Graph Representation Learning on Heterogeneous Academic Network

Zhenyang Ni
Department of Electronic Engineering
Shanghai Jiao Tong University
0107nzy@sjtu.edu.cn

Qiuwen Wang
Department of Electronic Engineering
Shanghai Jiao Tong University
wangqiuwen@sjtu.edu.cn

Haochen Zhao
Department of Electronic Engineering
Shanghai Jiao Tong University
zhaohaochen@sjtu.edu.cn

Abstract

Semi-supervised graph representation learning on academic network has recently been intensively studied. However, most existing methods focus on homogeneous network with original node feature such as Cora and Citeseer. To solve the graph representation problem on heterogeneous academic network without origin node feature, such as AceMap, we propose several graph relationship transforming methods and a novel feature transformation method for node classification. First, we transform the raw heterogeneous graph into two homogeneous graphs and one heterogeneous graph with the relationship transforming methods. Then we generate graph embedding and adjacency matrix for these graphs. Finally we use these graph embedding and adjacency matrix for multi-label node classification and link prediction. For the node classification task, we propose a novel heterogeneous graph convolutional network (HetGCN). For the link prediction task, we use the negative sampling method. We use our method for the kaggle graph data-mining competition. Our method ranks 4th in the node prediction task, and 1st in the link prediction task. All source codes are available at https://github.com/data-mining-group14/Heterogeneous-graph-data-mining.

1 Introduction

Most existing work about academic graph representation learning focus on homogeneous network. Node2vec[5] returns feature representations that maximize the likelihood of preserving network neighborhoods of nodes. GCN[9] uses a graph convolution neural network which encodes both local graph structure and features if nodes.

However, academic networks are usually heterogeneous network with different node types (e.g., authors,papers) and edge types(e.g.,coauthor,citation). In this project, we aim to learn the graph embedding for a heterogeneous academic network AceMap [11]. AceMap includes two types of node: author node and paper node, and two types of edges: the belonging of papers and citation between papers. Then we use the graph embedding for node classification and link prediction task. For the node classification task, we are given labels of 20% of the paper node, we are expected to predict the multiple labels of each author node on what conferences he has published paper in. For the link prediction task, the links to predict in this problem are future coauthor-ship or referencing between authors. The provided data is until 2019, with coauthor-ship and referencing implied in relationship between papers.

To handle the challenging tasks, we design several data pre-processing methods to transfer the raw heterogeneous network to two homogeneous graphs and one heterogeneous graph. Then we obtain the graph embedding used for node classification and link prediction based on node2vec and GCN.k

As a result, the mean F-score of our method for node classification achieve 0.50764, ranking 4th in the Kaggle competition; the AUC of our method for link prediction achieve 0.80793, ranking 1st in the Kaggle competition. The main contributions of this paper are summarized as follow:

• We transform the citation and author-paper relationships in the original graph into coauthor and citation relationships between papers nodes and authors nodes. Thus we can apply conditional node2vec and GCN on the new dataset.

• We propose a heterogeneous GCN (HetGCN) architecture based on GCN, in which a transformation between the author feature matrix and paper feature matrix is added.

• We conduct extensive experiments to find out the optimal walk length for node2vec. Results of the competition prove that node2vec with proper parameters is sufficient for the link prediction task.

2 Related Works

Node2vec. Node2vec[5] is a method based on random walk to generate embedding for a graph. It can collect insight from the graph and output vectors friendly to other machine learning applications. The tunable parameters are dimension d, walk length w, and walking direction preference p, q. w control the walk length and implies the range of graph info to be collected. q control the preference of walking to a close or remote node. It enables random walk to act like BFS for $q > 1$ or DFS for $q < 1$. p control the probability of returning to an already visited node. After walk is done, d control the following dimension of word2vec where each random walk is viewed as a word.

