Graph Mining: Node classification and Link Prediction on an Academic Network

Li Yufan[1], Tang Ziang[2] and Shen Zhen[3]

Abstract— **In this project, we study the representation learning problems of network, both heterogeneous and homogeneous one. We put our emphasis on node embedding algorithms to address a node classification problem and a link prediction problem. In the process of the experiment, we build a heterogeneous network based on metapath2vec to solve the node classification problem, which is dependent on chosen meta path.Then a homogeneous network based on node2vec is implemented to handle the link prediction problem without need for meta path in contrast to the former.**

I. Introduction

Machine Learning is an highly efficient way to mine the deeper mysteries behind massive data. Also, Data-driven machine learning is starved of raw input data to provide motivation for training and learning. However, experiments shows that the artificial output prediction sinking or swimming depends on the structure or representation of input data, which is originally chosen by human named after **Feature Engineering**. It reveals a drawback of machine learning that it is too superficial for complex data structures and connections to be captured. The phenomenon means that basic machine learning can hardly be called artificial intelligence on account of its high dependency on human beings. Therefore, we hope to establish new programs to remedy the defect for an artificial agent looking for latent regulations that cannot be easily found otherwise by human, liberating data pre-processing from redundant feature engineering. That is how **Representation Learning** is put forward.

Representation learning plays a key role in massive data mining with complex construction before everything. It operates as a tache of easing the way the agent treats the complex data. Through representation learning, the agent shows more reliability when building a classifier or predictor. As a result, the performance is bettered.

The most extensively used and effective way for representation learning is deep learning. It is ground truth that majority of the world stuff is complex structured instead of sheer numbers, for example, images. Generalizing neural networks has been applied to tackle learning problems so far, with experimental subjects of image, sentence, time series, etc. Almost all domains of artificial intelligence are embracing representation learning. It becomes indispensable and versatile, typical in image processing, natural language processing, knowledge graph, etc.

Our work is based on a graph, a natural form for representation of links or connections between nodes, which often embodies social relations of human. Graph is not a burgeoning data structure, and it is easily represented as an adjacent matrix. The problem lies in representation of nodes or links, also called **Network Embedding**.

Previous works have cast light on network embedding based on **Unsupervised Learning**. The application condition is divided into two groups on the basis of the characteristics of the graph itself — whether it is homogeneous or heterogeneous.

For homogeneous graph, all the nodes are of the same type. Accomplishments achieved in NLP is utilized in homogeneous network embedding, regarding a chain based on random walk with from node to node as a sentence and nodes as words. The ground breaking work is **Deepwalk**[1] put forward in 2014 based on random walk, by randomly sampling one link leading to a next node and implementing word2vec. **Node2vec**[2] in 2016 extends Deepwalk by introducing BFS to random walk process. Meanwhile **SDNE**[3] in 2016 brings in Auto-encoder for dimensionality reduction. For the last two years, new algorithms such as **GAT**[4] and **DGI**[5] sprang up that are independent of random walk model.

For heterogeneous network, the most famous and proved successful is **Metapath2vec**[6] in 2017. The process includes three steps:

- Random walk based on meta path.
- Heterogeneous Skip-gram.
- Negative sampling.

In our work, we construct a heterogeneous network and implement metapath2vec for node embedding to deal with the multi-label node classification problem, denoting allotting several conferences as labels to each author and predicting their future publications. We also build a homogeneous network for link prediction problem to predict the latent links between authors through node2vec. Generally speaking, the outcome is satisfactory.

II. Methodology

We will give the problem definition and introduce the techniques we use in our project in this section. We leverage the definition of graph and heterogeneous networks in paper [6], [7] to form our problem.

