node i (include itself). After obtaining the importance between meta-path based node pairs, we normalize them to get the weight coefficient α^{Φ}_{ij} via softmax function:

$$\alpha^{\Phi}_{ij} = softmax_j(e^{\Phi}_{ij}) = \frac{\exp\left(\sigma(a^{\mathrm{T}}_{\Phi} \cdot [h'_i \| h'_j])\right)}{\sum_{k \in \mathcal{N}^{\Phi}_i} \exp\left(\sigma(a^{\mathrm{T}}_{\Phi} \cdot [h'_i \| h'_k])\right)}, \tag{4.3}$$

where σ denotes the activation function, $\|$ denotes the concatenate operation and a_{Φ} is the node-level attention vector for meta-path Φ. As we can see from Eq. (4.3), the weight coefficient of (i, j) depends on their features. Also please note that the weight coefficient α^{Φ}_{ij} is asymmetric which means they make different contribution to each other. Not only because the concatenate order in the numerator, but also because they have different neighbors so the normalize term (denominator) will be quite difference.

Then, the meta-path based representation of node i can be aggregated by the neighbor's projected features with the corresponding coefficients as follows:

$$z^{\Phi}_i = \sigma\left(\sum_{j \in \mathcal{N}^{\Phi}_i} \alpha^{\Phi}_{ij} \cdot h'_j\right), \tag{4.4}$$

where z_i^{Φ} is the learned representation of node i for the meta-path Φ. To better understand the aggregating process of node-level, we also give a brief explanation in Fig. 4.3 (a). Every node representation is aggregated by its neighors. Since the attention weight α_{ij}^{Φ} is generated for single meta-path, it is semantic-specific and able to caputre one kind of semantic information.

Since heterogeneous graph present the scale-free property, the variance of graph data is quite high. To tackle the above challenge, we extend node-level attention to multihead attention so that the training process is more stable. Specifically, we repeat the node-level attention for K times and concatenate the learned representations as the semantic-specific representation:

$$z_i^{\Phi} = \Big\Vert_{k=1}^{K} \sigma\Big(\sum_{j \in \mathcal{N}_i^{\Phi}} \alpha_{ij}^{\Phi} \cdot h'_j \Big). \tag{4.5}$$

Given the meta-path set $\{\Phi_0, \Phi_1, \ldots, \Phi_P\}$, after feeding node features into node-level attention, we can obtain P groups of semantic-specific node representations, denoted as $\{Z_{\Phi_0}, Z_{\Phi_1}, \ldots, Z_{\Phi_P}\}$.

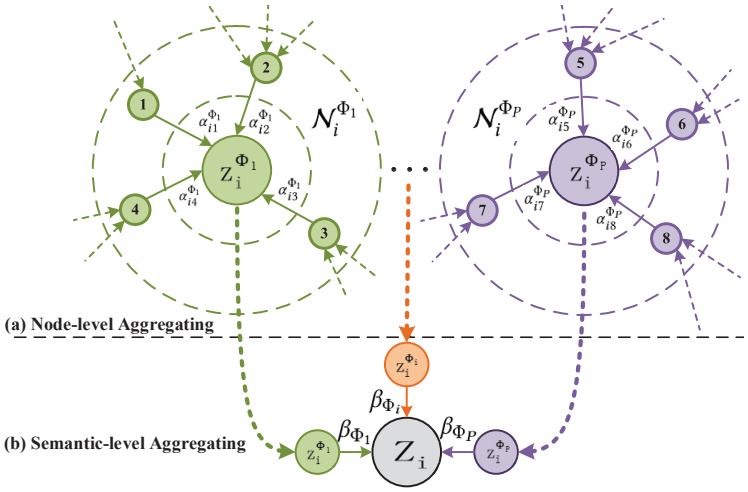


Fig. 4.3 Explanation of aggregating process in both node-level and semantic-level.

4.2.2.2 Semantic-level Attention

Generally, every node in a heterogeneous graph contains multiple types of semantic information and semantic-specific node representation can only reflect node from one aspect. To learn a more comprehensive node representation, we need to fuse multiple semantics which can be revealed by meta-paths. To address the challenge

of meta-path selection and semantic fusion in a heterogeneous graph, we propose a novel semantic-level attention to automatically learn the importance of different meta-paths and fuse them for the specific task. Taking P groups of semantic-specific node representations learned from node-level attention as input, the learned weights of each meta-path $(\beta_{\Phi_0}, \beta_{\Phi_1}, \ldots, \beta_{\Phi_P})$ can be shown as follows:

$$(\beta_{\Phi_0}, \beta_{\Phi_1}, \ldots, \beta_{\Phi_P}) = att_{sem}(Z_{\Phi_0}, Z_{\Phi_1}, \ldots, Z_{\Phi_P}). \tag{4.6}$$

Here att_{sem} denotes the deep neural network which performs the semantic-level attention. It shows that the semantic-level attention can capture various types of semantic information behind a heterogeneous graph.

To learn the importance of each meta-path, we first transform semantic-specific representation through a nonlinear transformation (e.g., one-layer MLP). Then we measure the importance of the semantic-specific representation as the similarity of transformed representation with a semantic-level attention vector q. Furthermore, we average the importance of all the semantic-specific node representation which can be explained as the importance of each meta-path. The importance of each meta-path, denoted as w_{Φ_i}, is shown as follows:

$$w_{\Phi_i} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} q^{\mathrm{T}} \cdot \tanh(W \cdot z_i^{\Phi} + b), \tag{4.7}$$

where W is the weight matrix, b is the bias vector, q is the semantic-level attention vector. Note that for the meaningful comparation, all above parameters are shared for all meta-paths and semantic-specific representation. After obtaining the importance of each meta-path, we normalize them via softmax function. The weight of meta-path Φ_i, denoted as β_{Φ_i}, can be obtained by normalizing the above importance of all meta-paths using softmax function,

$$\beta_{\Phi_i} = \frac{\exp(w_{\Phi_i})}{\sum_{i=1}^{P} \exp(w_{\Phi_i})}, \tag{4.8}$$

which can be interpreted as the contribution of the meta-path Φ_i for specific task. Obviously, the higher β_{Φ_i}, the more important meta-path Φ_i is. Note that for different tasks, meta-path Φ_i may has different weights. With the learned weights as coefficients, we can fuse these semantic-specific representations to obtain the final representation Z as follows:

$$Z = \sum_{i=1}^{P} \beta_{\Phi_i} \cdot Z_{\Phi_i}. \tag{4.9}$$

To better understand the aggregating process of semantic-level, we also give a brief explanation in Fig. 4.3 (b). The final representation is aggregated by all semantic-specific representation. Then we can apply the final representation to specific tasks and design different loss fuction. For semi-supervised node classification, we can minimize the Cross-Entropy over all labeled node between the ground-truth and the

prediction:

$$L = -\sum_{l \in \mathcal{Y}_L} Y^l \ln(C \cdot Z^l), \tag{4.10}$$

where C is the parameter of the classifier, $\mathcal{Y}_L$ is the set of node indices that have labels, Y^l and Z^l are the labels and representations of labeled nodes. With the guide of labeled data, we can optimize the proposed model via back propagation and learn the representations of nodes.

Table 4.1 Qantitative results (%) on the node classification task.

Datasets	Metrics	Training	DeepWalk	ESim	mp2vec	HERec	GCN	GAT	HAN$_{nd}$	HAN$_{sem}$	HAN
ACM	Macro-F1	20%	77.25	77.32	65.09	66.17	86.81	86.23	88.15	89.04	**89.40**
		40%	80.47	80.12	69.93	70.89	87.68	87.04	88.41	89.41	**89.79**
		60%	82.55	82.44	71.47	72.38	88.10	87.56	87.91	**90.00**	89.51
		80%	84.17	83.00	73.81	73.92	88.29	87.33	88.48	90.17	**90.63**
	Micro-F1	20%	76.92	76.89	65.00	66.03	86.77	86.01	87.99	88.85	**89.22**
		40%	79.99	79.70	69.75	70.73	87.64	86.79	88.31	89.27	**89.64**
		60%	82.11	82.02	71.29	72.24	88.12	87.40	87.68	**89.85**	89.33
		80%	83.88	82.89	73.69	73.84	88.35	87.11	88.26	89.95	**90.54**
DBLP	Macro-F1	20%	77.43	91.64	90.16	91.68	90.79	90.97	91.17	92.03	**92.24**
		40%	81.02	92.04	90.82	92.16	91.48	91.20	91.46	92.08	**92.40**
		60%	83.67	92.44	91.32	92.80	91.89	90.80	91.78	92.38	**92.80**
		80%	84.81	92.53	91.89	92.34	92.38	91.73	91.80	92.53	**93.08**
	Micro-F1	20%	79.37	92.73	91.53	92.69	91.71	91.96	92.05	92.99	**93.11**
		40%	82.73	93.07	92.03	93.18	92.31	92.16	92.38	93.00	**93.30**
		60%	85.27	93.39	92.48	93.70	92.62	91.84	92.69	93.31	**93.70**
		80%	86.26	93.44	92.80	93.27	93.09	92.55	92.69	93.29	**93.99**
IMDB	Macro-F1	20%	40.72	32.10	41.16	41.65	45.73	49.44	49.78	**50.87**	50.00
		40%	45.19	31.94	44.22	43.86	48.01	50.64	52.11	50.85	**52.71**
		60%	48.13	31.68	45.11	46.27	49.15	51.90	51.73	52.09	**54.24**
		80%	50.35	32.06	45.15	47.64	51.81	52.99	52.66	51.60	**54.38**
	Micro-F1	20%	46.38	35.28	45.65	45.81	49.78	55.28	54.17	55.01	**55.73**
		40%	49.99	35.47	48.24	47.59	51.71	55.91	56.39	55.15	**57.97**
		60%	52.21	35.64	49.09	49.88	52.29	56.44	56.09	56.66	**58.32**
		80%	54.33	35.59	48.81	50.99	54.61	56.97	56.38	56.49	**58.51**

4.2.3 Experiments

4.2.3.1 Experimental Settings

Datasets The experiments are conducted on three heterogeneous graphs including DBLP[1], ACM[2], and IMDB[3].