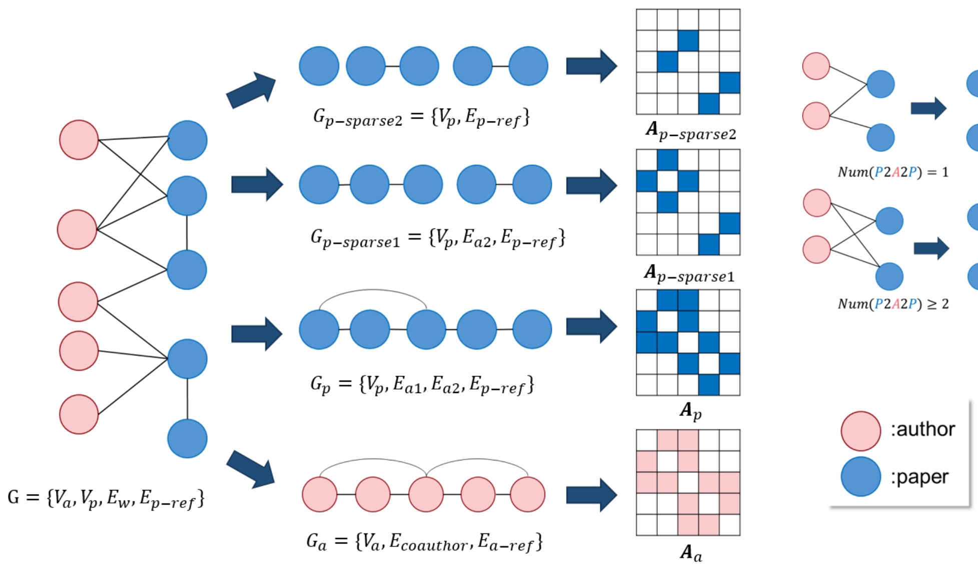
Graph Convolution network. Convolution in GCN is divided into spectral approaches and spatial approaches according to the domain in which it operates.The characteristics of a node and the characteristics of the surrounding nodes are merged as the new feature of the node. The spectral approaches were originally proposed by Bruna et al.[2], which convert the data of the graph structure into the Laplacian space domain, using the eigenvalues and eigenvectors of the Laplacian matrix to transform the Fourier transformation and convolution to graph structure data. Kipf et al.[9]proposed Graph Convolutional Network to scale linearly in the number of graph edges and learn hidden layer representations that encode both local graph structure and features of nodes. Defferrand et al.[3] use Chebyshev polynomials as the convolution kernel on the spectral domain to perform operations. The convolution operation in the spatial domain is more similar to the traditional convolution, which performs a weighted average on the node and its adjacent nodes to generate new features of the node.Hamilton et al.[6] changed the idea of trying to learn the embedding of all nodes on a graph, and instead learned a mapping that generates embedding for each node. It learns how to aggregate features from the local neighbors of a vertex by training a set of aggregator functions. Velikovi et al.[12] added an attention mechanism while performing convolution in the spatial domain to make the attention function adaptively learn the weight value of adjacent nodes to the current node. Fey et al.[4] filter the geometric structure based on B-splines. Shi et al.[10] introduces a masked label prediction strategy to incorporate feature and label propagation at both training and inference time. Kim et al.[7] exploit two attention forms compatible with a self-supervised task to predict edges, whose presence and absence contain the inherent information about the importance of the relationships between nodes. Bianchi et al.[1] were inspired by auto-regressive moving average (ARMA) filter and proposed ARMA convolution layer.

3 Data Pre-processing

3.1 Graph Formulation

The raw graph from the heterogeneous graph is defined as $G = \{V_a, V_p, E_{a2p}, E_{p-ref}\}$. Node set V_a denotes the author nodes and V_p denotes the paper nodes. E_{a2p} is the edge set from author nodes to the paper nodes written by them. E_{p-ref} is the edge set of citations between papers. We use several graph transforming methods to generate three types of new graph.

