Academic Network Embedding & Downstream Applications

Yifei Li
518030910306

Haoning Wu
518030910285

Longrun Zhi
518030910320

ABSTRACT

Prediction task over nodes and edges in networks require effective and accurate features produced by feature learning algorithms. Recent research in the broader field of representation learning has led to significant progress in automating prediction by learning the features themselves. However, for graph network with specific structure and specific data format, we cannot just apply these existing algorithms easily without thinking and specific data processing, which otherwise will lead to not very good feature results. Besides, the work we researched doesn't claim and verify the downstream models or algorithms they use clearly.

Here, we first specify that the graph network type is an academic graph with authors or papers as nodes and author or paper relationship as edges. According to the specific graph data, we make data pre-processing first and construct homogeneous network and heterogeneous network to represent the graph respectively. Through the experiments, we also find the data deficits of this specific graph, that is the existence of isolated nodes and we design several ways to deal with the deficits.

We mainly use node2vec for homogeneous network embedding and metapath2vec for heterogeneous network embedding. What's more, for downstream application, we claim and design several different algorithms and models.

KEYWORDS

Academic Network, Network Embedding, Multi-Label Node Classification, Link Prediction

1 INTRODUCTION

Many important tasks in network analysis involve the prediction of nodes and edges. In a typical multi-label node classification task, we are interested in predicting the most probable labels of nodes in a network. For example, in a social network, we might be interested in predicting interests of users, or in a protein-protein interaction network we might be interested in predicting functional labels of proteins, or in an academic network like one in this project, we are interested in the conferences that authors will probably publish papers on. Similarly, in link prediction task, we wish to predict whether a pair of nodes in a network should have an edge between them. For instance, in genomics, we are interested in discovering novel interactions between genes, and in social networks, we are interested in the friendship of two people. Besides, for academic network like the one in this project, we are interested in the cooperative relationship of two authors in next year.

However, it's hard to get the information of nodes and edges in network directly due to the complexity and high-dimensionality of graph network. In general, any supervised machine learning algorithm and some unsupervised algorithms require a set of informative, discriminating and independent features. In prediction problems on networks, this means that we need to construct a feature vector representation of nodes and edges. And the process above is exactly **Network Embedding**, which we want to talk about.

Therefore, the main idea of such task is as follows(see Figure 1):

(1) Construct the graph network using nodes and edges;
(2) Use network embedding algorithms or models to obtain the representation of nodes or edges;
(3) Make link prediction or node classification based on the representation vector, i.e. embeddings.

In fact, network embedding is a relatively mature research area. No matter the graph network is homogeneous or heterogeneous, there are corresponding embedding algorithms and models, such as Deep Walk(Perozzi et al. [9]), node2vec(Grover and Leskovec [6]) for homogeneous graphs and metapath2vec(Dong et al. [5]) for heterogeneous graphs. And basically, these works conducted experiments for link prediction and node classification tasks as evaluation on the performance of their representational learning algorithms.

However, current research works mainly focus on the network embedding techniques and their problem is relatively general. Not only did they fail to deliver a systematic process from nodes and edges to prediction results with an end-to-end model or claim and verify the downstream algorithms and models they used, but their works are so general that they cannot be applied to specific graph data greatly like the one for our project. What's more, they commonly use single graph construction type, either homogeneous or heterogeneous.

Alternatively, we can design an end-to-end system with nodes and edges as input and prediction results as output. We not only focus on the embedding we get, but also want to find the appropriate and effective algorithms or models for downstream applications. We choose specific graph data(academic network) and we can pre-process the specific data to make the embedding model we used fit our task. We also try different types of network construction and conduct extensive experiments.

Our Work. We research many kinds of network embedding works such as DeepWalk, node2vec, SDNE, Struct2vec, GAT and DGI for homogeneous graph, and metapath2vec for heterogeneous graph. We have implemented all above algorithms except GAT and DGI and carry out extensive experiments for corresponding algorithms.

Our **key** contribution is: Through the experiments, we find the existence of isolated nodes in the graph, which is the data deficit of this specific academic network in our project. And to deal with the isolated nodes, we try several methods: (1) giving random results, (2) adding self-loops and (3) constructing heterogeneous graph. After that we analyzed the shortcoming of these methods, in order

Conference'21, June 2021, Shanghai, SJTU, China

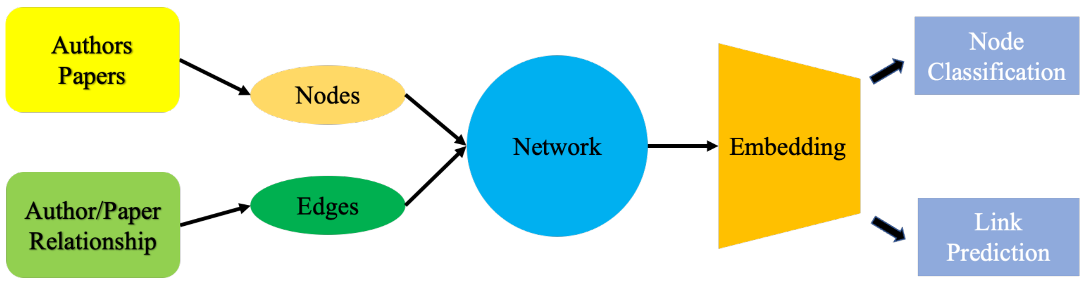


Figure 1: Main idea of academic network embedding and downstream applications: 1. construct the network 2. get the embedding 3. predict based on embedding

to improve the correlation of the data, we try to add features to nodes to get representation together and propose new ideas for node classification: use the prediction of paper labels to reflect the author labels.