[1] https://dblp.uni-trier.de

[2] http://dl.acm.org/

[3] https://www.kaggle.com/carolzhangdc/imdb-5000-moviedataset

Baselines We compare with some state-of-art baselines, include the (heterogeneous) graph representation methods (i.e., DeepWalk (26), ESim (29), metapath2vec (4), and HERec (30)) and graph neural network based methods (i.e., GCN (20) and GAT (33)), to verfify the effectiveness of the proposed HAN. To verify the effectiveness of our node-level attention and semantic-level attention, respectively, we also test two variants of HAN (i.e., HAN_{nd} and HAN_{sem}).

Implementation Details For the proposed HAN, we randomly initialize parameters and optimize the model with Adam (19). For the proposed HAN, we set the learning rate to 0.005, the regularization parameter to 0.001, the dimension of the semantic-level attention vector q to 128, the number of attention head K to 8, the dropout of attention to 0.6. And we use early stopping with a patience of 100, i.e. we stop training if the validation loss does not decrease for 100 consecutive epochs. To make our experiments repeatable, we make our dataset and codes publicly available at website[4].

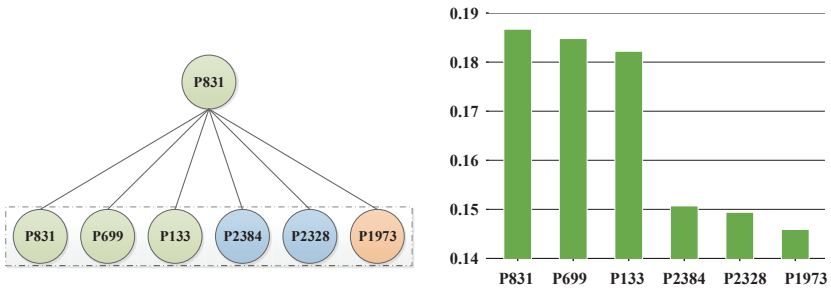
4.2.3.2 Classification

Here we employ KNN classifier with $k = 5$ to perform node classification. Since the variance of graph-structured data can be quite high, we repeat the process for 10 times and report the averaged *Macro-F1* and *Micro-F1* in Table 4.1.

Based on Table 4.1, we can see that HAN achieves the best performance. For traditional heterogeneous graph representation method, ESim which can leverage multiple meta-paths performs better than metapath2vec. Generally, graph neural network based methods which combine the structure and feature information, e.g., GCN and GAT, usually perform better. To go deep into these methods, compared to simply average over node neighbors, e.g., GCN and HAN_{nd}, GAT and HAN can weigh the information properly and improve the performance of the learned representation. Compared to GAT, the proposed HAN, which designs for heterogeneous graph, captures the rich semantics successfully and shows its superiority. Also, without node-level attention (HAN_{nd}) or semantic-level attention (HAN_{sem}), the performance becomes worse than HAN, which indicates the importance of modeling the attention mechanism on both of the nodes and semantics. Note that in ACM and IMDB, HAN improves classification results more significantly than in DBLP. Mainly because *APCPA* is the much more important than the rest meta-paths.

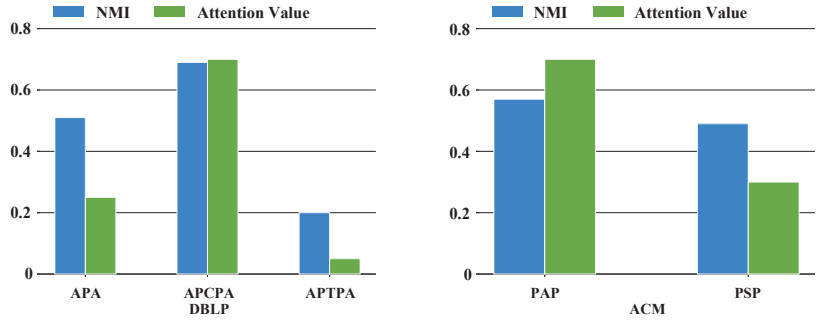
Through the above analysis, we can find that the proposed HAN achieves the best performance on all datasets. The results demonstrate that it is quite important to capture the importance of nodes and meta-paths in heterogeneous graph analysis.

[4] https://github.com/Jhy1993/HAN



(a) Meta-path based neighbors of P831 (b) Attention values of P831's neighbors

Fig. 4.4 Meta-path based neighbors of node P831 and corresponding attention values (Different colors mean different classes, e.g., *green* means Data Mining, *blue* means Database, *orange* means Wireless Communication).



(a) NMI values on DBLP (b) NMI values on ACM

Fig. 4.5 Performance of single meta-path and corresponding attention value.

4.2.3.3 Analysis of Hierarchical Attention Mechanism

A salient property of HAN is the incorporation of the hierarchical mechanism, which takes the importance of node neighbors and meta-paths into consideration in learning representative representation. Recall that we have learned the node-level attention weight α_{ij}^{Φ} and the semantic-level attention weight β_{Φ_i}. To better understand the importance of the neighbors and meta-paths, we provide a detailed analysis on the hierarchical attention mechanism.

Analysis of node-level attention. As mentioned before, given a specific task, our model can learn the attention values between nodes and its neighbors in a meta-path. Some important neighbors which are useful for the specific task tend to have larger

attention values. Here we take the paper P831 [5] in ACM dataset as an illustrative example. Given a meta-path Paper-Author-Paper which describes the co-author of different papers, we enumerate the meta-path based neighbors of paper P831 and their attention values are shown in Fig. 4.4. From Fig. 4.4(a), we can see that P831 connects to P699 [6] and P133 [7], which all belong to *Data Mining*; conects to P2384 [8] and P2328 [9] while P2384 and P2328 both belong to *Database*; connects to P1973 [10] while P1973 belongs to *Wireless Communication*. From Fig. 4.4(b), we can see that paper P831 gets the highest attention value from node-level attention which means the node itself plays the most important role in learning its representation. It is reasonable because all information supported by neighbors are usually viewed as a kind of supplementary information. Beyond itself, P699 and P133 get the second and third largest attention values. This is because P699 and P133 also belong to *Data Mining* and they can make significant contribution to identify the class of P831. The rest neighbors get minor attention values that because they do not belong to *Data Mining* and cannot make important contribution to identify the P831's class. Based on the above analysis, we can see that the node-level attention can tell the difference among neighbors and assigns higher weights to some meaningful neighbors.

Analysis of semantic-level attention. As mentioned before, the proposed HAN can learn the importance of meta-paths for the specific task. To verify the ability of semantic-level attention, taking DBLP and ACM as examples, we report the clustering results (*NMI*) of single meta-path and corresponding attention values in Fig. 4.5. Obviously, there is a positive correlation between the performance of a single meta-path and its attention value. For DBLP, HAN gives *APCPA* the largest weight, which means that HAN considers the *APCPA* as the most critical meta-path in identifying the author's research area. It makes sense because the author's research area and the conferences they submitted are highly correlated. For example, some natural language processing researchers mainly submit their papers to ACL or EMNLP, whereas some data mining researchers may submit their papers to KDD or WWW. Meanwhile, it is difficult for *APA* to identify the author's research area well. If we treat these meta-paths equally, e.g., HAN_{sem}, the performance will drop significantly. Based on the attention values of each meta-path, we can find that the meta-path *APCPA* is much more useful than *APA* and *APTPA*. So even the proposed HAN can fuse them, *APCPA* still plays a leading role in identifying the author's research area while *APA* and *APTPA* do not. It also explains why the performance

[5] Xintao Wu, Daniel Barbara, Yong Ye. Screening and Interpreting Multi-item Associations Based on Log-linear Modeling, KDD'03

[6] Xintao Wu, Jianpin Fan, Kalpathi Subramanian. B-EM: a classifier incorporating bootstrap with EM approach for data mining, KDD'02

[7] Daniel Barbara, Carlotta Domeniconi, James P. Rogers. Detecting outliers using transduction and statistical testing, KDD'06

[8] Walid G. Aref, Daniel Barbara, Padmavathi Vallabhaneni. The Handwritten Trie: Indexing Electronic Ink, SIGMOD'95

[9] Daniel Barbara, Tomasz Imielinski. Sleepers and Workaholics: Caching Strategies in Mobile Environments, VLDB'95

[10] Hector Garcia-Holina, Daniel Barbara. The cost of data replication, SIGCOMM'81

of HAN in DBLP may not be as significant as in ACM and IMDB. We get the similar conclusions on ACM. For ACM, the results show that HAN gives the most considerable weight to *PAP*. Since the performance of *PAP* is slightly better than *PSP*, so HAN_{sem} can achieve good performance by simple average operation. We can see that semantic-level attention can reveal the difference between these meta-paths and weights them adequately.