Author Graph The author graph $G_a = \{V_a, E_{coauthor}, E_{a-ref}\}$ records the co-authorship and referencing between authors. Edge set $E_{coauthor}$ denotes the co-authorship within the authors of one paper. If a paper has more than one authors, edges will be added between these authors. Edge set E_{a-ref} denotes the citation relationship between authors, which is transformed from E_{p-ref}. If there is a citation relationship between two papers, there will be edges between the two papers' authors.



$G_{p-sparse2} = \{V_p, E_{p-ref}\}$
$A_{p-sparse2}$
$G_{p-sparse1} = \{V_p, E_{a2}, E_{p-ref}\}$
$A_{p-sparse1}$
$G_p = \{V_p, E_{a1}, E_{a2}, E_{p-ref}\}$
A_p
$G = \{V_a, V_p, E_w, E_{p-ref}\}$
$G_a = \{V_a, E_{coauthor}, E_{a-ref}\}$
A_a
$Num(P2A2P) = 1$
$Num(P2A2P) \geq 2$
:author
:paper

Figure 1: The four adjacency matrices of homogeneous graphs. Raw graph G is transformed into three paper graph G_p,$G_{p-sparse1}$ and $G_{p-sparse2}$ and one author graph G_a. The rule of generating $G_{p-sparse2}$ is illustrated in the right of the figure.

Paper Graph The dense paper graph is defined as $G_p = \{V_p, E_{a2}, E_{a1}, E_{p-ref}\}$. E_{a2} is the edge set in which two paper are connected only if they have more than two common author. E_{a1} is the edge set in which two paper are connected only if they have one common author. Since G_P turns out to be too dense, we also define two sparse paper graph $G_{p-sparse1} = \{V_p, E_{a2}, E_{p-ref}\}$ and $G_{p-sparse2} = \{V_p, E_{p-ref}\}$.

Heterogeneous Graph The heterogeneous graph includes author nodes and paper nodes at the same time. The graph is defined as $G_{Het} = \{V_a, V_p, E_{coauthor}, E_{a2p}, E_{p-ref}\}$. G_{Het} can be split into three graphs:$G_{Het-a} = \{V_a, E_{coauthor}\}$, $G_{Het-a2p} = \{V_a, V_p, E_{a2p}\}$ and $G_{Het-p} = \{V_p, E_{p-ref}\}$.

3.2 Adjacency Matrix

The adjacency matrix A is a $\mid V \mid \times \mid V \mid$ 0-1 matrix. $A_{i,j} = 1$ i.f.f. an edge exists between node i and node j. We generate four adjacency matrices $\mathbf{A}_a$,$\mathbf{A}_p$,$\mathbf{A}_{p-sparse1}$, $\mathbf{A}_{p-sparse2}$ for homogeneous graph G_a,G_p,$G_{p-sparse1}$ and $G_{p-sparse2}$. For the heterogeneous graph G_{Het-a}, $G_{Het-a2p}$, G_{Het-p}, we generate three adjacency matrices $\mathbf{A}_{Het-a}$,$\mathbf{A}_{Het-a2p}$ and $\mathbf{A}_{Het-p}$. See the generation of the adjacency matrices in Figure 1 and Figure 2.

3.3 Graph Feature Extraction

We use node2vec[5] to extract the features in the graph. The features are expected to express the connection of nodes and underlying similarity between authors or papers. We generate four features matrices $\mathbf{F}_a, \mathbf{F}_{p1}, \mathbf{F}_{p2}$ and $\mathbf{F}_{p3}$ with node2vec. $\mathbf{F}_a, \mathbf{F}_{p1}, \mathbf{F}_{p2}$ are generated from G_a,G_p and $G_{p-sparse1}$. The generation of $\mathbf{F}_{p3}$ is illustrated in Figure 3. The raw graph G is viewed as a homogeneous graph. We apply node2vec on G and take the paper node part of the whole feature matrix as $\mathbf{F}_{p3}$.