We also propose several kinds of methods for link prediction and node classification: For link prediction, We first use cosine similarity to estimate the likelihood of a relationship between nodes and then we use tanimoto coefficient to estimate and make comparisons.While for node classification, we first use traditional multi-label classifier in scikit-learn and then we proposed a new deep learning model with full connection layers and 1D convolution layers to make prediction.

Our experiments are abundant. The comparisons we make can show the effectiveness of the model we mainly use. The large amounts of different parameter settings show the process where we try to get better results. And the methods and tricks we use to deal with isolated nodes getting better results shows the correctness of our key idea. What's more, the visualization we make shows the correctness and success of our embedding.

Overall our paper makes the following contributions:

1 We use both homogeneous and heterogeneous network constructions and we have implemented DeepWalk, node2vec, SDNE and Struct2vec to get embeddings for homogeneous graph and metapaht2vec for heterogeneous graph.
2 We find the data deficit in academic graph of this project: the existence of isolated nodes and we propose random prediction, self-loops construction and heterogeneous construction to deal with the problem.
3 We propose new tricks to improve data correlation: use nodes together with features to get embedding. And we propose new idea of node classification: classify single label to get the prediction of paper labels and then use author-paper relationship to get the corresponding author labels.
4 We claim and verify the methods or models we use for downstream classification and we design a new deep model for node classification.
5 We evaluate our model appropriately: we do a lot of contrast experiments on different embedding models, different parameter settings and different downstream models. And we make visualization to show the correctness of our embeddings.
6 At last, combining all above, we get an end-to-end system for our project.

The rest of the paper is structured as follows. In Section 2, we survey the related work in feature learning for networks and link prediction and node classification based on graph embedding. In Section 3, we clarify the problem definition of our project on academic network embedding. We propose our key ideas about the discovery, analysis and solution of isolated nodes in this specific academic network in Section 4. In Section 5, we deliver the models and algorithms which we use to get embeddings and make link prediction or node classification. Then we show our experiments and results in Section 6. After that, we make conclusion in Section 7 and clarify our team contribution in Section 8 .

2 RELATED WORK

2.1 Network Embedding

Network Embedding usually has two goals, i.e., **network reconstruction** and **network inference**. Traditionally, Network Embedding is regarded as a process of dimensionality reduction, and the main methods include Principal Component Analysis(PCA) and Multidimensional Scaling(MDS). Most methods can be understood as using an $n \times k$ matrix to represent the original $n \times m$ matrix, where $k << m$. The traditional graph embedding methods, mainly focus on network reconstruction. And the recently proposed network embedding methods aim to address the goal of network inference. The related works can be divided into three categories: (1) Structure and property preserving network embedding; (2) Network Embedding with Side Information; (3) Advanced Information Preserving Network Embedding.

To transform networks from original network space to embedding space, different models can be adopted to incorporate different types of information or address different goals. The commonly used models include matrix factorization, random walk, deep neural networks and their variations.

For **Matrix Factorization**, an adjacency matrix is commonly used to represent the topology of a network, where each column and each row represent a node, and the matrix entries indicate the relationships among nodes. Network embedding, aiming to learn a low-dimensional vector space to represent a network, in contrast with the $N-$dimensional space. In this sense, matrix factorization methods, with the same goal of learning low-rank space for the original matrix, can naturally be applied to solve this problem. In the series of matrix factorization models, Singular Value Decomposition(SVD) is commonly used in network embedding due to its optimality for low-rank approximation. Non-negative matrix factorization is often used because of its advantages as an additive model.

For **Random Walk**, neighborhood structure, describing the local structural characteristics of a node, is important for network embedding. Although the adjacency vector of a node encodes the first-order neighborhood structure of a node, it is usually a sparse, discrete, and high-dimensional vector due to the nature of sparseness in large-scale networks. The key problem is how to define "neighborhood" in networks. To make analogy with Word2Vector (Rong [12]), random walk models are exploited to generate random paths over a network. By regarding a node as a word, we can regard a random path as a sentence, and the node neighborhood can be identified by co-occurrence rate as in Word2Vector (Rong [12]). Some representative methods include DeepWalk (Cai et al. [2]) (Chen et al. [3]) (Cui et al. [4]) (Perozzi et al. [9]) and node2vec (Grover and Leskovec [6]).

