Node Representation and Link Prediction in Multi-Edge Networks

Peeyush Agarwal
peeyush@stanford.edu

Roman Polonsky
romandra@stanford.edu

Zihan Xie
xiezihan@stanford.edu

1 Abstract

Node classification and link prediction are classic problems in networks with many applications including e-commerce and social network recommendation systems. Many graphs, due to its complexity and large size, are flattened down to a simple graph before being analyzed. While this saves computation power and yields many good results, networks in real world have heterogeneous interactions with interdependent structures, which are better modeled as multigraphs.

In the project, we first review state-of-the-art algorithms for the tasks of node embedding, classification and link prediction in multigraphs. We then analyze GraphSAGE in more detail and extend it to leverage the multiple edge types and additional node attributes available in our constructed heterogeneous graph for better performance in the tasks of node classification and link prediction.

2. Problem Definition

Our project is targeting multimodal network node classification and link prediction. There are a number of recently developed methods to analyze real-world systems as heterogeneous networks. These networks are often multi partite, have different types of nodes and edges (along with metadata etc.) and might have temporal aspects associated with them. We apply the ideas from recent algorithms like GraphSAGE to research collaboration dataset to learn and evaluate node embeddings, perform multi relational link predictions, and evaluate how different representation and edge dimensions of heterogeneous networks influences the prediction performance.

3. Background

In this section, we review the existing literature for multi-graphs, node embeddings, node classification and link prediction.

3.1 Multi-relational Link Prediction in Heterogeneous Information Networks [1]

To predict the heterogenous relationships, Davis, Lichtenwalter and Chawla first apply variation of unsupervised common neighbors method called Adamic/Adar measure. Then authors introduce new multi-relational link prediction (MRLP) method which applies a weighting scheme for different edge type combinations. The weights are determined by counting the occurrence of each unique 3-node substructure in the network, called triad census or 3-node graphlets. The 3-node graphlets in heterogeneous networks have higher number of unique structures. They also provide the probability of each structure as shown in their Figure 1.

While new MRLP method works best for some interaction types, authors observed that there is no universally dominant method. Authors explore supervised learning approach by converting the heterogeneous network into a feature representation. They extend high performance link prediction (HPLP) approach to multi-relational case (MR-HPLP). The topological features represent potential link patterns between the node pair and their common neighbors. Experiments on our real-world networks indicate that the supervised method is dominant, outperforming the best unsupervised methods in all cases but one.

3.2 Multilayer Network [2]

Multilayer Networks is a survey paper that tries to unify existing disparate terminologies that deal with complex graphs by presenting and discussing a general multilayer network model. The paper 1) gives one definition for general multi-layer network, 2) discusses how various existing networks in literatures can be mapped to this general definition, 3) presents tensor representation & supra-adjacency representation, representing multi-layer networks as a big adjacency matrix of sorts, 4) shows some real-world datasets that are applicable to this model, and 5) discusses some models in dealing with multi-layer network including network aggregation and diagnostics for multilayer networks.

The paper presents the general definition of multilayer network as following. Define the term "aspect" to be a dimension of the type of an edge, and assume the graph has *d* aspects. Define Li to be the set of elementary layers for aspect i, then for the entire graph, we would have $L = \{L_1, L_2, \dots L_d\}$ as the set of layers. $V_m \subseteq V \times L_1 \times L_2 \times \dots \times L_d$ then defines the set of vertices, and $E_m \subseteq V_m \times V_m$ is the set of edges. Multilayer graph M is then defined as $\{V_m, E_m, V, L\}$.