The detailed method description and validation experiments can been seen in (35).

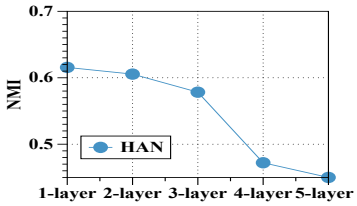
4.3 Heterogeneous Graph Propagation Network

4.3.1 Overview

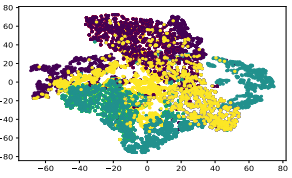
Recently, several HGNNs (35; 13; 38; 13) have been proposed to better analyze heterogeneous graphs, which usually follow two step aggregating process in a hierarchical manner: aggregate neighbors via single meta-path in node-level and then aggregate rich semantics via multiple meta-paths in semantic-level. When applying HGNNs in practice, we find an important phenomenon, called *semantic confusion*. Similar to over-smoothing in homogeneous GNNs (18), semantic confusion means HGNNs inject confused semantics extracted via multiple meta-paths into node representation, which makes the learned node representation indistinguishable and leads to worse performance with more hidden layers. Fig. 4.6 shows the clustering performance of HAN on ACM academic graph (35). It clearly displays that with the growth of model depth, the performance of HGNNs is getting worse and worse.

Semantic confusion makes HGNNs hard to become a really deep model, which severely limits their representation capabilities and hurts the performance of downstream tasks. Alleviating the semantic confusion phenomenon to build a more powerful deeper HGNNs is an urgent problem.

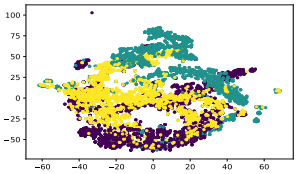
In this section, we theoretically analyze the semantic confusion in HGNNs and prove that HGNNs and multiple meta-paths based random walk (22) are essentially equivalent. Then we propose a novel **H**eterogeneous Graph **P**ropagation **N**etwork (HPN) to alleviate semantic confusion from the perspective of multiple meta-paths based random walk, which mainly consists of two parts: semantic propagation mechanism and semantic fusion mechanism. Besides aggregating information from meta-path based neighbors, the semantic propagation mechanism also absorbs node's local semantics with a proper weight. So even with more hidden layers, semantic propagation mechanism can capture the characteristics of each node rather than inject confused semantics into node representation and thus alleviates semantic confusion. The semantic fusion mechanism aims to learn the importance of meta-paths and fuses them for comprehensive node representation.



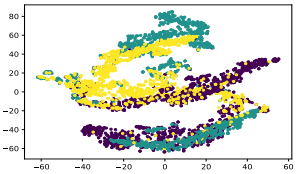
(a) Cluster results of HAN with 1, 2, 3, 4, 5 layers.



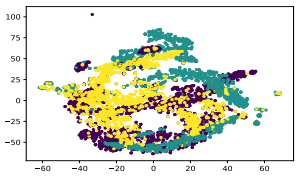
(b) Paper representation via 1-layer HAN



(c) Paper representation via 2-layer HAN



(d) Paper representation via 3-layer HAN



(e) Paper representation via 4-layer HAN

Fig. 4.6 The clustering results and visualization of paper representations via HAN with different layers. Each point denotes one paper and corresponding color indicates the label (i.e., research areas). With the growth of model depth, semantic confusion happens which means the learned node representations become indistinguishable. For example, the paper representations belonging to different research areas which are learned via 1-layer HAN located in different positions, while the paper representations learned via 4-layer HAN mixed together.

4.3.2 Semantic Confusion Analysis

We first give a brief review of HGNNs, and then prove that HGNNs and multiple meta-paths based random walk are essentially equivalent. Lastly, we explain why semantic confusion happens from the perspective of limit distribution of multiple meta-paths based random walk.

4.3.2.1 Heterogeneous Graph Neural Network

As shown in Fig. 4.3, HGNNs (e.g., HAN) aggregate information from multiple meta-paths and update node representation in both node-level and semantic-level. Specially, as shown in Fig. 4.3(a), given a meta-path Φ_1 and node i, node-level attention in HAN aggregates the meta-path Φ_1 based neighbors $\{1,2,3,4\}$ with attentions $\{\alpha_{i1}^{\Phi_1},\alpha_{i2}^{\Phi_1},\alpha_{i3}^{\Phi_1},\alpha_{i4}^{\Phi_1}\}$ to learn the semantic-specific node representation $z_i^{\Phi_1}$ for node i. Given one meta-path Φ, the node-level aggregating is defined as:

$$\begin{aligned} &Z^{\Phi,0}=X,\\ &Z^{\Phi,1}=\sigma\left(\boldsymbol{\alpha}^{\Phi,0}\cdot Z^{\Phi,0}\right),\\ &\cdots,\\ &Z^{\Phi}=Z^{\Phi,k}=\sigma\left(\boldsymbol{\alpha}^{\Phi,k-1}\cdot Z^{\Phi,k-1}\right), \end{aligned} \tag{4.1}$$

where X denotes node feature matrix, where the i-th row corresponds to the i-th node. And σ is an activate function, the element $\alpha_{ij}^{\Phi,k}$ of $\boldsymbol{\alpha}^{\Phi,k}$ denotes the learned attention weight between meta-path based node pair (i,j) via node-level attention by the k-th layer. Note that $\boldsymbol{\alpha}^{\Phi,k}$ is a (row-normalized) probability matrix and $Z^{\Phi,k}$ denotes the learned representation matrix by the k-th layer, where the i-th row corresponds to the i-th node. As shown in Fig. 4.3(b), given a node i and a set of meta-paths $\{\Phi_1,\Phi_2,\cdots,\Phi_P\}$, semantic-level aggregating in HAN fuses P semantic-specific node representations $\left\{z_i^{\Phi_1},\cdots,z_i^{\Phi_P}\right\}$ with attentions $\left\{\beta_{\Phi_1},\cdots,\beta_{\Phi_P}\right\}$ to get the final representation z_i for node i. The semantic-level aggregating is shown as follows:

$$Z=\sum_{p=1}^{P}\beta_{\Phi_p}\cdot Z^{\Phi_p}, \tag{4.2}$$

where Z denotes the final node representation.

4.3.2.2 Relationship between HGNNs and Multiple Meta-paths based Random Walk

As a classical heterogeneous graph algorithm, multiple meta-paths based random walk (22) mainly contains: single meta-path based random walk and multiple meta-path combinations. Given a meta-path Φ, we have the meta-path based probability matrix M^{Φ} whose element M_{ij}^{Φ} denotes the transition probability from node i to j via meta-path Φ. Then, the k-step single meta-path based random walk is defined as:

$$\boldsymbol{\pi}^{\Phi,k}=M^{\Phi}\cdot\boldsymbol{\pi}^{\Phi,k-1}, \tag{4.3}$$

where $\boldsymbol{\pi}^{\Phi,k}$ denotes the distribution of k-step single meta-path based random walk. Considering a set of meta-paths $\{\Phi_1,\Phi_2,\cdots,\Phi_P\}$ and their weights $\left\{w_{\Phi_1},w_{\Phi_2},\cdots,w_{\Phi_P}\right\}$, the k-step multiple meta-paths based random walk is defined as:

$$\boldsymbol{\pi}^k = \sum_{p=1}^{P} w_{\Phi_p} \cdot \boldsymbol{\pi}^{\Phi_p,k}, \tag{4.4}$$

where $\boldsymbol{\pi}^k$ denotes the distribution of k-step multiple meta-paths based random walk. For k-step single meta-path based random walk:

Theorem 1. *Assuming a heterogeneous graph is aperiodic and irreducible, if we take the limit $k \to \infty$, then k-step meta-path based random walk will converge to a meta-path specific limit distribution $\boldsymbol{\pi}^{\Phi,\lim}$ which is independent of nodes:*

$$\boldsymbol{\pi}^{\Phi,\lim} = M^{\Phi} \cdot \boldsymbol{\pi}^{\Phi,\lim}. \tag{4.5}$$

Different nodes connected via some relationships will influence each other and (18) demonstrates the influence distribution between two nodes is proportional to random walk distribution, shown as the following theorem:

Theorem 2 ((18)). *For the aggregation models (e.g., graph neural networks) on homogeneous graph, if the graph is aperiodic and irreducible, then the influence distribution I_i of node i is equivalent, in expectation, to the k-step random walk distribution.*