For **Deep Neural Networks**, if seeking for an effective non-linear function learning model, deep neural networks are certainly useful options be-cause of their huge successes in other fields. The key challenges are how to make deep models fit network data, and how to impose network structure and property-level constraints on deep models. Some representative methods, such as SDNE (Wang et al. [16]) and SiNE (Wang et al. [17]), propose deep learning models for network embedding to address these challenges.

LINE (Tang et al. [13]) uses the Breadth-First Search(BFS) strategy to generate context nodes: only nodes that are at most two hops away from a given node are considered as neighboring nodes. In addition, compared to the hierarchical softmax used in DeepWalk (Perozzi et al. [9]), it uses negative sampling to optimize the Skip-gram model. **Node2vec** (Grover and Leskovec [6]) is an extension of DeepWalk (Perozzi et al. [9]), which introduces a biased random walk, and combines DFS style and BFS style neighborhood exploration so that it can learns the **homophily** and **structural equivalence** information of nodes in the network. **Struc2vec** (Ribeiro et al. [11]), a novel and flexible framework to learn representations that capture the structural identity of nodes in a network, which overcomes limitation by focusing explicitly on structural identity. It assesses the structural similarity of node pairs by considering a hierarchical metric defined by the ordered degree sequence of nodes and uses a weighted multi-layer graph to generate context. **Walklets** (Perozzi et al. [10]) shows that DeepWalk (Perozzi et al. [9]) learns network embeddings from a weighted combination of A^1, $A^2, \ldots, A^k$. In particular, DeepWalk (Perozzi et al. [9]) is always more biased toward A^i than A^j if $i < j$. To avoid this, Walklets (Perozzi et al. [10]) proposes to learn multi-scale network embeddings from each of A^1, $A^2, \ldots, A^k$. Since the time complexity of computing A^i is at least quadratic in the number of nodes in the network, Walklets (Perozzi et al. [10]) approximates A^i by skipping over nodes in short random walks. It further learns network embedding from different powers of A to capture network's structural information at different granularities.

GraphAttention (Abu-El-Haija et al. [1]) proposes an attention model that learns a multi-scale representation which best predicts links in the original graph. Instead of pre-determining hyper-parameters to control the context nodes distribution, GraphAttention (Abu-El-Haija et al. [1]) automatically learns the attention over the power series of the graph transition matrix. **SDNE** (Wang et al. [16]) learns node representations that preserve the proximity between 2−hop neighbors with a deep autoencoder. It further preserves the proximity between adjacent nodes by minimizing the Euclidean distance between their representations. **Metapath2vec** (Dong et al. [5]) uses meta-path-based walks which capture the relationship between different types of nodes. For learning representation from random walk sequences, they propose heterogeneous Skip-gram which considers node type information during model optimization.

GCN (Kipf and Welling [7]) is a novel approach for semi-supervised classification on graph-structured data, which uses an efficient layer-wise propagation rule that is based on a first-order approximation of spectral convolutions on graphs. Experiments have proven that GCN (Kipf and Welling [7]) is capable of encoding both graph structure and node features in a way useful for semi-supervised classification, and it can model variable-sized sub-graph structures in homogeneous graphs and preserve different scales of structual similarity. **GAT** (Veličković et al. [14]) is also a convolution-style neural network that operates on graph-structured data, leveraging masked self-attention layers. The graph attention layer utilized throughout these networks is computationally efficient, allows for (implicitly) assigning different importance to different nodes within a neighborhood while dealing with different sized neighborhoods, and does not depend on knowing the entire graph structure upfront, thus addressing many of the theoretical issues with previous spectral-based approaches. **DGI** (Veličković et al. [15]) learns unsupervised representations on graph-structured data by leveraging local mutual information(Jensen-Shannon mutual information formula) maximization across the graph's patch representations, obtained by powerful graph convolutional architectures, and can obtain node embeddings that are mindful of the global structual properties of the graph while some local noise can be filtered out. **Graph Infoclust (GIC)** (Mavromatis and Karypis [8]) is based on DGI (Veličković et al. [15]). In addition to maximizing the mutual information of node representation and global representation, it also maximizes the mutual information of node and its corresponding cluster representation, that is, clustering all nodes, and then maximizing the mutual information of each node and its cluster center. This enables graph embedding to better grasp various structural properties and avoid falling into the misunderstanding of optimization based on single vector.

3 PROBLEM DEFINITION

In this project, We collected 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. The collected information is used to form an academic network, and there are two feasible ways:

1 Form a homogeneous network where each node represents an author, and each directed edge represents the citation relation or co-authorship between the two connected authors. This will need some extra data processing to convert the relation between papers to the relation between authors.
2 Build a heterogeneous network, which contains two types of nodes, one type of nodes represents authors, and the other represents papers. In this network, each edge between an author node

and a paper node means the authorship of the paper, and each directed edge between two paper nodes represents the citation relation.

We can choose either way to form an academic network. For node classification, 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. In this project, you are given the whole information needed to form a network and label information of 20% nodes (both author nodes and paper nodes). Our task is to predict the labels of the rest 80% author nodes.