[1] Li Yufan is B.Eng student with Faculty of Electronic Information and Electrical Engineering, IEEE, Shanghai Jiao Tong University, 800 Dongchuan, Shanghai, China charon@sjtu.edu.cn

[2]Tang Ziang is B.Eng student with Faculty of Electronic Information and Electrical Engineering, Information Engineering, Shanghai Jiao Tong University, 800 Dongchuan, Shanghai, China tza991101@sjtu.edu.cn

[3]Shen Zhen is B.Eng student with Faculty of Electronic Information and Electrical Engineering, Automation, Shanghai Jiao Tong University, 800 Dongchuan, Shanghai, China shenzhen2000@sjtu.edu.cn

A. Problem definition

A simple undirected **graph** *G = (V, E)* is a collection V of n vertices $v_1, v_2, \dots, v_n$ together with a set of E of edges, which are unordered pairs of the vertices. A **heterogeneous network** is a graph *G = (V, E, T)* in which each node v and each edge e are associated with their mapping functions $\phi(v) : V \to T_V$ and $\varphi(e) : E \to T_E$ respectively. And T_V and T_E represent the set of object and relation types where $|T_V| + |T_E| > 2$. Given a graph $G = (V, E)$ with n nodes, a network embedding maps each node $v \in G$ to a compact feature vector in $\mathbb{R}^d (d \ll n)$ such that the embedding vectors can capture the graph features surrounding v. A homogeneous network embedding is a type of network embedding that reflects the topology of G rather than labels associated with nodes or edge. The input of the problem is a graph and the output is a matrix whose $i-th$ row is a d dimensional vector representing a node in the graph.

B. Network embedding

We use node2vec[2] and metapath2vec[6] based methods to generate homogeneous and heterogeneous networks respectively for the link prediction and node classification problems. The first steps of the two methods are both random walk. This step aims to transform the structure of a network into skip-gram by incorporating the node paths traversed by random walkers over a network into the neighborhood function. The skip-gram structures used in node2vec and metapath2vec are the same and shown in fig. 1. The skip-

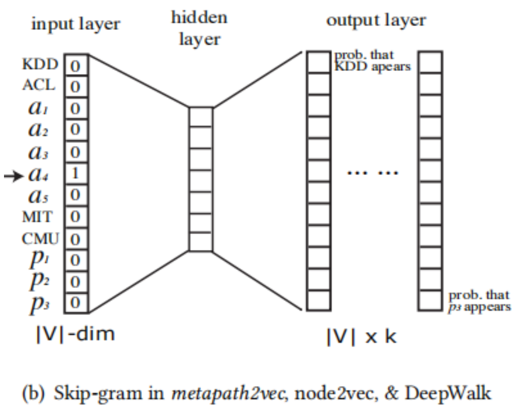

Fig. 1. Skip-gram in metapath2vec and node2vec[6]

gram model is used to learn continuous feature representation for words by optimizing a neighborhood preserving likelihood objective. And in a network, "nodes" are regarded as "words". By extending the skip-gram architecture to networks, the objective function becomes:

$$arg \max_{\theta} \sum_{u \in V} \sum_{n_t \in N_t(u)} logPr(n_t|u;\theta)$$

which maximize the log-probability of observing a network neighborhood $N_t(u)$ for a node u conditioned on its feature representation given by θ. Pr function is commonly defined as a softmax function. In metapath2vec, in order to model the heterogeneous neighborhood of a node, the objective function takes types of nodes into account:

$$arg \max_{\theta} \sum_{u \in V} \sum_{t \in T_u} \sum_{n_t \in N_t(u)} logPr(n_t|u;\theta)$$

where $N_t(u)$ denotes u's neighborhood with the t^{th} type of nodes. The objective function can be approximated using negative sampling[8] to sample a small set of words(nodes) from the corpus(network) to avoid expensive computational cost when the network is large and optimized using stochastic gradient descent.

Random walk applied by node2vec is biased which means that neighbors of a node v are not accessed with the same probability. And node2vec uses a 2nd order biased random walk approach with parameters p and q to guide the walk. The bias α is associated with the shortest path distance d_{12} between the two nodes v_1 and v_2 where a random walk just traversed the edge (v_1, v_2) and now resides at v_2 and it has to decide where to go next. The probability distribution from v_2 to it's neighbors xs denotes as $\pi_{v_2 x} = \alpha_{v_1 x}$:

$$\alpha_{v_1 x} = \begin{cases} \frac{1}{p} & d_{v_1 x} = 0 \\ 1 & d_{v_1 x} = 1 \\ \frac{1}{q} & d_{v_1 x} = 2 \end{cases}$$

The illustration of random walk in node2vec is shown in fig. 2. As we can see in the figure, parameter p controls the likelihood of immediately revisiting a node in the walk (from v_2 back to v_1). If its value is high, it's less likely to sample an already-visited node in the following step which avoids which encourages moderate exploration and avoids 20hop redundancy. Parameter q allows the search to distinguish the "inward" and "outward" nodes. If value of q is large than 1, the walk is biased to the common neighbor of v_1 and v_2 this is to some degree like the behaviour of BFS.