By Theorems 1 and 2, we conclude the influence distribution revealed by single meta-path based random walk is independent of nodes. Comparing Eq. 4.1 with Eq. 4.3, we find they both propagate and aggregate information via meta-path Φ. The difference is that $\boldsymbol{\alpha}^{\Phi,k}$ is a parameter matrix learned via node-level attention, while M^{Φ} is a predefined matrix. Since M^{Φ} and $\boldsymbol{\alpha}^{\Phi,k}$ are both probability matrix, they are actually meta-path related Markov Chain. So we find that node-level aggregation in HGNNs is essentially equivalent to meta-path based random walk if activate function is a linear function. So, if we stack infinite layers in node-level aggregating, the learned node representations Z^{Φ} will only be influenced by the meta-path Φ and therefore are independent of nodes. So the learned node representations cannot capture the characteristics of each node and therefore are indistinguishable. For k-step multiple meta-paths based random walk, we have:

Theorem 3. *Assuming k-step single meta-path based random walk is independent of each other, if we take the limit $k \to \infty$, then the limit distribution of k-step multiple meta-paths based random walk is a weighted combination of single meta-path based random walk limit distribution, shown as follows:*

$$\boldsymbol{\pi}^{\lim} = \sum_{p=1}^{P} w_{\Phi_p} \cdot \boldsymbol{\pi}^{\Phi_p,\lim}. \tag{4.6}$$

By Theorems 2 and 3, we conclude the influence distribution revealed by multiple meta-paths based random walk is independent of nodes. Comparing Eq. 4.2 with Eq. 4.4, we can see that they both combine multiple meta-paths according to their weights. The difference is that semantic-level aggregating in HAN leverages neural network to learn the weight of meta-path β_{Φ_p}, while multiple meta-paths based

random walk assigns predefined weight w_{Φ_p} to meta-path Φ_p by hand. Recall that in node-level aggregation, the node representations learned via single meta-path cannot capture the characteristics of each node and therefore are indistinguishable. In semantic-level aggregation, HGNNs fuse multiple node representations learned via multiple node-level aggregations with semantic-wise weights. In summary, the final node representations learned via node- and semantic-level only influenced by a set of meta-paths and still remain indistinguishable, which is the critical limitation of HGNNs and leads to semantic confusion.

4.3.3 The HPN Method

4.3.3.1 Semantic Propagation Mechanism

Given one meta-path Φ, the semantic propagation mechanism $\mathcal{P}_\Phi$ first projects node into semantic space via semantic projection function f_Φ. Then, it aggregates information from meta-path based neighbors via semantic aggregation function g_Φ to learn semantic-specific node representation, shown as follows:

$$Z^\Phi = \mathcal{P}_\Phi(X) = g_\Phi(f_\Phi(X)), \tag{4.7}$$

where X denotes initial feature matrix and Z^Φ denotes semantic-specific node representation. To handle heterogeneity graph, the semantic projection function f_Φ projects node into semantic space, shown as follows:

$$H^\Phi = f_\Phi(X) = \sigma(X \cdot W^\Phi + b^\Phi), \tag{4.8}$$

where H^Φ is the projected node feature matrix, W^Φ and b^Φ denote weight matrix and bias vector for meta-path Φ, respectively. To alleviate semantic confusion, we design semantic aggregation function g_Φ, shown as follows:

$$Z^{\Phi,k} = g_\Phi(Z^{\Phi,k-1}) = (1-\gamma) \cdot M^\Phi \cdot Z^{\Phi,k-1} + \gamma \cdot H^\Phi, \tag{4.9}$$

where $Z^{\Phi,k}$ denotes node representation learned by k-th layer semantic propagation mechanism and we take it as the semantic-specific node representation Z^Φ. Note that H^Φ reflects the characteristics of each node in meta-path Φ (also can be viewed as $Z^{\Phi,0}$) and $M^\Phi \cdot Z^{\Phi,k-1}$ means aggregating information from meta-path based neighbors. Here γ is a weight scalar which indicates the importance of characteristic of node in aggregating process.

Why semantic aggregation function g_Φ works. Here we establish the relationship between semantic aggregation function g_Φ and k-step meta-path based random walk with restart. k-step meta-path based random walk with restart for node i is defined as:

$$\boldsymbol{\pi}^{\Phi,k}(\boldsymbol{i}) = (1-\gamma) \cdot M^\Phi \cdot \boldsymbol{\pi}^{\Phi,k-1}(\boldsymbol{i}) + \gamma \cdot \boldsymbol{i}, \tag{4.10}$$

where $\boldsymbol{i}$ is a one-hot vector of node i, γ means the restart probability. For k-step meta-path based random walk with restart:

Theorem 4. *Assuming a heterogeneous graph is aperiodic and irreducible, if we take the limit $k \to \infty$, then* k*-step meta-path based random walk with restart will converge to $\boldsymbol{\pi}^{\Phi,\lim}(\boldsymbol{i})$ which is related to the start node i:*

$$\boldsymbol{\pi}^{\Phi,\lim}(\boldsymbol{i}) = \gamma \cdot (I - (1-\gamma) \cdot M^{\Phi})^{-1} \cdot \boldsymbol{i}. \tag{4.11}$$

By Theorems 2 and 4, we conclude that the influence distribution revealed by meta-path based random walk with restart is related to nodes. Comparing Eq. 4.9 to Eq. 4.10, we find they both emphasis node's local semantics with a proper weight γ. By Theorem 4, we can see that the semantic aggregation function g_{Φ} absorbs node's local semantics and makes semantic-specific node representation $Z^{\Phi,k}$ distinguish from each other even if we take the limit $k \to \infty$. So semantic propagation mechanism can alleviate the semantic confusion.

4.3.3.2 Semantic Fusion Mechanism

Generally, every node in a heterogeneous graph contains multiple types of semantic information and semantic-specific node representation can only reflect node from one aspect. To describe node more comprehensively, we leverage multiple meta-paths to capture rich semantics and describe node from different aspects.

Given a set of meta-paths $\{\Phi_1, \Phi_2, \cdots, \Phi_P\}$, we have P group semantic-specific node representations $\left\{Z^{\Phi_1}, Z^{\Phi_2}, \cdots, Z^{\Phi_P}\right\}$. Then, we propose the semantic fusion mechanism $\mathcal{F}$ to fuse them for the specific task. Taking P groups of semantic-specific node representations learned from semantic propagation mechanism as input, the final node representation Z learned via semantic fusion mechanism $\mathcal{F}$, shown as follows:

$$Z = \mathcal{F}(Z^{\Phi_1}, Z^{\Phi_2}, \cdots, Z^{\Phi_P}). \tag{4.12}$$

Intuitively, not all meta-paths should be treated equally. So semantic fusion mechanism should be able to tell the difference of meta-paths and assign different weights to them. To learn the importance of meta-paths, we project each semantic-specific node representation into the same latent space and adopt semantic fusion vector q to learn the importance of meta-paths. The importance of meta-path Φ_p, denoted as w_{Φ_p}, is defined as:

$$w_{\Phi_p} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} q^{\mathrm{T}} \cdot \tanh(W \cdot z_i^{\Phi_p} + b), \tag{4.13}$$

where W and b denote weight matrix and bias vector, respectively, which are shared for all meta-paths. Note that all parameters in semantic fusion mechanism are shared for all nodes and semantics. After obtaining the importance of meta-paths, we normalize them via softmax function to get the weight of each meta-path. The weight of meta-path Φ_p, denoted as β_{Φ_p}, is defined as:

$$\beta_{\Phi_p} = \frac{\exp(w_{\Phi_p})}{\sum_{p=1}^{P} \exp(w_{\Phi_p})}. \tag{4.14}$$

Obviously, the higher β_{Φ_p}, the more important meta-path Φ_p is. With the learned weights as coefficients, we can fuse P semantic-specific representations to obtain the final representation Z as follows:

$$Z = \sum_{p=1}^{P} \beta_{\Phi_p} \cdot Z^{\Phi_p}. \tag{4.15}$$

Then we can optimize the whole model for the specific task and learn the final node representation. Note that semantic fusion mechanism is quite flexible and be optimized for various types of tasks. For different tasks, each semantic may make different contribution which means β_{Φ_p} may change a lot. For semi-supervised node classification, we calculate Cross-Entropy and update parameters in HPN:

$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} Y_l \cdot \ln(Z_l \cdot C), \tag{4.16}$$

where C is a projection matrix which projects the node representation as a node label vector, $\mathcal{Y}_L$ is the set of labeled nodes, Y_l and Z_l are the label vector and representation of the labeled node l, respectively.

For unsupervised node recommendation, we leverage BPR loss with negative sampling (34) to update parameters in HPN:

$$\mathcal{L} = -\sum_{(u,v)\in\Omega} \log\sigma\left(z_u^\top z_v\right) - \sum_{(u,v')\in\Omega^-} \log\sigma\left(-z_u^\top z_{v'}\right), \tag{4.17}$$

where $(u, v) \in \Omega$ and $(u, v') \in \Omega^-$ denote the set of observed (positive) node pairs and the set of negative node pairs sampled from all unobserved node pairs, respectively.

4.3.4 Experiments

4.3.4.1 Experimental Settings

Datasets Three datasets including Yelp[11], ACM[12], and IMDB[13], and MovieLens[14] (ML for short) are used to evaluate the proposed model.