For link prediction, noted that what we provide is the author and paper information from 2016 to 2019, and some authors will have new works published in 2020. Consequently, there will be new cooperation and citation relationships between authors, and new edges will be connected between corresponding author nodes in the network. Our task is to predict each pair of author nodes in the test set based on the information provided. If there will be an edge between the two given nodes in 2020, mark it as 1, otherwise mark it as 0.

4 DATA PROCESSING

In this section, we're going to introduce data pre-processing, the data deficits we found and the ways we try to deal with them.

4.1 Homogeneous Construction

According to the downstream application, we want to predict the labels of authors and the relation between authors. That means if we want to construct a homogeneous network of this problem, the nodes type of the network must be author. However, as we have mentioned in Section 2, we already have relations between authors and papers, i.e. authorship, and relations between papers and papers, i.e. paper citation. So we need to process the original relations to the relations we need.

Given the author-paper authorship relation, for each paper, we can get its authors. Then the relation among the authors of the same paper is **coauthor**.

Then given the paper-paper citation relation, using each paper's authors we get before, we can construct the **author citation** relations easily.

4.2 Heterogeneous Construction

There is no need or limitations for relation processing in heterogeneous network, since both author nodes and paper nodes are included in such network and authorship together with paper citation is enough to describe the whole network.

But it's worth noting that, since both the author ids and paper ids from the original data start from '0', when the embedding model gets a node id, for example '0', it cannot judge whether it's an author or a paper.

To avoid the confusion of node ids in heterogeneous network, we add **prefix** 'a-' to each author id and 'p-' to each paper id.

4.3 Data Deficits

Note that, downstream application in our main idea is based on node embedding which means such a node must be in the network to get its embedding. However, in the process of trying to construct a homogeneous network, during prediction, we found that we didn't get the corresponding embedding of some node either in train set or test set. We infer such node is not in the network, which means that such author is the single author of his paper and his paper has no citation relation with others.

To verify our inference, we design experiment to count the author ids in coauthor relation together with author citation relation. And the result shows that for total 42,614 author ids, there are 561 author ids not in the co-citation relation which are **isolated nodes**.

And such data deficits(isolated nodes), whose embeddings cannot be calculated directly or accurately, make trouble in prediction task.

4.4 Deficits Solution

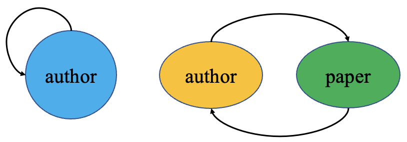


Figure 2: Add self-loops or construct heterogeneous graph to deal with isolated nodes

To deal with the problems of isolated nodes, we try following ways:

Random Predict: If nodes are not in the network, we cannot get their embeddings. The easiest way is when we predict the result, if we detect the node in the test set is isolated, then we give the author node a random prediction result. This is our primary idea, however, such way is so random that the performance is not well.

Add Self-loop: Isolated nodes cannot be put in graph, since they have no relations with others. But they can have relation with themselves. To put such nodes in a homogeneous graph, we add self-loops to them .

Heterogeneous Construction: The most natural way is to construct the heterogeneous graph, since every author has authorship with his paper, there is no apparently isolated node.

5 METHODOLOGY

5.1 New Tricks

However, even if we use self-loops or heterogeneous construction to put isolated nodes in graph, the embeddings of such nodes we obtain are still inaccurate, since in theory, in random-walk based embedding algorithm, such author node just walks to itself or its own paper, so actually it's still "isolated", which means that we need to improve the correlation between nodes further.

5.1.1 Node-Feature Embedding. We find that in original data, except relations and labels, there is some other information, for example, paper publish year. It's natural to think that papers published in the same year should have more correlation. To improve the correlation between nodes, we combine paper nodes with their features together to get the embeddings.

5.1.2 Papers Labels To Author Labels. For node classification task, we need to predict multi labels of authors. The direct idea is to predict the author labels based on author embeddings by multi-label classifier. But in common sense, single label classification is easier and more accurate. And the paper label is single because one paper can only be published in one conference. So we first classify the single label of papers, then according to the authorship, we can find the papers one author write, and then combine the labels of such papers as the multi labels of the author.

5.2 Embedding

For homogeneous network, we've researched and implemented DeepWalk, node2vec, SDNE and Struct2vec with the best performance in node2vec. For heterogeneous network, we've researched and implemented metapath2vec. Here, we mainly introduce node2vec and metapath2vec whose feature learnings are both based on Skip-Gram and whose node neighborhood findings are both based on random walk.

5.2.1 Node2vec. Let $G = (V, E)$ be a given network. Let $f : V \rightarrow \mathbb{R}^d$ be the mapping function from nodes to feature representations we aim to learn for a downstream prediction task. d is a parameter specifying the number of dimensions of our feature representation. For every source node $u \in V$, we define $N_S(u) \subset V$ as a network neighborhood of node u generated through a neighborhood sampling strategy S.