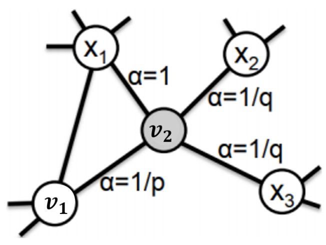

Fig. 2. Illustration of the random walk procedure in node2vec.

Random walk used in metapath2vec is driven by a meta path that defines the node type order by which the random walker explores the graph and thus it can be used in heterogeneous network. A metapath scheme $\mathcal{P}$ is defined as a path that is dennoted in the form of $V_1 \xrightarrow{R_1} V_2 \xrightarrow{R_2} \cdots V_t \xrightarrow{R_t} V_{t+1} \cdots \xrightarrow{R_{l-1}} V_{l-1}$ where in $R = R_1 \circ R_2 \circ \cdots \circ R_{l-1}$ defines the composite relations between node types V_1 and V_l. For example in fig. 3, a metapath "APPA" represents the citation relationships on a paper P cited by another paper between two authors A.

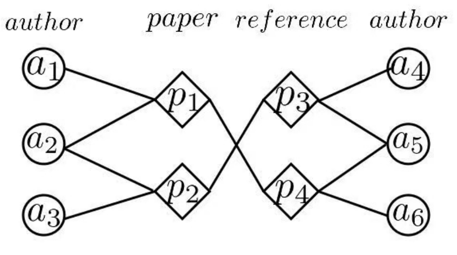

Fig. 3. A example of network

Random walks are computationally efficient compared to BFS or DFS search strategy to construct a corpus in both time and space. The space complexity to store the immediate neighbors of all nodes in a graph is $O(|E|)$. For 2nd order random walk, it is helpful to store the interconnections between neighbors of every node, so the space complexity can be $O(a^2|V|)$ where a is the average degree of the graph and usually small for real world networks.

After we get the learnt feature representations of all the nodes in the graph, we can do node classification task. But for link prediction, we get the feature of links on the basis of nodes. We tried four operations to produce link embedding from node embedding:

- Hadamard—-$u * v$
- L1—-$|u - v|$
- L2—-$(u - v)^2$
- Average—-$\frac{u + v}{2}$

where v and u are two endpoints of an edge.

C. Classification & Prediction

We use decision tree and MLP (multilayer perceptron) to complete the node classification task after obtaining the representation of the nodes. Decision tree is used as a baseline. The basic structure of MLP is shown in fig. 4. It has 5 layers with one input and output layer and 3 hidden

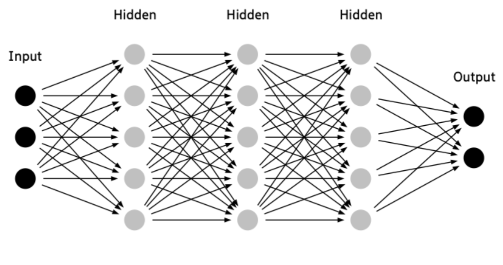

Fig. 4. Basic network structure of MLP

layers. And we add another 3 dropout layers to boost our network performance. For link prediction task, we simply use logistic regression to do the binary classification problem and the probability we get from the classifier is used as the prediction result.

III. Experiment

We carry out a series of experiments, implementing metapath2vec and node2vec to solve different downstream tasks, i.e. node classification and link prediction, respectively.

Data used in this project are provided by Acemap group. This dataset consists of 42,614 authors and corresponding 24,251 papers from 10 top conferences (2016 - 2019) in the field of Artificial Intelligence and Data Mining as well as citation information of their publications.

A. Node Classification

For the node classification task, we construct a heterogeneous collaboration network, in which there are two types of nodes: authors and papers. The edges represent different types of relationships among two sets of nodes. Edges between author nodes represent co-authorships, and edges between authors and papers represent authorship relationships, while edges between papers represent citation relationships. In summary, the heterogeneous network consists of 42,614 author nodes, 24,251 paper nodes, 181,338 author-author edges, 93,230 author-paper edges, and 24,525 paper-paper edges.