3.3 Node2Vec: Scalable Feature Learning for Networks [3]

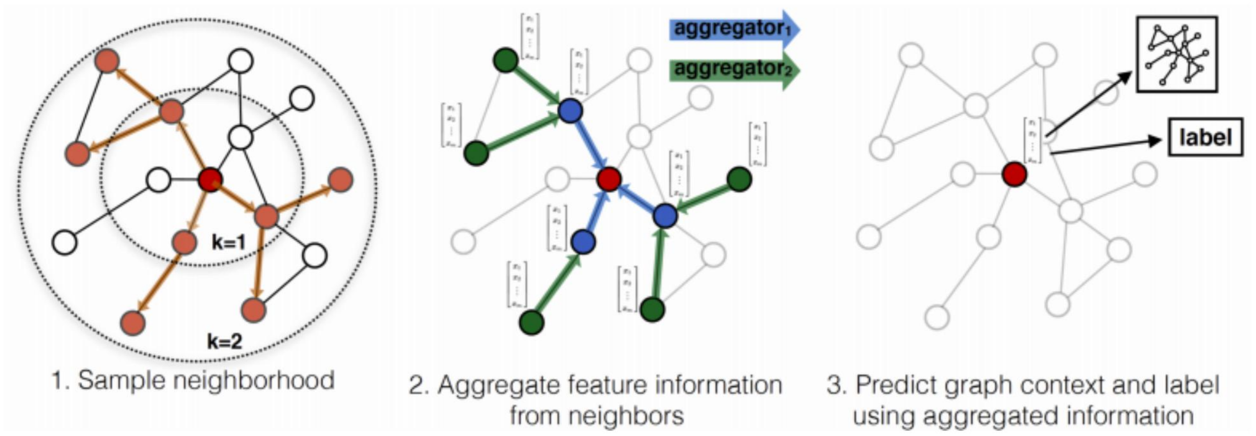
Node2Vec describes a framework of learning node embeddings/representations by doing a random walk over the graph. The objective that node2vec tries to optimize is that it tries to preserve the notion of node neighborhoods in the embeddings, i.e., nodes closer together in the network are also expected to be closer together in the learnt embeddings space. This is done by doing a “random walk” over the graph, thus linearizing the graph. Then, the probability of node neighborhood is optimized using SGD similar to the skip-gram model architecture.

The biased random walk process is described to be more general than the random walk process described in previous works like DeepWalk (uniform random walk), LINE etc. The added flexibility from this newly proposed random walk is shown to improve the performance on classification and prediction tasks. The random walk described in the paper combines BFS and DFS strategies in an elegant way.

The approach is evaluated for the task of multi-label classification. The classification network is a simple one-vs-rest logistic regression classifier with L2 regularization. The algorithm is able to achieve state-of-the-art performance compared to previously proposed node embedding approaches.

3.4 GraphSAGE: Inductive Representation Learning on Large Graphs [4][5]

GraphSAGE is a framework for learning low dimensional vector embeddings for nodes. The framework builds upon the recently proposed graph convolutional network (GCN) based approaches and makes it more scalable and general. More specifically, the proposed approach is inductive (thus not constrained to only nodes seen during training) and leveraged node features and metadata for learning the node embedding.



1. Sample neighborhood 2. Aggregate feature information from neighbors 3. Predict graph context and label using aggregated information

As shown in the figure above (from [4]), GraphSAGE proposes a SAmble and aggreGatE based strategy.

- The authors do a uniform sampling of fixed size set of neighbors in order to keep the single batch running time deterministic and predictable.
- The aggregation function is responsible for aggregating the information from the node neighborhood. The aggregation function is learnt as a part of model training by doing SGD over positive (nearby nodes) as well as negative examples (far-away nodes). Authors proposed and benchmarked various aggregation functions like mean, LSTM, pooling etc. and found LSTM to work best among the tried approaches.
- The embedding function is learnt over the node features (like image representation from CNN, text vector, node features etc.) which helps generalize to previously unseen data. The authors propose updating the embedding K times (with the node features as input the first time). This helps capture the neighborhood notion upto depth K.

For training the model parameters, the authors have proposed supervised as well as unsupervised loss functions.

The algorithm is evaluated on

- Predicting post category on Reddit post data where the predictions are made on last month of data (not used during the training).
- Predicting paper category on citation data.
- Predicting protein function on protein-protein interaction data (supervised).