In addition, we tried to use only the citation relationships between papers to extract feature, and also tried to encode the authors ID to a high dimension vector and then use PCA dimensionality reduction to obtain the features of the papers, but these two methods will lose the relevant information of the author and lead to poor final results, so we wont go into details.

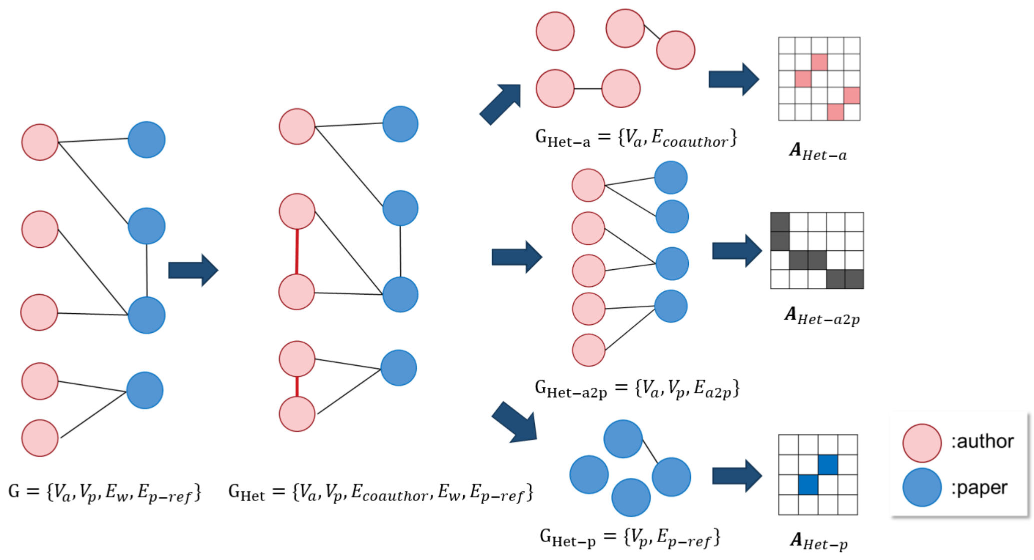

$G = \{V_a, V_p, E_w, E_{p-ref}\}$
$G_{Het} = \{V_a, V_p, E_{coauthor}, E_w, E_{p-ref}\}$
$G_{Het-a} = \{V_a, E_{coauthor}\}$
A_{Het-a}
$G_{Het-a2p} = \{V_a, V_p, E_{a2p}\}$
$A_{Het-a2p}$
$G_{Het-p} = \{V_p, E_{p-ref}\}$
A_{Het-p}
:author
:paper

Figure 2: The three adjacency matrices for sub-graphs of the Heterogeneous Graph. Raw graph G is transformed into G_{Het} and split into three graphs G_{Het-a}, $G_{Het-a2p}$ and G_{Het-p}. The shape of $\mathbf{A}_{Het-a}$ is $|V_a| \times |V_a|$, the shape of $\mathbf{A}_{Het-p}$ is $|V_p| \times |V_p|$, and the shape of $\mathbf{A}_{Het-a2p}$ is $|V_p| \times |V_a|$.