In the network, each author node has multiple labels: if he/she has published papers in conference A, he/she will be labeled as A; if he/she has published papers in several different conferences, he/she will get multiple labels. Label information of 20% nodes (both author nodes and paper nodes) is given, and our task is to predict the labels of the rest 80% author nodes.

Note that the node representations are learned from the full dataset. The embeddings of above labeled nodes are then used as the input to a multi-label classifier and the remaining nodes are for testing. We tune the hyperparameters and try different classifiers, repeat the prediction experiments and report the performance in terms of F1 scores.

The metapath2vec representation learning algorithm is implemented using components from the stellargraph [9] and gensim [10] libraries.

Results. fig. 5 visualizes the embeddings of author and paper nodes, projected from 128-dimensional vectors down to 2 dimensions. table I lists the results of our experimental process. The walk lengths are all 100 and word2vec epoch numbers are all 5, thus not listed in the table. We analyze the experimental results in several aspects.

Multi-label classifier. Different classifiers vary in their performances given the same node embeddings. In the beginning we tried the Decision Tree classifier and got F1 scores less than 0.40. After switching to MLP classifier, we achieved better performance with F1 scores more than 0.45.

Parameter sensitivity and tuning. In skip-gram-based metapath2vec model and downstream MLP classifier model, there exist several parameters that decide the embedding quality and multi-label classification performance. According to [6], the increase of author classification performance

TABLE I

MULTI-LABEL AUTHOR NODE CLASSIFICATION PARAMETERS AND RESULTS

Metapath	Number of walks per node	Embedding dimension	Number of neighbors	Classifier	MLP training epochs	Confidence threshold	F1 score
APA, APPA	1	128	5	DecisionTree	N/A	N/A	0.36348
APA, APPA, APPPA	5	128	7	DecisionTree	N/A	N/A	0.37049
APA, APPA	100	128	3	DecisionTree	N/A	N/A	0.32208
APA, APPA	1	128	5	Baseline MLP	100	0.5	0.44957
APA, APPA	1	256	5	Baseline MLP	100	0.5	0.44898
APA, APPA, AA, APPAA	1	128	5	Dropout layer MLP	200	0.28	0.49111
APA, APPA, AA, APPAA	1	128	5	Dropout layer MLP	200	0.2	0.50476
APA, APPA, AA, APPAA	1	128	5	Dropout layer MLP	200	0.1	0.51140
APA, APPA, AA, APPAA	1	128	5	Dropout layer MLP	150	0.1	0.51201
APA, APPA, AA, APPAA	1	128	5	Dropout layer MLP	150	0.08	0.51463
APA, APPA, AA, APPAA	1	128	5	Dropout layer MLP	150	0.06	0.51330

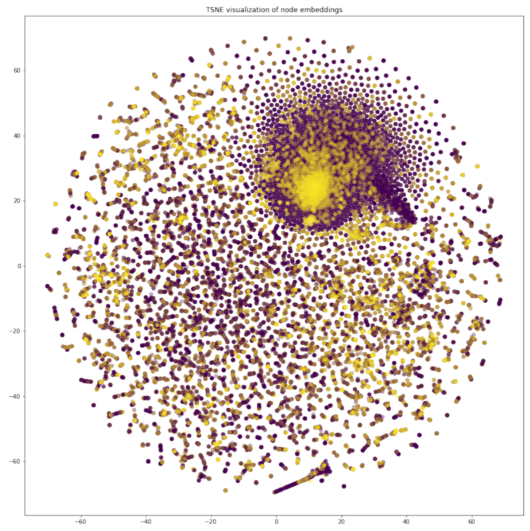

Fig. 5. Visualization of node embeddings

converges as the number of walks w rooting from each node and the length l of each walk reach around 1000 and 100, respectively, and a smaller neighborhood size k actually produces the best embeddings for separating authors, which differs from the case in a homogeneous environment. However, tuning the above parameters didn't give out an improvement in the classification performance, and F1 score even dropped by more than 0.04 after we increased w significantly and decreased k. We think that it is due to the different characteristics of the datasets and structures of the graphs accordingly. With far less nodes and only two node types, the semantic representation of the heterogeneous graph may not be well developed, even though conducting a large number of random walks and changing the metapaths. Overall, metapath2vec is not strictly sensitive to these parameters, thus we made a relatively cost-effective parameter choice. We set number of walks per node $w = 1$, walk length $l = 100$, neighborhood size $k = 5$, embedding dimension $d = 128$, and set metapaths as "APA, APPA, AA, APPAA" in following experiments.