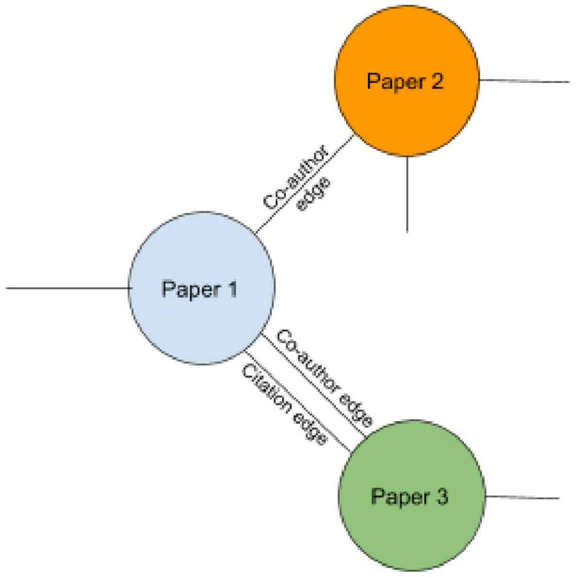
4. Technical Approach and Findings

4.1 Data Set

As our primary dataset, we used research collaboration datasets. The research paper network (https://aminer.org/citation) [7] is a rich citation network that contains information like paper title, authors, paper venue, published year, citation papers and abstract. This makes it a very interesting network to analyze. The nodes can be research papers and edges can be papers having the same authors, being cited by the same papers, appearing in the same publication, etc. Our citation dataset contains a few metadata about each academic paper, including title, author, publication venue, publication year, list of references, and abstract.

4.1.1 Data format

```
#* --- paperTitle
#@ --- Authors
#t ---- Year
#c  --- publication venue
#index 00---- index id of this paper
#% ---- the id of references of this paper
   (multiple lines, with each indicating a reference)
#! --- Abstract
```



The figure above describes the Research Collaboration multi-graph. Here, the nodes are research papers. The nodes have abstract and paper title as node features and year of publication as node attribute (for node-classification task). The nodes are connected if they share a co-author or citation, therefore, creating 2 types of edges in the graph.

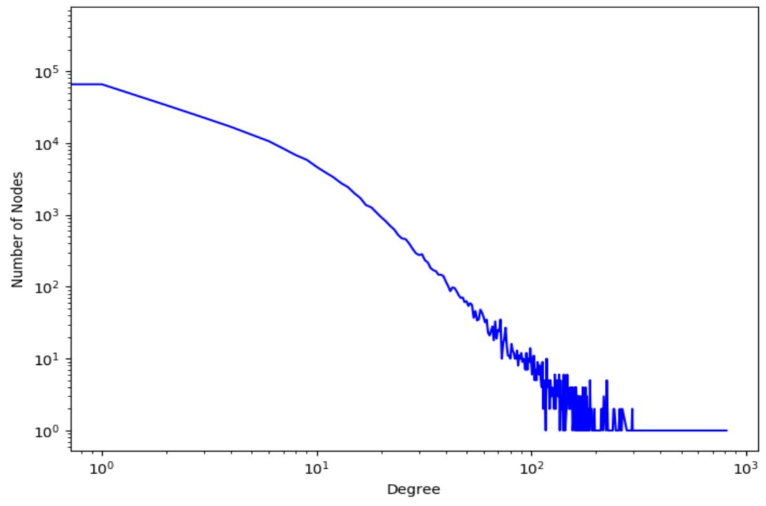
4.1.2 Graph Statistics

Basic Statistics

- Number of nodes: 629814

- Number of citation relationships: 631120 (edge density: 3e-6)
- Number of co-author relationships: 5306414 (edge density: 2.6e-5)

Degree Distribution



As can be seen from the statistics above, the citation edges are very sparse (number of citation edges ~= number of nodes in the network). The co-author edges are denser and more typical of a real world network.

4.2 Algorithm

As described in the section above, our research collaboration graph consists of research papers as nodes, citation and co-author as edges and paper title and abstract as node features. We use Gensim implementation of Word2Vec[8] and NLTK word tokenizer to featurize the paper title and abstract. For the word embeddings, we use Google's pre-trained 300 dimensional word embeddings trained on News dataset. The word embeddings from all words in the title or abstract are averages to generate the feature embeddings.