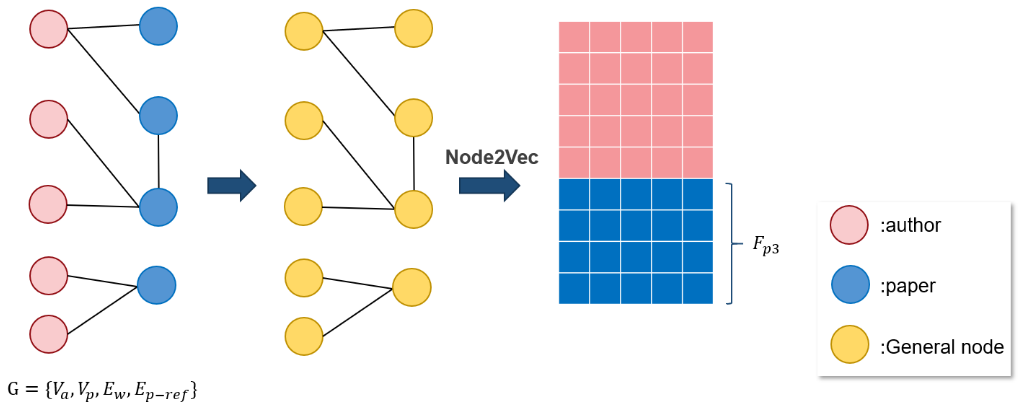


Figure 3: Generation of $\mathbf{F}_{p3}$. The raw graph G is viewed as a homogeneous graph. After applying node2vec we have the whole feature matrix which contains the paper node part (blue) and the author node part (pink).

4 Node Classification

4.1 Homogeneous paper graph convolutional network

In the problem of node classification for authors, since we found that many authors have the case where some of their published papers are labeled, and the other part is not labeled, so our first idea is to use the papers as nodes to create an homogeneous graph , And use the conferences of the papers as the label of the node. This kind of homogeneous method predicts the conference labels of the papers without labels, and then derives the conferences of the authors through the known relationships between the papers and the authors. In this method, we use a variety of methods to obtain the features and adjacency matrix of the paper as shown in section 3. The first method extracts the features of the paper nodes from $\mathbf{M}_{AP}$, and use $\mathbf{A}_{p-sparse2}$ as the adjacency matrix. The second method is to use $\mathbf{M}_{PD}$ as feature of paper nodes. In this case, we use both $\mathbf{A}_P$ and $\mathbf{A}_{p-sparse1}$ as the adjacent matrix respectively.

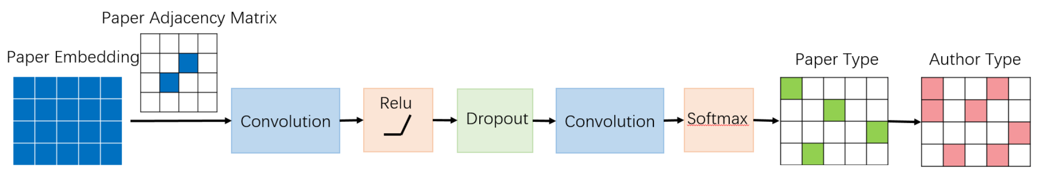


Figure 4: The architecture of Homogeneous Paper GCN. Paper's feature and adjacency matrix are entered into the network and the network outputs the classification of the paper nodes, then obtains the classification of authors.

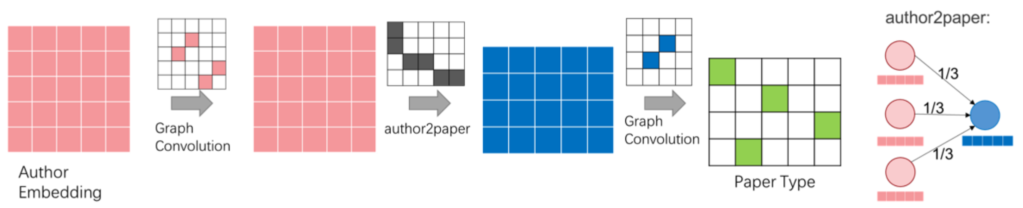


Figure 5: The architecture of HetGCN. HetGCN contains two phases: author convolution and paper convolution.An author2paper transformation is used to concat the two phases, which is illustrated in the right of the figure.