Baselines HPN are compared with some state-of-the-art baselines, including the (heterogeneous) graph representation including metapath2vec (4) and HERec (30),

[11] https://www.yelp.com

[12] http://dl.acm.org/

[13] https://www.kaggle.com/carolzhangdc/imdb-5000-moviedataset

[14] https://grouplens.org/datasets/movielens/

and (heterogeneous) GNNs including GCN (20), GAT (33), PPNP (21), HAN (35), MEIRec (6), MAGNN (8), and HGT (14) to verify the effectiveness of the proposed HPN. Meanwhile, we also test two variants of HPN (i.e., HPN$_{pro}$ and HPN$_{fus}$) to verify the effectiveness of different parts in our model.

4.3.4.2 Clustering

Following the previous work (35), we get the learned node representations of all models via feed forward, and then leverage classical node clustering to test their effectivenesses. Here we utilize the K-Means to perform node clustering and the number of clusters K is set to the number of classes, then select *NMI* and *ARI* to evaluate the clustering task and report the averaged results of 10 runs in Table 4.2.

As can be seen, the proposed HPN performs significantly better than all baselines. It shows the importance of alleviating semantic confusion in HGNNs. We also find that graph neural networks always perform better than graph representation methods. Moreover, heterogeneous graph neural networks including HAN, MEIRec, HGT, MAGNN, and HPN outperform homogeneous GNNs because they can capture rich semantics and describe the characteristic of node more comprehensively. Note that the performance of HPN$_{pro}$ and HPN$_{fus}$ both show different degradations, which imply the importance of semantic propagation mechanism and semantic fusion mechanism. Based on the above analysis, we can find that the proposed HPN can propagate and fuse semantic information effectively and shows significant improvements.

Table 4.2 Qantitative results (%) on the node clustering task. The larger values, the better performace.

Datasets	Metrics	mp2vec	HERec	GCN	GAT	PPNP	MEIRec	HAN	HGT	MG	HPN$_{pro}$	HPN$_{fus}$	HPN
Yelp	NMI	42.04	0.30	32.58	42.30	40.60	30.09	45.46	47.82	47.56	44.36	12.86	**48.90**
	ARI	38.27	0.41	23.30	41.52	37.72	27.88	41.39	42.91	43.24	42.57	10.54	**44.89**
ACM	NMI	21.22	40.70	51.40	57.29	61.68	61.56	61.56	60.89	64.12	65.60	67.55	**68.21**
	ARI	21.00	37.13	53.01	60.43	65.15	61.46	64.39	59.85	66.29	69.30	71.53	**72.33**
IMDB	NMI	1.20	1.20	5.45	8.45	10.20	11.32	10.87	11.59	11.79	9.45	12.01	**12.31**
	ARI	1.70	1.65	4.40	7.46	8.20	10.40	10.01	9.92	10.32	8.02	12.32	**12.55**

4.3.4.3 Robustness to Model Depth

A salient property of HPN is the incorporation of the semantic propagation mechanism which is able to alleivate the semantic confusion and build a deeper and more powerful HGNN. Comparing to the previous HGNNs (e.g., HAN), the proposed HPN can stack more layers and learn more representative node representation. To show the superiority of semantic propagation in HPN, we test HAN and HPN with 1, 2, 3, 4, 5 layers, shown in Fig. 4.7.

As can be seen, with the growth of model depth, the performance of HAN performs worse and worse on both ACM and IMDB and we believe this phenomenon is the semantic confusion, leading to the degradation of previous heterogeneous GNNs (e.g., HAN). Obviously, semantic confusion makes HGNNs hard to become a really deep model, which severely limits their representation capabilities and hurts the performance of downstream tasks (e.g., node clustering). On the other hand, with the growth of model depth, the performance of the proposed HPN is getting better and better, indicating that semantic propagation mechanism is able to effectively alleviate the semantic confusion. So even stacking for more layers, the node representations learned via the proposed HPN are still distinguishable. In summary, the proposed HPN is able to capture high-order semantics and learns more representative node representation with deeper architecture, rather than learning indistinguishable node representation.

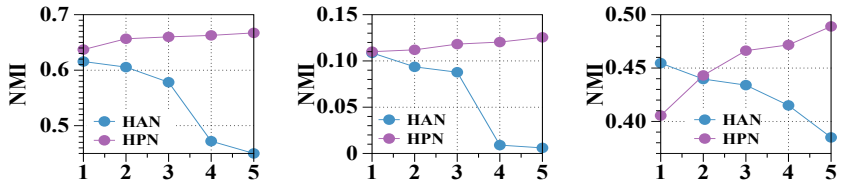


Fig. 4.7 Clustering results of HAN/HPN with 1,2,3,4,5 layers.

The detailed method description and validation experiments can been seen in (15).

4.4 Heterogeneous Graph Structure Learning

4.4.1 Overview

Most HGNNs follow a message-passing scheme where the node representation is learned by aggregating and transforming the representations of its original neighbors (39; 42; 11) or metapath-based neighbors (35; 38; 14; 8), which rely on one fundamental assumption, i.e. the raw heterogeneous graph structure is good. However, as heterogeneous graphs are usually extracted from complex interaction systems by some pre-defined rules, such assumption cannot be always satisfied. One reason is that, these interaction systems inevitably contain some uncertain information or mistakes. Taking a user-item graph built from recommendation as an example, it is well accepted that users may misclick some unwanted items, bringing noisy information to the graph. The other reason is that, the heterogeneous graphs are

often extracted with data cleaning, feature extraction and feature transformation by some pre-defined rules, which are usually independent to the downstream tasks and lead to the gap between the extracted graph and the optimal graph structure for the downstream tasks. Therefore, learning an optimal heterogeneous graph for GNN is a fundamental problem.

Recently, to alleviate the limitation of GNNs and adaptively learn graph structures for GNNs, graph structure learning (GSL) methods (7; 16; 3; 17) are proposed to jointly learns the GNN parameters and graph structure simultaneously. However, these methods cannot be directly applied to HGs with the following challenges: (1) The heterogeneity of graphs. When learning a homogeneous graph with only one type of relation, we usually only need to parameterize one adjacency matrix. However, a heterogeneous graph consists of multiple relations, each of which reflects one aspect of the heterogeneous graph. Since treating these heterogeneous relations uniformly will inevitably restrict the capability of graph structure learning. How to deal with this heterogeneity is a challenging problem. (2) The complex interactions in heterogeneous graphs. Different relations and node features have complex interactions, which drives the formation of different kinds of underlying graph structure (40). Moreover, the combination of different relations further forms a large number of high-order relationships with diverse semantics, which also implies distinct ways of graph generation. The heterogeneous graph structure will be affected by all these factors, therefore, these complex interactions must be thoroughly considered in heterogeneous graph structure learning.

In this section, we introduce an interesting work that makes the first attempt to investigate **H**eterogeneous **G**raph **S**tructure **L**earning for graph neural networks, called **HGSL**. In HGSL, the heterogeneous graph and the GNN parameters are jointly learned towards better node classification performance. Particularly, in the graph learning part, aiming to capture the heterogeneous metric of different relation generation, each relation subgraph is separately learned. Specifically, for each relation, three types of candidate graphs, i.e. the feature similarity graph, feature propagation graphs and semantic graphs, are generated by mining the complex correlations from heterogeneous node features and graph structures. The learned graphs are further fused to a heterogeneous graph and fed to a GNN. The graph learning parameters and the GNN parameters are jointly optimized towards classification objective.

4.4.2 The HGSL Method

We firstly introduce some basic concepts and formalize the problem of heterogeneous graph structure learning as follows:

Definition 1. **Node Relation Triple** A node relation triple $\langle v_i, r, v_j \rangle$ describes that two nodes v_i (head node) and v_j (tail node) are connected by relation $r \in \mathcal{R}$. We further define the type mapping functions $\phi_h, \phi_t : \mathcal{R} \rightarrow \mathcal{T}$ that map the relation to its head node type and tail node type respectively.

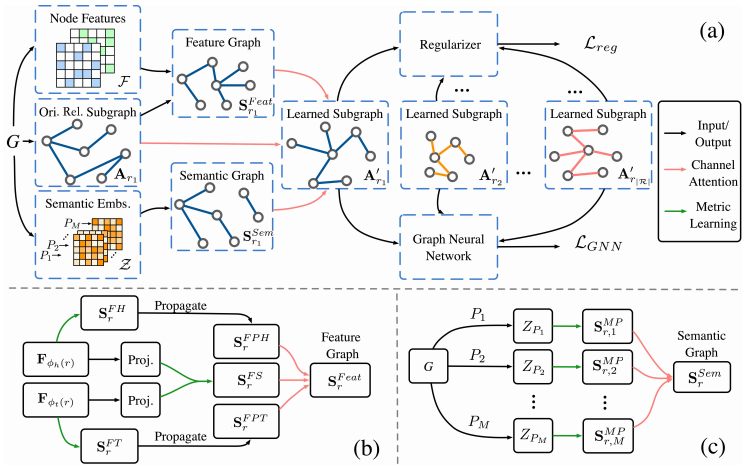


Fig. 4.8 Overview of the HGSL framework. (a) Model framework. (b) Feature graph generator. (c) Semantic graph generator.

Example 1. In a user-item heterogeneous graph, say $r =$ "UI" (a user buys an item), then we have $\phi_h(r) =$ "$User$" and $\phi_t(r) =$ "$Item$".