4.2.1 Simple Graph Learning

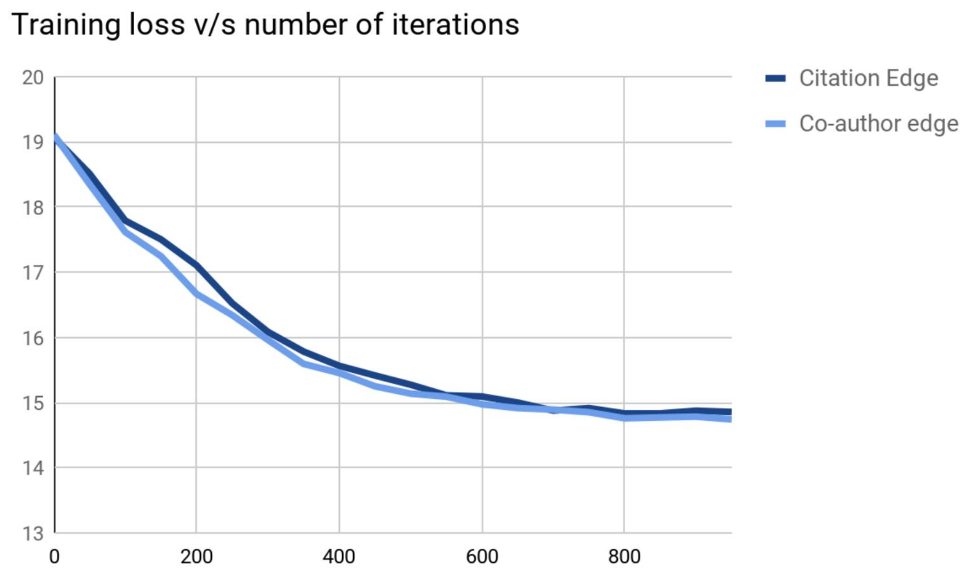
We first treat the graph as simple (consisting of only one kind of edge) and run the GraphSAGE unsupervised node embedding training code to train the network embeddings. We experiment with and tune various parameters while learning the node embeddings. The effect of each parameter on the final embedding is described below.

- Model: We try various aggregator models like GraphSAGE mean (mean based aggregator) and GraphSAGE seq (LSTM based aggregator), GraphSAGE maxpool (max-pooling aggregator). We observe that the mean is the fastest aggregator (as also expected intuitively) in terms of runtime and the performance is very comparable to the more complex aggregators. For this reason, we decided to stick with the mean aggregator for the rest of the experiments.
- Identity dimension: We observe that the performance is significantly better when setting identity dimension to a positive high value. This is also expected as the model can now use the node ID itself as an additional feature and hence the learnt embeddings are much better. Adding the identity dimension comes with the trade-off that we can no longer add new nodes during the

validation and testing phase but we did not plan to do that anyway. We decided to set the identity dimension to 64.

- Weight decay: This parameter controls the weight for L2 loss on embedding matrix. We set the weight decay to 0.5 in order to ensure that the model does not overfit to the graph.
- Embedding dimension: We find that the default parameter of 128 for each output dimension (and hence an embedding of size 256) works quite well for us.

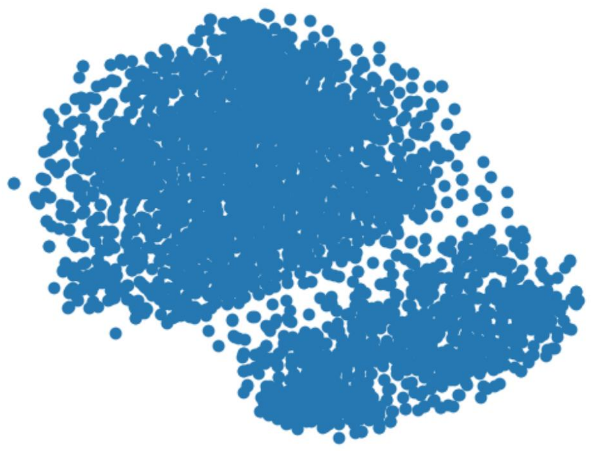
We also experimented with other parameters like negative sampling rate, dropout, random walk parameters etc. but did not find them to significantly impact the embedding quality. The biggest improvement to node quality was observed from adding node attributes (paper title and abstract). This is probably because the research collaboration graph is quite sparse and therefore “side-input” in the form of node features helps learn richer node embeddings.