4.2 Heterogeneous graph convolutional network

To utilize all the information in the raw heterogeneous net work, we propose a heterogeneous graph convolutional network (HetGCN). HetGCN uses the architecture of GCN, including graph convulution and MLP. The input of HetGCN is the author feature matrix $\mathbf{F}_a$, and the graph embedding is generated by two author graph convolution based on $\mathbf{A}_{Het-a}$ and two paper graph convolution based on $\mathbf{A}_{Het-p}$. A transformation between the author feature matrix and the paper feature matrix is needed, in which the feature vector of a paper node is the average of the feature vectors of authors who write the paper. We can just multiple the normalized author-paper adjacency matrix $\mathbf{A}_{Het-a2p}$ with the author feature matrix to implement the transformation. Classification results of the classification is produced by MLP. See the architecture in Figure 5.

5 Link Prediction

The links to predict in this problem are future coauthor-ship or referencing between authors. The provided data is until 2019, with coauthor-ship and referencing separately provided. A rough idea is that if two authors are similar, the likelihood of such a link is high. However, the challenge is to make proper assessment on how "similar" any pair of authors are. Also noteworthy, unlike traditional date sets like Cora, there is no detailed information in any paper. In essence, we predict the graph in 2020, with only the same graph up to 2019.

With the limitation of available data in mind, our proposed method only relies on features extracted from node2vec as input, and translate the link prediction problem to a binary classification problem. Every author is described as a N-dim vector, containing all their information. For any pair of author, their feature vectors are piece-wise multiplied (Hadamard Product in Eq 1) to yield one single vector. This vector is then fed into a Logistic Regression classifier to find the probability. It is important to make sure that during training, there are equal number of positive and negative samples. Negative sampling is employed to achieve that. For every positive sample (pair of authors with edge), another negative sample (pair of authors with no edge) is randomly selected. This whole process for link prediction is shown at Fig 6.

$$\text{Hadamard}(f(u), f(v))_i = f(u)_i \times f(v)_i \tag{1}$$

6 Experiments and Analysis

6.1 Experimental setup

Datasets. We evaluate our method on the heterogeneous academic network AceMap [11]. AceMap includes two types of node: author node and paper node, and two types of edges: the belonging of

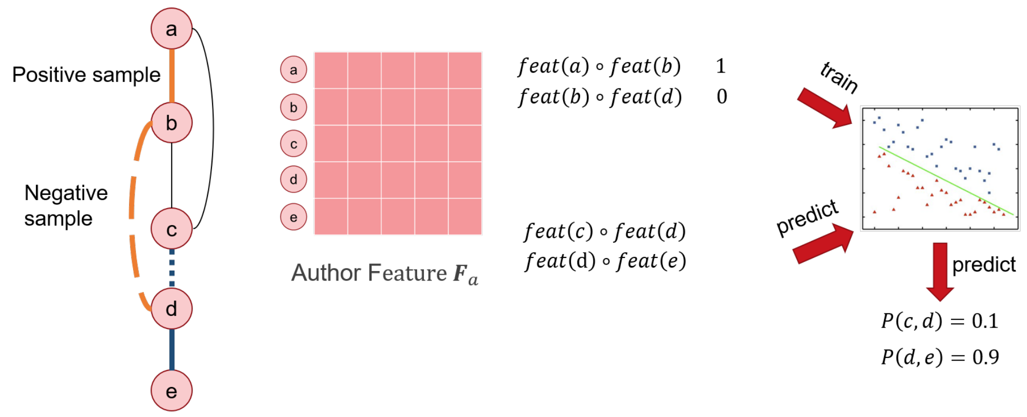


Figure 6: Process of link prediction. Training link labels are generated with negative sampling from author graph. Inputs to Logistic Regression classifier are the Hadamard Product of pair of author feature vectors. Training inputs are sampled author pair with label 0 or 1 and their two feature vectors. Prediction inputs are feature vectors of author pair, output is the probability of edge existence.