Definition 2. **Relation Subgraph** Given a heterogeneous graph $G = (V, E, \mathcal{F})$, a relation subgraph G_r is a subgraph of G that contains all node-relation triples with relation r. The adjacency matrix of G_r is $A_r \in \mathbb{R}^{|V_{\phi_h(r)}| \times |V_{\phi_t(r)}|}$, where $A_r[i,j] = 1$ if $\langle v_i, r, v_j \rangle$ exists in G_r, otherwise $A_r[i,j] = 0$. $\mathcal{A}$ denotes the relation subgraph set of all the relation subgraphs in G, i.e. $\mathcal{A} = \{A_r, r \in \mathcal{R}\}$.

Definition 3. **Heterogeneous Graph Strcture Learning (HGSL)** Given a heterogeneous graph G, the task of heterogeneous graph structure learning is to jointly learn a heterogeneous graph structure, i.e. a new relational subgraph set $\mathcal{A}'$, and the parameters of GNN for downstream tasks.

4.4.2.1 Model Framework

Fig. 4.8 (a) illustrates the framework of the proposed HGSL. As we can see, given a heterogeneous graph, HGSL firstly constructs the semantic representation matrices $\mathcal{Z}$ by metapath-based node representations from M metapaths. Afterwards, the heterogeneous graph structure and GNN parameters are trained jointly. For the graph learning part, HGSL takes the information from the original relation subgraph, the node features, and the semantic representations as input and generates relation subgraphs separately. Specifically, taking relation r_1 as an example, HGSL learns a feature graph $S_{r_1}^{Feat}$ and a semantic graph $S_{r_1}^{Sem}$ and fuse them with the original graph A_{r_1} to obtain the learned relation subgraph A'_{r_1}. Then, the learned subgraphs are fed into a GNN and a regularizer to perform node classification with regularization. By

minimizing the regularized classification loss, HGSL optimizes the graph structure and the GNN parameters jointly.

4.4.2.2 Feature Graph Generator

Since the original graph may not be optimal for downstream task, a natural idea would be to augment the original graph structure via fully utilizing the rich information inside heterogeneous node features. Usually, there are two factors that affect the formation of graph structure based on features. One is the similarity between node features, and the other is the relationship between node feature and relation in HG (36). As shown in Fig. 4.8 (b), we first propose to generate a feature similarity graph that captures the potential relationship generated by node features via heterogeneous feature projection and metric learning. Then we propose to generate the feature propagation graph, by propagating feature similarity matrices through topology structure. Finally, the generated feature similarity graph and feature propagation graph are aggregated to a final feature graph through a channel attention layer.

4.4.2.2.1 Feature Similarity Graph

The feature similarity graph S_r^{FS} determines the possibility of an edge with type $r \in \mathcal{R}$ between two nodes based on node features. Specifically, for each node v_i of type $\phi(v_i)$ with feature vector $f_i \in \mathbb{R}^{1\times d_{\phi(v_i)}}$, we adopt a type-specific mapping layer to project the feature f_i to a d_c-dimensional common feature $f_i' \in \mathbb{R}^{1\times d_c}$:

$$f_i' = \sigma\left(f_i \cdot W_{\phi(v_i)} + b_{\phi(v_i)}\right), \tag{4.1}$$

where $\sigma(\cdot)$ denotes a non-linear activation function, $W_{\phi(v_i)} \in \mathbb{R}^{d_{\phi(v_i)}\times d_c}$ and $b_{\phi(v_i)} \in \mathbb{R}^{1\times d_c}$ denote the mapping matrix and the bias vector of type $\phi(v_i)$, respectively. Then, for a relation r, we perform metric learning on the common features and obtain the learned feature similarity graph $S_r^{FS} \in \mathbb{R}^{|V_{\phi_h(r)}|\times|V_{\phi_t(r)}|}$, where the edge between nodes v_i and v_j is obtained by:

$$S_r^{FS}[i,j] = \begin{cases} \Gamma_r^{FS}(f_i', f_j') & \Gamma_r^{FS}(f_i', f_j') \geq \epsilon^{FS} \\ 0 & \text{otherwise,} \end{cases} \tag{4.2}$$

where $\epsilon^{FS} \in [0,1]$ is the threshold that controls the sparsity of feature similarity graph, and larger ϵ^{FS} implies a more sparse feature similarity graph. Γ_r^{FS} is a K-head weighted cosine similarity function defined as:

$$\Gamma_r^{FS}(f_i', f_j') = \frac{1}{K}\sum_{k}^{K} \cos\left(w_{k,r}^{FS} \odot f_i', w_{k,r}^{FS} \odot f_j'\right), \tag{4.3}$$

where $\odot$ denotes the Hadamard product, and $W_r^{FS} = [w_{k,r}^{FS}]$ is the learnable parameter matrix of Γ_r^{FS} that weights the importance of different dimensions of the feature

vectors. By performing metric learning as in Equation 4.3 and ruling out edges with little feature similarity by threshold ϵ^{FS}, HGSL learns the candidate feature similarity graph S_r^{FS}.

4.4.2.2.2 Feature Propagation Graph

The feature propagation graph is the underlying graph structure generated by the interaction between node features and topology structure. The key insight is that two nodes with similar features may have similar neighbors. Therefore, the process of generating feature propagation graph is two-fold: Firstly, generate the feature similarity graphs, i.e. find the similar nodes; secondly, propagate the feature similarity graph by topological structure to generate new edges, i.e. find the neighbors of the nodes with similar features.

Specifically, for each relation r, assume that we have two types of nodes $V_{\phi_h(r)}$ and $V_{\phi_t(r)}$ and the topology structure between them is $A_r \in \mathbb{R}^{|V_{\phi_h(r)}|\times|V_{\phi_t(r)}|}$. For the nodes $v_i, v_j \in V_{\phi_h(r)}$ with the same type $\phi_h(r)$, we can obtain the feature similarity:

$$S_r^{FH}[i,j] = \begin{cases} \Gamma_r^{FH}(f_i, f_j) & \Gamma_r^{FH}(f_i, f_j) \geq \epsilon^{FP} \\ 0 & \text{otherwise,} \end{cases} \tag{4.4}$$

where the threshold ϵ^{FP} controls the sparsity of feature similarity graph S_r^{FH}. Γ_r^{FH} is the metric learning function in the framework of Equation 4.3 with different parameters W_r^{FH}. Then we can model the head feature propagation graph $S_r^{FPH} \in \mathbb{R}^{|V_{\phi_h(r)}|\times|V_{\phi_t(r)}|}$ using S_r^{FH} and A_r as follows:

$$S_r^{FPH} = S_r^{FH} A_r. \tag{4.5}$$

As we can see, the feature similarity is propagated through the original graph topological structure and further generates the potential feature propagation graph structure. As for the nodes $V_{\phi_t(r)}$ with the same type $\phi_t(r)$, similar to Eq. 4.4, we can obtain the corresponding feature similarity graph S_r^{FT} with parameters W_r^{FT}. Therefore, the corresponding feature propagation graph S_r^{FPT} can be obtained as follows:

$$S_r^{FPT} = A_r S_r^{FT}. \tag{4.6}$$

Now, we have generated one feature similarity graph S_r^{FS} and two feature propagation graphs S_r^{FPH} and S_r^{FPT}. The overall feature graph for relation r, denoted as $S_r^{Feat} \in \mathbb{R}^{|V_{\phi_h(r)}|\times|V_{\phi_t(r)}|}$, can be obtained by fusing these graphs through a channel attention layer (38):

$$S_r^{Feat} = \Psi_r^{Feat}([S_r^{FS}, S_r^{FPH}, S_r^{FPT}]), \tag{4.7}$$

where $[S_r^{FS}, S_r^{FPH}, S_r^{FPT}] \in \mathbb{R}^{|V_{\phi_h(r)}|\times|V_{\phi_t(r)}|\times 3}$ is the stacked matrix of the feature candidate graphs, and Ψ_r^{Feat} denotes a channel attention layer with parameters $W_{\Psi,r}^{Feat} \in \mathbb{R}^{1\times1\times3}$ which performs 1×1 convolution on the input using $\text{softmax}(W_{\Psi,r}^{Feat})$. In this way, HGSL balances the importance of each candidate feature graph for each relation r by learning different weights respectively.

4.4.2.3 Semantic Graph Generator

The semantic graph is generated depending on the high-order topology structure in HG, describing the multi-hop structural interactions between two nodes. Notably, in heterogeneous graphs, these high-order relationships differ from each other with different semantics determined by metapaths. In light of this, we propose to learn semantic graph structures from different semantics.

Given a metapath P with the corresponding relations $r_1 \circ r_2 \circ \cdots \circ r_l$, a straightforward way to generate semantic graph would be fusing the adjacency matrices, i.e. $A_{r_1} \cdot A_{r_2} \cdot \cdots \cdot A_{r_l}$ (38). However, this method not only costs large memory with the computation of stacking multiple layers of adjacency matrices, but also discards the intermediate nodes which leads to information loss (8).