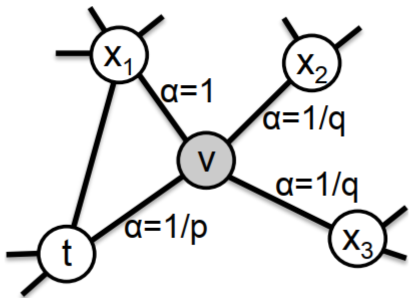


Figure 3: Illustration of the random walk procedure in node2vec. The walk just transitioned from t to v and is now evaluating its next step out of node v. Edge labels indicate search biases α

Extending the Skip-gram architecture to networks, it seeks to optimize the following objective function, which maximizes the log-probability of observing a network neighborhood $N_S(u)$ for a node u conditioned on its feature representation, given by f:

$$\max_f \sum_{u \in V} \log \Pr\left(N_S(u) \mid f(u)\right) \tag{1}$$

With assuming conditional independence and symmetry in feature space, the objective simplifies to :

$$\max_f \sum_{u \in V}\left[-\log Z_u + \sum_{n_i \in N_S(u)} f(n_i) \cdot f(u)\right] \tag{2}$$

node2vec is a flexible neighborhood sampling strategy which can smoothly interpolate between BFS and DFS with a flexible biased random walk procedure that can explore neighborhoods in a BFS as well as DFS fashion.

Return parameter, p. Parameter p controls the likelihood of immediately revisiting a node in the walk. **In-out Parameter,** q. Parameter q allows the search to differentiate between "inward" and "outward" nodes.

5.2.2 Metapath2vec. To model the heterogeneous neighborhood of a node, *metapath2vec* introduces the heterogeneous skip-gram model. To incorporate the heterogeneous netork structures into skip-gram, propose meta-path-based random walks in heterogeneous networks.

Heterogeneous Skip-Gram: In *metapath2vec*, we enable skip-gram to learn effective node representations for a heterogeneous network $G = (V, E, T)$ with $|T_V| > 1$ by maximizing the probability of having the heterogeneous context $N_t(v), t \in T_V$ given a node v:

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

where $N_t(v)$ denotes $v's$ neighborhood with the t^{th} type of nodes and $p(c_t|v;\theta) = \frac{e^{X_{c_t} \cdot X_v}}{\sum_{u \in V} e^{X_u \cdot X_v}}$, where X_v is the v^{th} row of X, representing the embedding vector for node v.

Meta-Path-Based Random Walks: How to effectively transform the structure of a network into skip-gram? In DeepWalk(Perozzi et al. [9]) and node2vec(Grover and Leskovec [6]) , this is achieved by incorporating the node paths traversed by random walkers over a network into the neighborhood function.

Naturally, we can put random walkers in a heterogeneous network to generate paths of multiple types of nodes. At step i, the transition probability $p(v^{i+1}|v^i)$ is denoted as the normalized probability distributed over the neighbors of v^i by ignoring their node types. The generated paths can be then used as the input of node2vec and DeepWalk. However, Sun et al. demonstrated that heterogeneous random walks are biased to highly visible types of nodes—those with a dominant number of paths—and concentrated nodes—those with a governing percentage of paths pointing to a small set of nodes.

In light of these issues, we design meta-path-based random walks to generate paths that are able to capture both the semantic and structural correlations between different types of nodes, facilitating the transformation of heterogeneous network structures into metapath2vec's skip-gram.

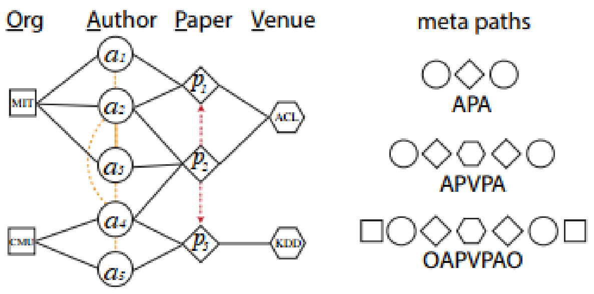
Formally, a meta-path scheme P is defined as a path that is denoted in the form of $V_1 \xrightarrow{R_1} V_2 \xrightarrow{R_2} \cdots V_t \xrightarrow{R_t} V_{t+1} \cdots \xrightarrow{R_{l-1}} V_l$, wherein $R = R_1 \circ R_2 \circ \cdots \circ R_{l-1}$ defines the composite relations between node types V_1 and V_l. Figure 4(a) is an example.

5.3 Link Prediction

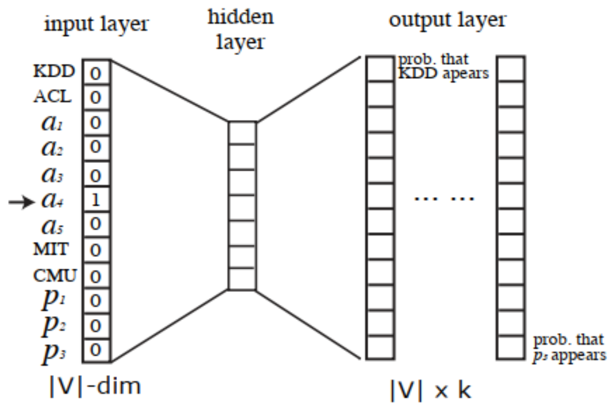
Link prediction is based on author embeddings. According to the principle of node2vec and metapath2vec, for a pair of author nodes, the more similar their embeddings are, the more similar their neighborhood will be, which means that there will be more probability on the relation existence of such pair of author nodes.