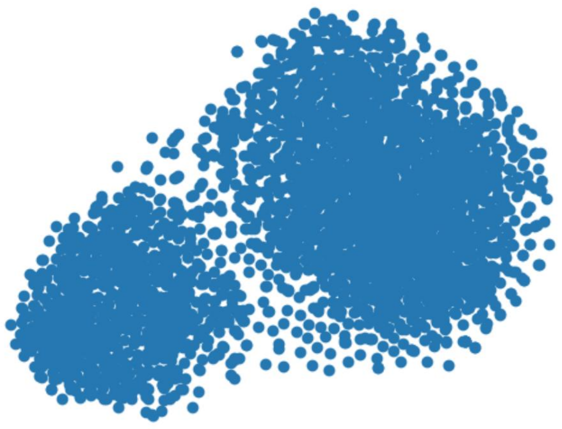


For both citation and co-author edge simple graphs, we saw that the train and validation loss starts around 20 (slight difference based on edge type) and converges around 15.

Collaboration graph with citation edges

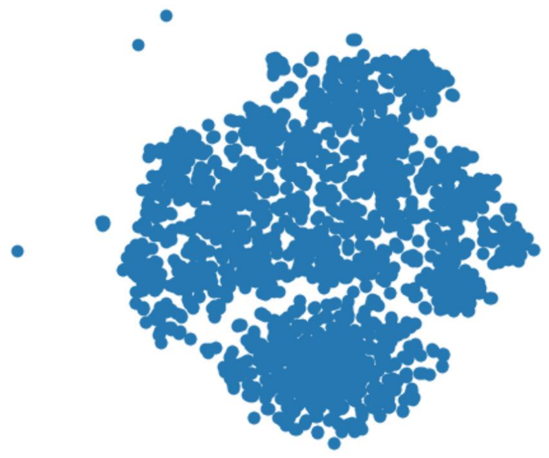
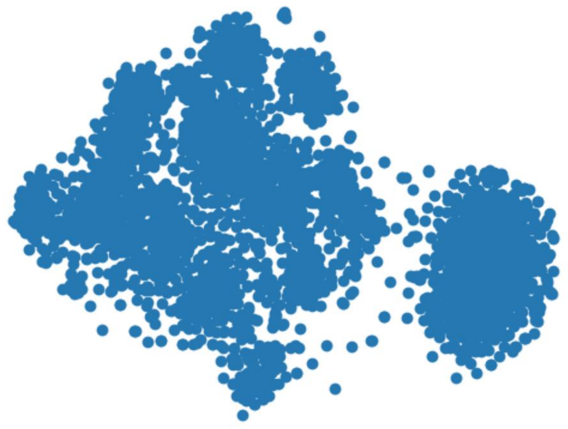
We applied t-SNE technique for dimensionality reduction to visualize our learnt node embeddings. Due to the large size of the graph, random 0.5% nodes were chosen (slightly more than 3K out of 630K). Trying different perplexity, below graphs are generated with perplexity 30 and 100 respectively:





Collaboration graph with co-author edges

Same t-SNE technique was applied to visualise the embeddings generated from co-author edges. This time, perplexity of 5 and 50 were used on respective graphs below.

4.2.2 Link Prediction Task on Multi Graphs

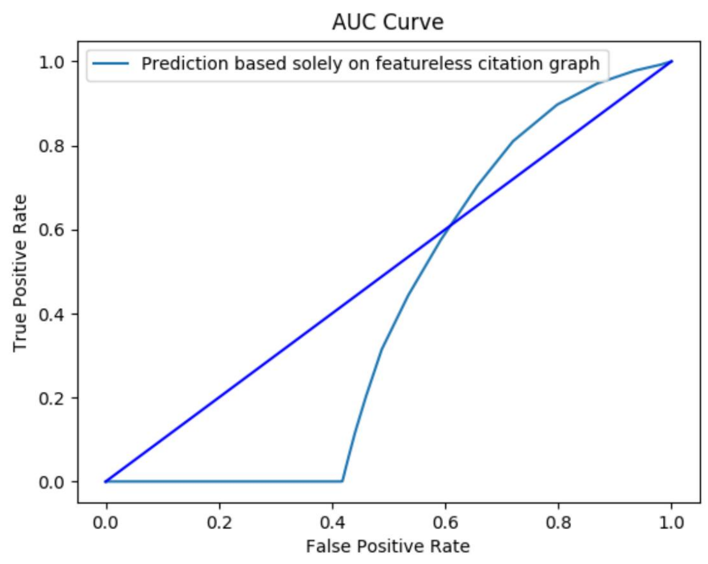
Evaluation Methodology

Given the set of node embeddings we generated from training, the goal of link prediction task is to predict whether a citation relationship exists between two nodes. In order to evaluate our node embeddings, we randomly omitted and added ~1% of edges into validationEdge set when parsing raw data into networkx graph. In addition, we randomly selected a set of validationNonEdge that are of the same size as validationEdge, and not present in the parsed networkx graph.