Alternatively, we propose a semantic graph generator shown in Fig. 4.8 (c). The semantic graph generator generates the potential semantic graph structure by metric learning on trained metapath-based node representations. Specifically, for an interested metapath set $\mathcal{P} = \{P_1, P_2, ..., P_M\}$ with M metapaths, HGSL uses trained MP2Vec (5) representations, denoted as $\mathcal{Z} = \{Z_{P_1}, Z_{P_2}, \cdots, Z_{P_M} \in \mathbb{R}^{|V| \times d}\}$, to generate semantic graphs. Since the training process of semantic representations is off-line, the computation cost and model complexity is largely reduced. Moreover, thanks to the mechanism of heterogeneous skip-gram, the information of intermediate nodes is well preserved.

After obtaining the semantic representations $\mathcal{Z}$, for each metapath P_m , we generate a candidate semantic subgraph adjacency matrix $S_{r,m}^{MP} \in \mathbb{R}^{|V_{\phi_h(r)}| \times |V_{\phi_t(r)}|}$, where each edge is calculated by:

$$S_{r,m}^{MP}[i,j] = \begin{cases} \Gamma_{r,m}^{MP}(z_i^m, z_j^m) & \Gamma_{r,m}^{MP}(z_i^m, z_j^m) \geq \epsilon^{MP} \\ 0 & \text{otherwise,} \end{cases} \tag{4.8}$$

where z_i^m stands for the i th row of Z_{P_m}, and $\Gamma_{r,m}^{MP}$ is the metric learning function with parameters $W_{r,m}^{MP}$. We can see that a relation r will generate M candidate semantic subgraphs, so the overall semantic subgraph for relation r, denoted as S_r^{Sem}, can be obtained by aggregating them:

$$S_r^{Sem} = \Psi_r^{MP}([S_{r,1}^{MP}, S_{r,2}^{MP}, \cdots, S_{r,M}^{MP}]), \tag{4.9}$$

where $[S_{r,1}^{MP}, S_{r,2}^{MP}, ..., S_{r,M}^{MP}]$ is the stacked matrix of M candidate semantic graphs. Ψ_r^{MP} denotes a channel attention layer whose weight matrix $W_{\Psi,r}^{MP} \in \mathbb{R}^{1\times1\times M}$ represents the importance of different metapath-based candidate graphs. After we obtain the aggregated semantic graph S_r^{Sem}, the overall generated graph structure A_r' for relation r can be obtained by aggregating the learned feature graph and semantic graph along with the original graph structure:

$$A_r' = \Psi_r([S_r^{Feat}, S_r^{Sem}, A_r]), \tag{4.10}$$

where $[S_r^{Feat}, S_r^{Sem}, A_r] \in \mathbb{R}^{|V_{\phi_h(r)}| \times |V_{\phi_t(r)}| \times 3}$ is the stacked matrix of the candidate graphs. Ψ_r is the channel attention layer whose weight matrix $W_{\Psi,r} \in \mathbb{R}^{1\times1\times3}$ denotes

the importance of candidate graphs in fusing the overall relation subgraph A'_r. With a new relation adjacency matrix A'_r for each relation r, a new heterogeneous graph structure is generated, i.e. $\mathcal{A}' = \{A'_r, r \in \mathcal{R}\}$.

4.4.2.4 Optimization

In this section, we show how HGSL jointly optimizes the graph structure $\mathcal{A}'$ and the GNN parameters for downstream task. Here we focus on GCN (20) and node classification. Please note that, with the learned graph structure $\mathcal{A}'$, our model can be applied to other homogeneous or heterogeneous GNN methods and other tasks. A two layer GCN with parameters $\theta = (W_1, W_2)$ on the learned graph structure $\mathcal{A}'$, can be described as:

$$f_\theta(X, A') = \text{softmax}\left(\hat{A}\sigma\left(\hat{A}XW_1\right)W_2\right), \tag{4.11}$$

where X is the original node feature matrix, i.e. $X[i,:] = f_i^T$ if the dimensions of all features are identical; otherwise, we use the common feature to construct X, i.e. $X[i,:] = f_i'^T$. The adjacency matrix A' is constructed from the learned heterogeneous graph $\mathcal{A}'$ by considering all nodes as one type. $\hat{A} = \tilde{D}^{-1/2}(A' + I)\tilde{D}^{-1/2}$, where $\tilde{D}_{ii} = 1 + \sum_j A'_{ij}$. Thus, the classification loss of GNN, i.e. $\mathcal{L}_{GNN}$, on the learned graph can be obtained by:

$$\mathcal{L}_{GNN} = \sum_{v_i \in V_L} \ell\left(f_\theta(X, A')_i, y_i\right), \tag{4.12}$$

where $f_\theta(X, A')_i$ is the predicted label of node $v_i \in V_L$ and $\ell(\cdot,\cdot)$ measures the difference between prediction and the true label y_i such as cross entropy.

Since graph structure learning methods enable the original GNN with stronger ability to fit the downstream task, it would be easier for them to over-fit. Thus, we apply regularization term $\mathcal{L}_{reg}$ to the learned graph as follows:

$$\mathcal{L}_{reg} = \alpha\|A'\|_1. \tag{4.13}$$

This term encourages the learned graph to be sparse. The overall loss $\mathcal{L}$ can be obtained by:

$$\mathcal{L} = \mathcal{L}_{GNN} + \mathcal{L}_{reg}. \tag{4.14}$$

By minimizing $\mathcal{L}$, HGSL optimizes heterogeneous graph structure and the GNN parameters θ jointly towards better downstream task performance.

4.4.3 Experiments

4.4.3.1 Experimental Settings

Datasets Three datasets including Yelp[15], ACM[16], and DBLP[17] are used to evaluate the proposed model.

Baselines HGSL are compared with eleven state-of-the-art representation methods including four homogeneous graph representation methods, i.e., DeepWalk (27), GCN (20), GAT (33), and GraphSAGE (10), four heterogeneous graph representation methods, i.e., MP2Vec (5), HAN (35), HeGAN (12), and GTN (38), and three graph structure learning related methods, i.e. LDS (7), Pro-GNN (17), and Geom-GCN (25).

Experimental Settings For all GNN-related models, the number of layers are set as 2 for a fair comparison. The feature dimension in common space d_c and the representation dimension d for all methods are set as 16 and 64 respectively. We choose the popular metapaths adopted in previous methods (35; 5; 24) for metapath based models and report the best result. For our proposed model, we use 2-head cosine similarity function defined in Equation 4.3, i.e. K=2. We set learning rate and weight decay as 0.01 and 0.0005 respectively. Other hyper-parameters, namely ε^{FS}, ε^{FP}, ε^{MP}, and α, are tuned by grid search. The code and datasets are publicly available on Github[18].

4.4.3.2 Node Classification

In this section, the performance of HGSL on node classification task is shown. Macro-F1 and Micro-F1 are selected as the metrics for evaluation. The mean and standard deviation of percentage of the metric values are shown in Table 4.3, from which we have following observations: (1) With the capability to adaptively learn the heterogeneous graph structure, HGSL consistently outperforms all the baselines. It demonstrates the effectiveness of our proposed model. (2) Graph structure learning methods generally outperform the original GCN since it enables GCN to aggregate feature from the learned structure. (3) HGNN methods, i.e. HAN, GTN, and HGSL achieve better performance compared to GNNs since the heterogeneity is addressed. (4) GNN-based methods mostly outperform random walk-based graph representation methods since the node features are utilized. This phenomenon becomes more obvious when it comes to Yelp dataset, since the node features, i.e. keywords, are helpful in classifying business categories.

[15] https://www.yelp.com

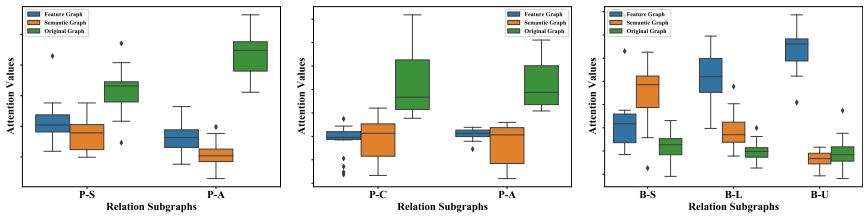
[16] http://dl.acm.org/

[17] https://dblp.uni-trier.de

[18] https://github.com/Andy-Border/HGSL

Table 4.3 Performance evaluation of node classification (mean in percentage ± standard deviation).