So we use similarity of two embeddings to reflect the probability of edge between two nodes. Here, we propose to use **cosine similarity**:

$$\cos\text{ sim } = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}} \tag{4}$$

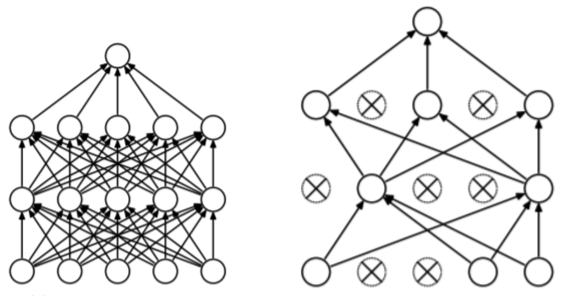


(a) An academic network



(b) Motion Optimization

Figure 4: An example of a heterogeneous academic network and skip-gram architectures of metapath2vec for embedding this network. (a). Yellow dotted lines denote coauthor relationships and red dotted lines denote citation relationships. (b) The skip-gram architecture used in metapath2vec when predicting for a_4, which is the same with the one in node2vec if node types are ignored. $|V|$=12 denotes the number of nodes in the heterogeneous academic network in (a) and a_4's neighborhood is set to include CMU, a_2, a_3, a_5, p_2, p_3, ACL, & KDD, making k = 8.



(a) Standard Neural Net (b) After Applying Dropout

Figure 5: Dropout Neural Net Model. (a): A standard neural net with 2 hidden layers. (b):An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

We also use tanimoto coefficient to estimate the probability as comparison:

$$E_j(A, B) = \frac{A * B}{||A||^2 + ||B||^2 - A * B} \tag{5}$$

5.4 Node Classification

5.4.1 scikit-learn. We first try traditional multi-label classification using sklearn-multilabel.

5.4.2 self-designed deep model. We propose a deep learning model designed by ourselves using fully connected layers with dropout and 1D convolution layers. (See in Figure 5)

6 EXPERIMENTS

In this section, we do abundant experiments for the following purposes:

- To show the performance of our data processing and tricks
- To visualize the feature representations we obtained
- To demonstrate the performance of different embedding models and make comparisons
- Try different parameter settings to get better result
- To show the performance of different downstream models and make comparisons

6.1 Experimental Setup

Our experiments visualize the feature representations obtained by several embedding models with our data processing & tricks and evaluate the feature representations on standard supervised learning tasks: multi-label classification for nodes and link prediction for edges. For both tasks, we mainly use node2vec and metapath2vec as embedding models.

Based on node2vec, the parameters we changed are return parameter p, in-out parameter q, walk length and number of walks. Based on metapath2vec, the parameters we changed are walk length, number of walks and vector size. Besides, for metapath2vec, our path type is as following:

- author-paper-author
- author-paper-paper-author
- paper-paper
- author-author

We try lots of parameter settings to get better results.

Data: We use the data as we mentioned in Section 3. In homogeneous graph, we use 42,614 authors as nodes and in heterogeneous graph we use 42,614 author nodes and 24,251 paper nodes. We use authorship, paper citation, coauthor relation and author citation relationships as edges. Note that, in multi-label classification, our performance is based on 20% data labels for training and 80% data for test.

Evaluation: According to the requirement of kaggle competition, our evaluation standard of multi-label classification 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 positive tp to all predicted positives $tp + fp$. Recall is the ratio of true positives to all actual positives $tp + fn$. The F1 score is given by:

$$F1 = 2\frac{p \cdot r}{p + r} \quad \text{where} \quad p = \frac{tp}{tp + fp}, \quad r = \frac{tp}{tp + fn} \tag{6}$$