On the other hand, performance can be greatly improved by tuning the structure of the MLP neural network. The baseline MLP consists of the input layer with 128 neurons, three hidden layers with 200, 200, 100 neurons respectively, and the output layer with 10 neurons that represent the probabilities of the classes of the author after a sigmoid function. After adding three dropout layers whose dropout rates are 0.5, the validation loss drops significantly as shown in fig. 6 (the sequential_2 and sequential_4 in the figure are baseline MLP and dropout layer MLP respectively), and the dropout layer MLP classifier outperforms the baseline MLP classifier due to the reduction of overfitting and the learning of more robust features by the neural network.

F1 metric and confidence threshold. The evaluation metric for this task is Mean F1-Score. The F1 score, commonly used in information retrieval, measures accuracy using the statistics precision p and recall r. Precision is the ratio of true positives tp to all predicted positives $tp + fp$. Recall is the ratio of true positives tp to all actual positives $tp + fn$. The F1 score is given by:

$$F1 = 2\frac{p \cdot r}{p + r} \quad \text{where} \quad p = \frac{tp}{tp + fp}, \quad r = \frac{tp}{tp + fn}$$

Multi-label deep learning classifiers usually output a vector of per-class probabilities, and these probabilities can be converted to a binary vector by setting the values greater than a certain threshold to 1 and all other values to 0. This threshold is known as the confidence threshold. The choice of confidence threshold affects what is known as the precision/recall trade-off. Increasing the threshold increases precision while decreasing the recall, and vice versa. The higher we set the confidence threshold, the fewer classes the model will predict. This is because as we increase the confidence threshold less classes will have a probability higher than the threshold. This leads to the model having higher precision, because the few predictions the model makes are highly confident, and lower recall because the model will miss many classes that should have been predicted. On the other hand, the lower we set the confidence threshold, the more classes the model will predict. This leads to the model having higher recall because it predicts more classes so it misses fewer that should be predicted, and lower precision because it makes more incorrect predictions.

In our case, we found the best results when the threshold is around 0.08 through trial and error.

Other issues. There are some tricks that can further improve the performance. First, the embeddings can be normalized before sending them to the downstream clas-

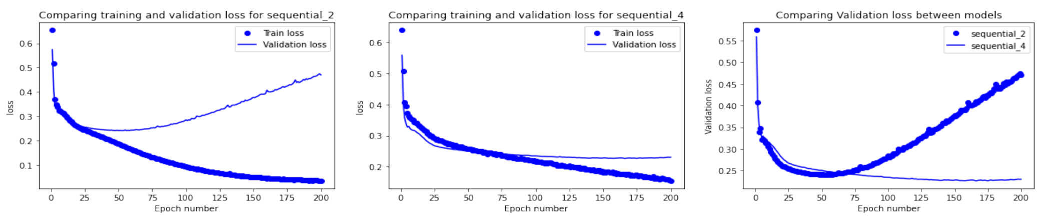

Fig. 6. Comparing validation loss between models

sifier. Second, the given label information can replace the corresponding predicted author labels as they are ground truths. (The above two tricks could both actually elevate the performance slightly in our experiments. The following tricks are just tentative ideas.) Third, in our experiments when the output of the classifier allocates no label to an author because the probabilities of all labels are below the threshold, we just assign him/her the label with the highest probability. However, there possibly are better strategies. Fourth, some voting/model fusion strategy can be adopted, which means comparing several well-performed classification results and doing some filtering and selection based on them. Fifth, the years of publication can be added as features of author and paper nodes.