	DBLP		ACM		Yelp	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
DeepWalk	88.00 ± 0.47	89.13 ± 0.41	80.65 ± 0.60	80.32 ± 0.61	68.68 ± 0.83	73.16 ± 0.96
GCN	83.38 ± 0.67	84.40 ± 0.64	91.32 ± 0.61	91.22 ± 0.64	82.95 ± 0.43	85.22 ± 0.55
GAT	77.59 ± 0.72	78.63 ± 0.72	92.96 ± 0.28	92.86 ± 0.29	84.35 ± 0.74	86.22 ± 0.56
GraphSage	78.37 ± 1.17	79.39 ± 1.17	91.19 ± 0.36	91.12 ± 0.36	93.06 ± 0.35	92.08 ± 0.31
MP2Vec	88.86 ± 0.19	89.98 ± 0.17	78.63 ± 1.11	78.27 ± 1.14	59.47 ± 0.57	65.11 ± 0.53
HAN	90.53 ± 0.24	91.47 ± 0.22	91.67 ± 0.39	91.57 ± 0.38	88.49 ± 1.73	88.78 ± 1.40
HeGAN	87.02 ± 0.37	88.34 ± 0.38	82.04 ± 0.77	81.80 ± 0.79	62.41 ± 0.76	68.17 ± 0.79
GTN	90.42 ± 1.29	91.41 ± 1.09	91.91 ± 0.58	91.78 ± 0.59	92.84 ± 0.28	92.19 ± 0.29
LDS	75.65 ± 0.20	76.63 ± 0.18	92.14 ± 0.16	92.07 ± 0.15	85.05 ± 0.16	86.05 ± 0.50
Pro-GNN	89.20 ± 0.15	90.28 ± 0.16	91.62 ± 1.28	91.55 ± 1.31	74.12 ± 2.03	77.45 ± 2.12
Geom-GCN	79.43 ± 1.01	80.94 ± 1.06	70.20 ± 1.23	70.00 ± 1.06	84.28 ± 0.70	85.36 ± 0.60
HGSL	**91.92 ± 0.11**	**92.77 ± 0.11**	**93.48 ± 0.59**	**93.37 ± 0.59**	**93.55 ± 0.52**	**92.76 ± 0.60**



(a) ACM (b) DBLP (c) Yelp

Fig. 4.9 Channel attention distributions of relation subgraphs.

4.4.3.3 Importance Analysis of Candidate Graphs

In order to investigate whether HGSL can distinguish the importance of candidate graphs, we analyze the weight distribution of the channel attention layer for fusing each relation subgraphs, i.e. the weights of Ψ_r in Equation 4.10, on three datasets. We train HGSL 20 times and set all the thresholds of HGSL as 0.2. The attention distributions are shown in Fig. 4.9. As we can observe, for relation subgraphs in ACM and DBLP, the original graph structure is the most important structure for GNN-based classification. However, as for Yelp, the channel attention values of different relation subgraphs differ from each other. Specifically, for B-U (business-user) and B-L (business-rating level) relation subgraphs, the feature graphs are assigned with large channel attention value in graph structure learning. This phenomenon implies that the information in node features plays a more important role than that of semantic representations which agrees with the the previously discussed experiments and further demonstrates the capability of HGSL in adaptively learning a larger channel attention value for more important information.

The detailed method description and validation experiments can been seen in (41).

4.5 Conclusions

In attribute-assisted HG, diverse types of nodes are assisted with different attributes, reflecting the characteristics of nodes. In this chapter, we introduce three attribute-assisted HG representation models including HAN, HPN, and HGSL, which simultaneously integrate both structural information and attribute information into the node representations. Taking node attributes and graph structures as inputs, HAN learns the importance of neighbor and meta-path and then aggregates them to learn node representation in a hierarchical manner. Further, HPN improves the node-level aggregating process in HAN via emphasizing the local semantic of each node, significantly alleviating the deep degradation phenomenon (a.k.a, semantic confusion). After that, HGSL mines the complex correlations from heterogeneous node features and graph structures, and then jointly learns the GNN parameters and graph structure simultaneously.

References

[1] Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *ICLR* (2015).

[2] Chen, T., and Sun, Y. Task-guided and path-augmented heterogeneous network embedding for author identification. In *WSDM* (2017), pp. 295–304.

[3] Chen, Y., Wu, L., and Zaki, M. J. Deep iterative and adaptive learning for graph neural networks. *arXiv preprint arXiv:1912.07832* (2019).

[4] Dong, Y., Chawla, N. V., and Swami, A. metapath2vec: Scalable representation learning for heterogeneous networks. In *SIGKDD* (2017), pp. 135–144.

[5] Dong, Y., Chawla, N. V., and Swami, A. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD* (2017), pp. 135–144.

[6] Fan, S., Zhu, J., Han, X., Shi, C., Hu, L., Ma, B., and Li, Y. Metapath-guided heterogeneous graph neural network for intent recommendation. In *KDD* (2019), ACM, pp. 2478–2486.

[7] Franceschi, L., Niepert, M., Pontil, M., and He, X. Learning discrete structures for graph neural networks. In *ICML* (2019), pp. 1972–1982.

[8] Fu, X., Zhang, J., Meng, Z., and King, I. MAGNN: metapath aggregated graph neural network for heterogeneous graph embedding. In *WWW* (2020), pp. 2331–2341.

[9] Hamilton, W., Bajaj, P., Zitnik, M., Jurafsky, D., and Leskovec, J. Embedding logical queries on knowledge graphs. In *Advances in Neural Information Processing Systems* (2018), pp. 2030–2041.

[10] Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *NIPS* (2017), pp. 1024–1034.

[11] Hong, H., Guo, H., Lin, Y., Yang, X., Li, Z., and Ye, J. An attention-based graph neural network for heterogeneous structural learning. In *AAAI* (2020), pp. 4132–4139.

[12] Hu, B., Fang, Y., and Shi, C. Adversarial learning on heterogeneous information networks. In *KDD* (2019), pp. 120–129.

[13] Hu, L., Yang, T., Shi, C., Ji, H., and Li, X. Heterogeneous graph attention networks for semi-supervised short text classification. In *EMNLP-IJCNLP* (2019), pp. 4820–4829.

[14] Hu, Z., Dong, Y., Wang, K., and Sun, Y. Heterogeneous graph transformer. In *WWW* (2020), pp. 2704–2710.

[15] Ji, H., Wang, X., Shi, C., Wang, B., and Yu, P. S. Heterogeneous graph propagation network. In *IEEE Trans. Knowl. Data Eng.* (2021).

[16] Jiang, B., Zhang, Z., Lin, D., Tang, J., and Luo, B. Semi-supervised learning with graph learning-convolutional networks. In *CVPR* (2019), pp. 11313–11320.

[17] Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., and Tang, J. Graph structure learning for robust graph neural networks. In *KDD* (2020), pp. 66–74.

[18] Keyulu, X., Chengtao, L., Yonglong, T., Tomohiro, S., Ken-ichi, K., and Stefanie, J. Representation learning on graphs with jumping knowledge networks. In *ICML* (2018), pp. 5453–5462.

[19] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *ICLR* (2015).

[20] Kipf, T. N., and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR* (2017).

[21] Klicpera, J., Bojchevski, A., and Gunnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR* (2019).

[22] Lee, S., Park, S., Kahng, M., and Lee, S.-G. Pathrank: Ranking nodes on a heterogeneous graph for flexible hybrid recommender systems. *Expert Systems with Applications 40*, 2 (2013), 684–697.

[23] Li, X., Wu, Y., Ester, M., Kao, B., Wang, X., and Zheng, Y. Semi-supervised clustering in attributed heterogeneous information networks. In *WWW* (2017), pp. 1621–1629.

[24] Lu, Y., Shi, C., Hu, L., and Liu, Z. Relation structure-aware heterogeneous information network embedding. In *AAAI* (2019), pp. 4456–4463.

[25] Pei, H., Wei, B., Chang, K. C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks. In *ICLR* (2020).

[26] Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *SIGKDD* (2014), pp. 701–710.

[27] Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *KDD* (2014), pp. 701–710.

[28] Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., and Welling, M. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference* (2018), Springer, pp. 593–607.

[29] Shang, J., Qu, M., Liu, J., Kaplan, L. M., Han, J., and Peng, J. Meta-path guided embedding for similarity search in large-scale heterogeneous information networks. *CoRR abs/1610.09769* (2016).

[30] Shi, C., Hu, B., Zhao, X., and Yu, P. Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2018).

[31] Sun, Y., Han, J., Yan, X., Yu, P. S., and Wu, T. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *VLDB 4*, 11 (2011), 992–1003.

[32] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *NIPS* (2017), pp. 5998–6008.

[33] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph Attention Networks. *ICLR* (2018).

[34] Wang, X., He, X., Wang, M., Feng, F., and Chua, T.-S. Neural graph collaborative filtering. In *SIGIR* (2019), pp. 165–174.

[35] Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., and Yu, P. S. Heterogeneous graph attention network. In *WWW* (2019), pp. 2022–2032.

[36] Wang, X., Zhu, M., Bo, D., Cui, P., Shi, C., and Pei, J. AM-GCN: adaptive multi-channel graph convolutional networks. In *KDD* (2020), R. Gupta, Y. Liu, J. Tang, and B. A. Prakash, Eds.

[37] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In *ICML* (2015), pp. 2048–2057.

[38] Yun, S., Jeong, M., Kim, R., Kang, J., and Kim, H. J. Graph transformer networks. In *NIPS* (2019), pp. 11960–11970.

[39] Zhang, C., Song, D., Huang, C., Swami, A., and Chawla, N. V. Heterogeneous graph neural network. In *KDD* (2019), pp. 793–803.

[40] Zhang, C., Swami, A., and Chawla, N. V. SHNE: representation learning for semantic-associated heterogeneous networks. In *WSDM* (2019), pp. 690–698.

[41] Zhao, J., Wang, X., Shi, C., Hu, B., Song, G., and Ye, Y. Heterogeneous graph structure learning for graph neural networks. In *35th AAAI Conference on Artificial Intelligence (AAAI)* (2021).

[42] Zhao, J., Wang, X., Shi, C., Liu, Z., and Ye, Y. Network schema preserving heterogeneous information network embedding. In *IJCAI* (2020), pp. 1366–1372.