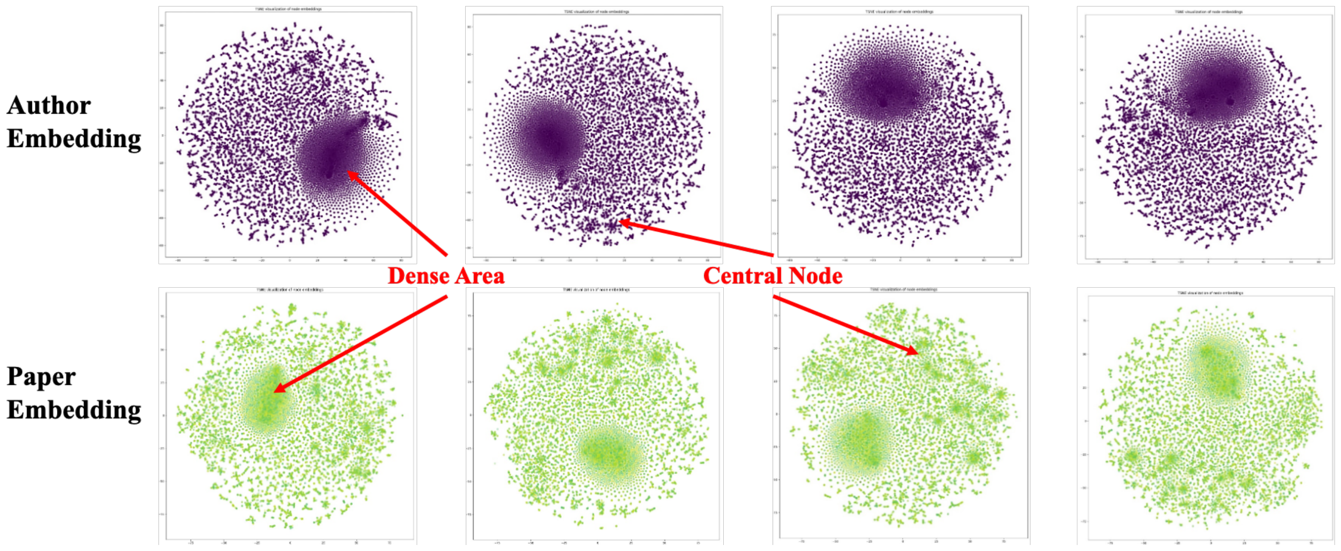


Figure 6: TSNE Dimension Reduction of Embedding Visualization

As for link prediction task, our evaluation metric is AUC. The area under the Receiver Operating Characteristic (ROC) curve, measures accuracy using the statistics True Positive Ratio (TPR) and False Positive Ratio (FPR). TPR is the ratio of true positives (tp) to all samples (tp+fp+tn+fn). FPR is the ratio of false positives to all samples. The AUC is given by:

$$AUC(f) = \frac{\sum_{t_0 \in D^0} \sum_{t_1 \in D^1} 1[f(t_0) < f(t_1)}{|D^0| \cdot |D^1|} \tag{7}$$

where $1|f(t_0) < f(t_1)|$ denotes an indicator function which returns 1 iff $f(t_0) < f(t_1)$ otherwise return 0; D^0 is the set of negative examples, and D^1 is the set of positive examples.

6.2 Embedding Visualization

The first step of our pipeline is to obtain the node embeddings. Rather than evaluating feature representing in downstream, we can visualize the embedding to evaluate roughly but directly from our human perception.

But due to the data complexity, the vector size we use for embedding is 64, 128 and 150 which is impossible to describe in 2D or 3D images. So here we use **TSNE** to reduce the dimension to 2D.

For different embedding model parameter settings, we've got 2D visualization of author embeddings and paper embeddings as shown in Figure 6.

6.3 Link Prediction

In link prediction, we are given a network with certain fraction of edges "removed", and we would like to predict these "missing" edges.

We mainly use node2vec and metapath2vec with different parameter settings to get embedding and during the experiment we use our methods to deal with isolated nodes and add tricks to models.

Since we use similarity of embeddings to demonstrate the probability of edges, we try both cosine similarity and Jaccard coefficient as comparison metrics.

Experimental Result: We summarize our results for link prediction in Table 1 and Table 2:

node2vec	1	2	3	4
walk length	80	100	120	120
# of walks	10	10	10	10
p	0.25	0.25	0.2	0.2
q	4	4	5	5
AUC	0.68375	0.68017	**0.69567**	0.68826

Table 1: Link prediction based on node2vec embedding

metapath2vec	1	2	3	4	5(featured)
walk length	80	100	120	120	120
# of walks	10	10	10	10	20
vector size	128	128	128	128	128
AUC	0.69217	0.69758	0.70634	0.71524	**0.72400**

Table 2: Link prediction based on metapath2vec embedding

6.4 Multi-label Classification

In the multi-label classification setting, every node is assigned one or more labels from a finite set $\mathcal{L}$. During the training phase, we observe a certain fraction of nodes and all their labels. The task is to predict the labels for the remaining nodes. This is a challenging task especially if $\mathcal{L}$ is large.

We first use several default similar parameter settings to evaluate different embedding models. After that we mainly use metapath2vec model with different parameter settings to get the embeddings and for downstream classification, we use traditional multi-label classifier and deep learning model we designed as comparison.

Experimental Result: For various embedding models, we select DeepWalk, node2vec, SDNE, Struct2vec and metapath2vec. Giving them several similar parameter settings, we evaluate their

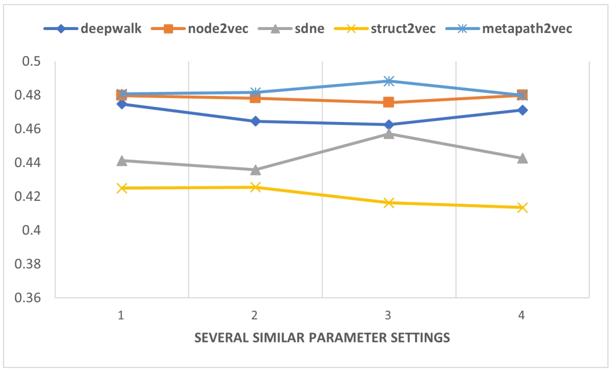


Figure 7: Performance of various embedding models on several similar parameter settings

performance roughly(see figure 7). And we find that in general, the metapath2vec model outperforms other methods.