B. Link Prediction

In link prediction, we first pre-process the dataset and form a homogeneous network where each node represents an author, and each edge represents the citation relation or co-authorship between the two connected authors. In summary, the homogeneous network consists of 42,614 nodes and 842,297 edges. It is worth mention that there are 564 isolated nodes, which means that the 564 authors are not engaged in any relationship with any other author. Then we learn the node embeddings from the full dataset.

Next we split the network into **Train Graph**, **training set of link examples**, and **set of link examples for model selection**: Randomly sample a fraction 0.1 of all positive links, and same number of negative links, from the original network, and obtain the reduced graph **Train Graph** with the sampled links removed. The latter two subsets are arranged in a 0.75:0.25 ratio. Calculate edge embeddings for the positive and negative edge samples, which are the **training set of link examples**, by applying a binary operator on the embeddings of the source and target nodes of each sampled edge. Given the embeddings of the positive and negative examples, we train a logistic regression classifier to predict a binary value indicating whether an edge between two nodes should exist or not. Then we evaluate the performance of the link classifier for each of the 4 operators-Hadamard, L1, L2 and average operator-on the **set of link examples for model selection** with node embeddings calculated on the **Train Graph**, and select the best classifier. The best classifier is then used to calculate scores on the test data with edge embeddings calculated on the complete network.

We repeat the prediction experiments and report the performance in terms of AUC metric.

The node2vec representation learning algorithm is implemented using components from the stellargraph [9] and gensim [10] libraries.

Results. table II lists our experimental results. The Hadamard operator turns out to be the best binary operator in almost all experiments. p and q are the return parameter and in-out parameter that guide the biased random walk in the original paper. The embedding dimensions are all 128 and neighborhood sizes are all 10, thus not listed in the table.

Parameter sensitivity and tuning. Because the best p and q parameter settings for each node2vec entry are omitted for ease of presentation in [2], we turn to [11] for reference. As is claimed in the paper which compares all kinds of unsupervised network representation learning methods, these two parameters do not play a huge role in the performance of node2vec: AUC remains around 0.68 when we keep other parameters unchanged and tune only p and q. Also in [11] the authors summarize that for the LP task for both undirected graphs, PPR based methods and LINE that directly uses adjacency matrix as context matrix outperform random walk based methods, and node2vec favors graphs with high clustering coefficient and high reciprocity. That may explain the unsatisfactory performance of node2vec for link prediction. According to fig. 7 (fig. 7 is a subset of the entire graph, and contains 500 nodes and more than 80,000 edges), the graph is not highly clustered.

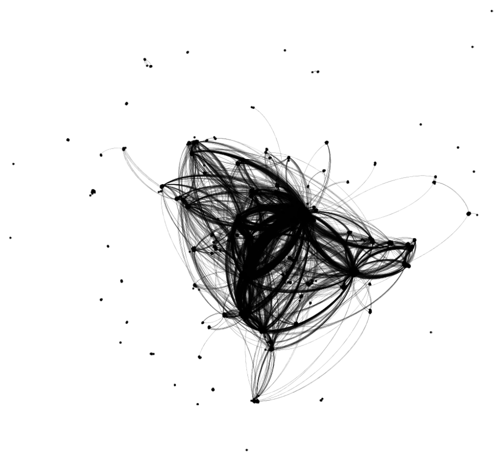

Fig. 7. Homogeneous network for link prediction

In addition, we encountered a seemingly unlikely problem

TABLE II

LINK PREDICTION PARAMETERS AND RESULTS

p	q	Number of walks per node	Walk length	Word2vec epochs	AUC
1	1	1	80	5	0.67
0.8	0.5	5	80	5	0.65
0.8	0.5	1	80	5	0.68
0.5	0.5	1	80	5	0.68
0.25	4	1	80	1	0.63
0.25	4	1	80	5	0.68
0.25	4	1	80	10	0.66
0.25	4	1	80	50	0.64

when tuning the word2vec epochs. In fig. 8, we observe that the loss keeps decreasing with the increase of the number of epochs. However, the AUC metric reaches its peak when epoch number is around 5, and then goes down with the increase of the epoch number. From this we infer that the per-epoch loss isn't a direct measure of model quality for outside purposes, but only an indicator whether training is still helping on its internal optimization task. Further more, it possibly indicates that the overfitting is occurring, for the (possibly oversized) model is memorizing features of the (likely undersized) training data, and therefore getting better at its internal optimization goals in ways that no longer generalize to the test dataset.