Table 1: Test result of node classification using homogeneous paper graph

Graph formulation	$\mathbf{F}_{p3} + \mathbf{A}_{p2}$	$\mathbf{F}_{p3} + \mathbf{A}_{p2}$	$\mathbf{F}_{p3} + \mathbf{A}_{p2}$	$\mathbf{F}_{p3} + \mathbf{A}_{p2}$	$\mathbf{F}_{p1} + \mathbf{A}_{p}$	$\mathbf{F}_{p1} + \mathbf{A}_{p1}$
Convolution kernel	GAT	GAT	ARMA	ARMA	GAT	GAT
hidden layers	8	16	16	32	8	8
Mean F-Score	0.47456	0.47413	0.47423	0.47590	0.48185	0.48802

papers and citation between papers. The whole dataset and its description are available at `https://www.kaggle.com/c/EE226-2021spring-problem1/data`. In homogeneous Paper GCN, we use 4853 papers to train and 300 papers to validate the model.

Metrics. For the node classification task, mean F-score is used for the metric. For the link prediction task, area under receiver operating characteristic curve is used for the metric.

Implementation details. In the homogeneous paper GCN, we use two-layer convolution network. The specific network structure is shown as Figure 4. In the first method, we use $\mathbf{F}_{p3}$ generated by node2vec with dimension $d = 1000$,walk length $w = 10$, and walking direction preference $p, q = 1$ as the feature matrix and use $\mathbf{A}_{p-sparse2}$ as adjacency matrix. In this method, we use GAT[12], ARMA[1], as convolution kernels. For GAT, we use eight-head and set the hidden layers to 8 and 16. For ARMA, we use 2 stacks and set the hidden layers to 16 and 32.In the second method, we use $\mathbf{F}_{p1}$ generated by node2vec with dimension $d = 100$,walk length $w = 3$, and walking direction preference $p, q = 1$ as the feature matrix and use $\mathbf{A}_p$ and $\mathbf{A}_{p-sparse1}$as adjacency matrix. We use only GAT in this method, and also set it to be 8-head. In both of these methods, we use log-softmax and the loss function is the negative log likelihood loss (NLLloss). In addition, we set L2 regularization coefficient to 0.0005, learning rate to 0.01. We train 300 epochs and the final model is the one with highest validation accuracy. In the experiment, we use other convolution kernels and parameters, but they finally perform poor test results and will not be discussed here.

In the HetGCN, we use the feature matrix $\mathbf{F}_a$ generated by node2vec with dimension $d = 128$, walk length $w = 3$, and walking direction preference $p, q = 1$. The dimension of hidden layer is 64, and the dropout rate is 10%. Four types of graph convolution are used, including GCN[9], Cheb(k=2)[3], ARMA(stack=2)[1] and GAT(head=18)[12]. The model is trained by Adam[8] for 200 epochs, with the learning rate=0.01, weight decay=5e-4. The learning rate decay every 30 epochs with gamma=0.9.

In link prediction, we use author graph G_a to generate positive and negative label samples. The author feature matrix $\mathbf{F}_a$ is feature input and its generation parameters from node2vec are subject to further tuning and will be discussed later.

6.2 Node classification

We predict the multiple labels of author nodes with homogeneous Paper GCN and HetGCN, the results are shown in Table 1 and Table 2 respectively.

Table 2: Test result of node classification using HetGCN

Convolution kernel	GCN	ARMA	GAT(head=8)	GAT(head=1)	Cheb
Mean F-Score	0.48136	0.48233	0.48561	0.47955	0.47681

Table 3: Comparison of result AUC with different parameters for node2vec

	w=1	w=2	w=3	w=4	w=5	w=10	w=15	w=3~5	w=2~6	w=2~9
AUC	0.51	0.756	0.772	0.770	0.763	0.752	0.736	0.792	0.804	0.805

We used model integration finally. We use the models shown in Table 1 and Table 2 to vote on the classification results of the papers. Models with public mean F-Scores between 0.47 and 0.48 are given a weight of 1, and models with scores above 0.48 are given a weight of 2. Then we got the final result and the Mean F-Score of it is 0.51438 in public scale and 0.50765 in private scale.