So then we mainly use metapath2vec to get the embeddings on different parameter settings and evaluate the embeddings with traditional classifier and our designed deep model as comparisons. Our results are shown in Table 3 and Table 4.

sklearn	1	2	3	4	5
walk length	80	120	120	120	100
# of walk	10	10	20	20	10
vector size	128	128	128	128	128
threshold	0.2	0.2	0.08	0.175	0.2
F1-score	0.48065	0.48486	0.47838	**0.48894**	0.48084

Table 3: Traditional classifier performance on multi-label classification

deep model	1	2	3	4
walk length	100	120	120	120
# of walk	1	10	10	20
vector size	128	128	128	128
threshold	0.08	0.1	0.2	0.1
F1-score	0.48084	**0.49471**	0.48779	0.49025

Table 4: Deep learning model performance on multi-label classification

6.5 Ensemble Learning

Based on our main idea and experiment, we also do ensemble learning to improve our final result in Kaggle:

Model Fusion: After we get the features, we put nodes together with features into **DGI**, another model we refer to, to do ensemble learning and get good results in link prediction.(See in Table 5)

Results Voting: Collecting several embedding results which perform relatively well, we design vote algorithms for both link prediction and node classification tasks and based on the results we obtained, we successfully make improvements.(See in Table 5 and Table 6)

Ensemble	1	2	3	4
link prediction AUC	0.75612	0.75530	0.76512	**0.77322**

Table 5: Ensemble learning performance in link prediction

Ensemble	1	2	3	4
node classification F1-score	0.49471	0.49791	0.50219	**0.50558**

Table 6: Ensemble learning performance in node classification

7 CONCLUSION

Our Contribution: In this work, we implement both homogeneous and heterogeneous models to get embeddings for outline methods which make abundant comparisons between such two kinds of models. The core work is that we find the data deficits: isolated nodes and design several methods together with data correlation strengthening tricks to deal with the deficits. We also claim and verify clearly the methods and models we use for downstream application in detail, so we get an end-to-end system for our project. At last, we do abundant experiments for our work to evaluate our embedding models, our tricks and our downstream models.

Experimental Analysis & Conclusion: In our experiment, we get the following analysis and conclusion:

Embedding makes a difference:

- Different parameter settings result in different results
- Generally, larger walk length and number of walks will lead to better results
- Models based on random walk have limitations

Downstream model makes a difference:

- Deep model in classification performs better than traditional methods in sklearn

Certain data needs certain tricks:

- The try and tricks we use to deal with the isolated nodes are the core of our project

Final Performance: In this work, for both link prediction and node classification, we get great results in both public and private leaderboards(See in Table 7):

Leaderboard	Public Leaderboard	Private Leaderboard
link prediction	0.77322	**0.77558**
AUC&rank	#3	**#3**
node classification	0.50558	**0.50280**
F1-score&rank	#8	**#6**

Table 7: Final results in leaderboards

Future Work: Throughout the experiments, after discussing with other teams, we find the limitation of embedding models based on random walk. So we would like to try more embedding models such as GCN, GAT, etc.

8 TEAM CONTRIBUTION

We have reasonable division for this project:

Yifei Li:

- Research & Proposal
- Data Processing & Graph Embedding
- Report & Presentation

Haoning Wu:

- Research & Proposal
- Link Prediction
- Report & PPT

Longrun Zhi:

- Research & Proposal
- Node Classification
- Report & PPT

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alex Alemi. 2017. Watch your step: Learning graph embeddings through attention. *arXiv preprint arXiv:1710.09599* 1, 2 (2017), 3.

[2] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1616–1637.

[3] Haochen Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2018. A tutorial on network embeddings. *arXiv preprint arXiv:1808.02590* (2018).

[4] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2018. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering* 31, 5 (2018), 833–852.

[5] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 135–144.

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

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

[8] Costas Mavromatis and George Karypis. 2020. Graph InfoClust: Leveraging cluster-level node information for unsupervised graph representation learning. *arXiv preprint arXiv:2009.06946* (2020).

[9] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[10] Bryan Perozzi, Vivek Kulkarni, and Steven Skiena. 2016. Walklets: Multiscale graph embeddings for interpretable network classification. *arXiv preprint arXiv:1605.02115* (2016), 043238–23.

[11] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 385–394.

[12] Xin Rong. 2014. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738* (2014).

[13] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. 1067–1077.

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

[15] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rklz9iAcKQ

[16] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 1225–1234.

[17] Suhang Wang, Jiliang Tang, Charu Aggarwal, Yi Chang, and Huan Liu. 2017. Signed network embedding in social media. In *Proceedings of the 2017 SIAM international conference on data mining*. SIAM, 327–335.