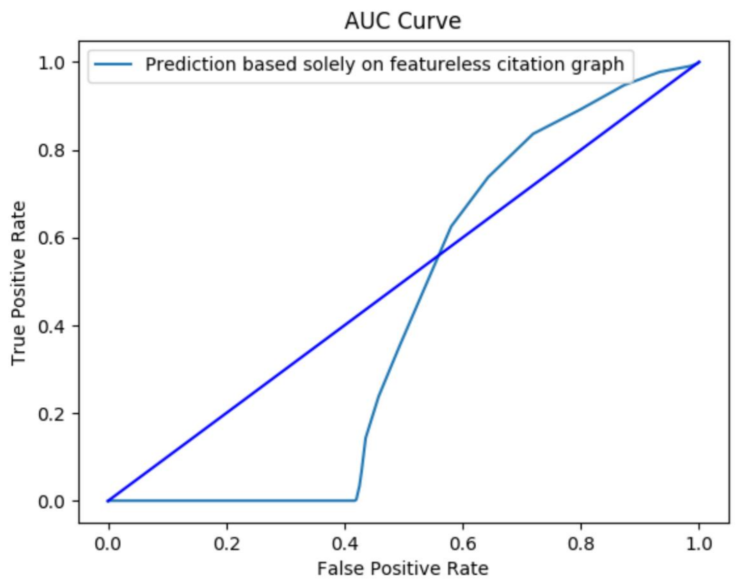
To make a prediction based on node embeddings, we used the dot product of the two nodes' embedding vectors, since dot product was used in the loss function of the training phase. To normalize, we randomly picked 1000 node pairs from the graph and compute the means and standard deviations of their dot products. Then for every edge in validationEdge set and validationNonEdge set, we computed the dot product, normalized it against the 1000 random samples to get a z-score, and used it as likelihood of link existence. For each set, we recorded the mean and area-under-the-curve (AUC) graph of all the z-scores of edge likelihood.

Baseline Result

As a baseline, we used embeddings trained from featureless citation network alone. We took care to select and remove validation edges only from nodes that have at least one other edge (thus not making the node disconnected from the rest of the network). The result was surprisingly unimpressive, almost to the point where the generated prediction can be interpreted inversely to produce better result. A likely explanation for this is that our graph is so sparse that each node only contains a very limited amount of information. With validation edge removed from training network and no information otherwise available, our model assumes the two nodes are very unlike each other.



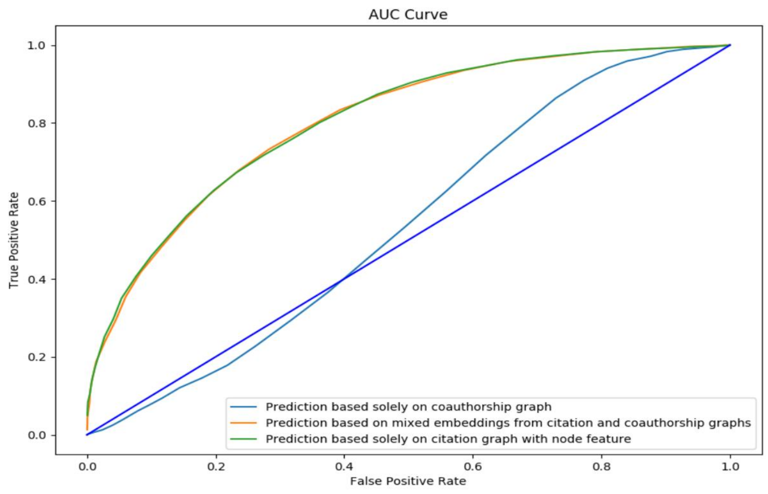
An unsuccessful attempt to improve on this naive model was to pick validation edges only from nodes that are reasonably well connected to the rest of the network. For example, below is the AUC graph of validation edges taken only from nodes with degree >= 5.



This curve only improves ever so slightly on the previous one, showing that when only considering citation network itself, very limited information can be learnt for most of the nodes in the network.