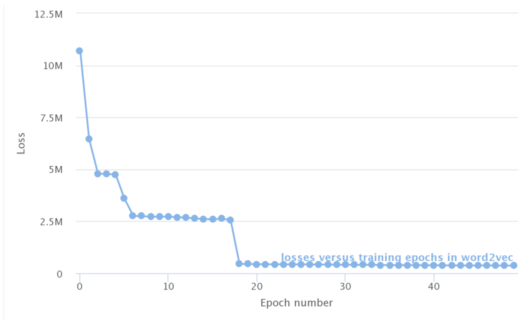

Fig. 8. Word2vec training losses

IV. Conclusion

In this work, we study the problem of representation learning of nodes in homogeneous and heterogeneous network structure and apply node embedding to further research, to be specific, node classification and link prediction, in order to inspect the performance of our node embedding algorithms. The network we need to deal with is an academic network of authors and papers as nodes and links as edges denoting connections like co-authors, publishing, reference, etc.

For node classification, we construct a heterogeneous network with raw data of authors and papers together with their links. Then we choose metapath2vec algorithm to get node embedding. First, we allow a random walk along the heterogeneous network links on the basis of certain meta paths, which is effective to learn the latent links and semantic contexts of nodes, forming a corpus. Then we utilize word2vec algorithm to realize skip-gram model. It is used to get the optimization of parameters to guarantee the max probability of certain words(here refers to nodes) appearing in the context. Finally, we accomplish optimization by negative sampling to reduce the occupation of time and space. After getting node embeddings, we build a suitable neural network model with training data of authors and their labels to predict other authors labels. Considering it as a multi-label problem, labels are coded as one-hot codes to simplify the construction of classifier and predictor. By evaluating it with Mean F1-score, we achieved a decent score on the private leaderboard of the bronze medalist.

For link prediction, we decide to construct a homogeneous network with only authors as nodes. We do with data pre-processing to achieve data fusion of the past four years. We then simply apply node2vec algorithm to get node embedding. At last, Logistic Regression is employed to predict links possible for the next year. We use AUC as evaluation index and get a score on the private leaderboard — not so good but worthy for reference. Our problem lies in that we ignore the meaningful information of temporal tendency and only focus on the overall situation, which is an ex parte strategy. Future research includes how to add temporal information in node representation learning. We conceive that node2vec can be part of the model for learning node embedding of each snapshot — of one year. Then LSTM may come into play for its time-based characteristics. GAN is a choice to restore the network with embeddings predicted by LSTM. And since we have got a heterogeneous network embedding, we can also use this to generate the link representation as mentioned in section II-B and try some other operations to transform node representation to edge representation. Furthermore, more works about representation learning in network is to be done.

V. Individual Contributions

- Li Yufan: node classification & report & PPT
- Tang Ziang: node classification & link prediction & report & PPT
- Shen Zhen: node classification & report & presentation

References

[1] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, p. 701–710.

[2] A. Grover, J. Leskovec, Node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, p. 855–864.

[3] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, p. 1225–1234.

[4] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks (2018). arXiv:1710.10903.

[5] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, R. D. Hjelm, Deep graph infomax (2018). arXiv:1809.10341.

[6] Y. Dong, N. V. Chawla, A. Swami, metapath2vec: Scalable representation learning for heterogeneous networks, in: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, 2017, pp. 135–144.

[7] H. Chen, B. Perozzi, R. Al-Rfou, S. Skiena, A tutorial on network embeddings, arXiv preprint arXiv:1808.02590 (2018).

[8] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, arXiv preprint arXiv:1310.4546 (2013).

[9] C. Data61, Stellargraph machine learning library, `https://github.com/stellargraph/stellargraph` (2018).

[10] R. Řehůřek, P. Sojka, Software Framework for Topic Modelling with Large Corpora, in: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, ELRA, Valletta, Malta, 2010, pp. 45–50, `http://is.muni.cz/publication/884893/en`.

[11] M. Khosla, V. Setty, A. Anand, A comparative study for unsupervised network representation learning, IEEE Transactions on Knowledge and Data Engineering (2019).