6.3 Link prediction

Our experiment focus on the influence of different parameters in node2vec embedding. The tunable parameters are dimension d, walk length w, and walking direction pcitation p, q. We tune these parameters sequentially.

During preliminary testing with arbitraryly set default parameters $d = 100, w = 10, p = q = 1$, we found that there is a strong tendency of overfitting. Our method reached an local test AUC of 0.999 while only 10% positive edges are sampled, in which 75% is used for training. The actual AUC is about 0.76, a significant deviation from our test. We conjecture that this is because in our test, we are actual using part of existing connections to predict deliberately hidden connections. The actual AUC is calculated on future connections. These two things are fundamentally different and latter one is expected to be lower than previous one. This fact renders performing local test with existing data useless. Nonetheless, we still split the test set and do testing to estimate an upper bound for the performance. The shown results are discussed based on public test-set data on Kaggle if not specified otherwise.

Tuning walk length w results in a rapid increasing and slow decreasing trend as shown in Table 3. For w=1, the AUC is only 0.51, indicating an significantly inability to capture the relationship between authors. AUC increases with w up to 3, where it culminates at 0.77. With further increase in w, AUC slowly decreases. Our test ended at w=15 with AUC=0.74. This parameter controls the walk length to generate features. Small w means dominance of local connection, while large w means a broad connection range. Combining results from different w will enable us to collect prediction from different breadth of connection, providing possible accuracy gain. w=4 is selected as new default value for later tuning.

Tuning feature dimension d does not result in significant change. Reducing d by 20% to 80 results in marginal performance decrease. However, increasing d by 20% to 120 results in some degradation from 0.770 to 0.761. This may mean d=100 is at the border line for catastrophic over-fitting. We keep d=100 as the default.

Tuning p and q also does not result in significant change. We individually tuned p and q to be 0.8 and 1.2. All of them result in worse results. It may indicate that edges out from and returning to the author are equally important. No preference for remote node or local node should be necessary.

The final result was an equal combination of many different prediction. We collected results from w=2~9, d=80,120, p=1.2, and q=0.8. The final result was the average of all of them. Although none of individual predictions had AUC>0.78, the averaged results surprisingly gave AOC=0.80. We also tested with only different w from 2 to 6, the result was slightly lower but still higher than 0.80.

7 Conclusion

In this paper, we propose several graph relationship transformation method for heterogeneous academic network. Then we generate graph embedding and adjacency matrix for the node classification and link prediction. For the node classification task, we proposed a new heterogeneous graph convolutional network. For the link prediction task, we find out that node2vec combined with negative sampling perform well enough that other complex methods are useless. No matter which task, we find the key to good perfoemance is selecting good parameters and graph structure for node2vec.

8 Contribution

Zhenyang Ni implemented the HetGCN and proposed methods for node2vec parameter selection and relation transformation.

Qiuwen Wang implemented the homogeneous paper GCN and tuned the parameters and kernels in the network.

Haochen Zhao implemented all data pre-processing and link prediction code. He also tuned the parameters for node2vec.

References

[1] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Graph neural networks with convolutional arma filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[3] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *arXiv preprint arXiv:1606.09375*, 2016.

[4] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 869–877, 2018.

[5] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[6] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*, 2017.

[7] Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Wi5KUNlqWty.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[10] Yunsheng Shi, Zhengjie Huang, Wenjin Wang, Hui Zhong, Shikun Feng, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

[11] Zhaowei Tan, Changfeng Liu, Yuning Mao, Yunqi Guo, Jiaming Shen, and Xinbing Wang. Acemap: A novel approach towards displaying relationship among academic literatures. In *Proceedings of the 25th international conference companion on world wide web*, pages 437–442, 2016.

[12] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.