Evaluation Result

With embeddings learned from coauthor relationships and paper abstract and titles, we tried several ways to combine different signals and output a single prediction for citation relationships. They performed significantly better than embeddings trained on simple citation graph. Below we have plotted the AUC curves for 1) predictions made only using co-authorship graph embeddings, 2) predictions made only using citation graph embeddings with paper abstract as node feature, and 3) predictions combining embeddings from 1 and 2.



Interpretation

As shown by the graph, exploring additional information in our dataset and train embeddings from heterogeneous sources proved to be essential in link prediction of the very sparse citation network. With number of citation almost as small as the number of nodes, most of the nodes carry very little information when we look at them in a simple citation graph. Link prediction made by simple citation graph GCN downright failed. With co-authorship information added, the prediction task performance got slightly better than random, but adding in additional node features really helped learning embeddings of otherwise unknown nodes, and we were able to make link existence predictions with good AUC scores on such sparse graphs, where majority of training edges come from nodes that have degrees less than 5.

4.2.3 Node classification task on Multi Graphs

In this section, we explore how multi edges in graph can be leveraged to improve model performance on node classification task. For this, we try to predict the year of publication of the paper given the node features / learnt embeddings. This can be done by learning a linear regression over the embeddings feature vector. We evaluate the performance by comparing the coefficient of determination.

We evaluated the predictive power of node features for the task of predicting the year of publication and compared that against the predictive power of embeddings learnt from unsupervised citation and co-author networks. We find that the citation edges and co-author edges have relatively lower predictive power than node embeddings. However, concatenating the embeddings from the 2 kinds of edges leads to better performance compared to using node features. The performance can be further improved by supervised learning of node embeddings but it might not be possible to run supervised training in many real world scenarios.

Model	**Score***
Node features (paper title & abstract)	0.1596
Embedding from graph with citation edge	0.1465
Embedding from graph with co-author edge	0.1031

Combined embedding from citation and co-author edges	0.1692

*Score = (1 - u/v), where u is the residual sum of squares ((y_true - y_pred) ** 2).sum() and v is the total sum of squares ((y_true - y_true.mean()) ** 2).sum()*

5. Future work

There are several directions where this project can be explored more extensively.

From dataset's perspective, we could incorporate more dimensions as node features or relationships between nodes and see if they can improve our embeddings. One specific dimension of interest is the publication year of the paper, which can be explored as a temporal dimension and used to study how the network evolve over time. However, in our network the evolution happens not just with edges, but also nodes (as opposed to a network with a fixed set of nodes and an evolving set of edges), so it is interesting to explore which temporal algorithms can apply.

From the application's perspective, we could explore better ways to incorporate different types of edges into training node embeddings that are capable of wider applications. We scoped our project to utilizing all edge types and node features into predicting only citation links and year of publication, but using all heterogeneous information in the network we could potentially predict links of different types as well as various other node attributes (publication, author, etc), or perform node clustering tasks.

References

All code is located in https://github.com/apeeyush/cs224w-project

1. D. Davis, R. Lichtenwalter, and N.V. Chawla, "Multi-relational Link Prediction in Heterogeneous Information Networks" in *ASONAM '11 Proceedings of the 2011 International Conference on Advances in Social Networks Analysis and Mining*, https://ieeexplore.ieee.org/document/5992590
2. M. Kivelä, A. Arenas, M. Barthelemy, J.P. Gleeson, Y. Moreno, M.A. Porter, "Multilayer Networks" in *Journal of Complex Networks*, 1 Sept. 2014, pp. 203–271. https://doi.org/10.1093/comnet/cnu016
3. 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*, https://cs.stanford.edu/people/jure/pubs/node2vec-kdd16.pdf
4. W.L. Hamilton, R.Ying, J.Leskovec, "Inductive Representation Learning on Large Graphs" in *NIPS 2017*, https://www-cs-faculty.stanford.edu/people/jure/pubs/graphsage-nips17.pdf
5. J.Leskovec, "Large-scale Graph Representation Learning" video on https://www.youtube.com/watch?v=oQL4E1gK3VU&t=1132s
6. L. Maaten, G. Hinton, "Visualizing Data using t-SNE" in *Journal of Machine Learning Research 9 (2008)*, http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf
7. Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. ArnetMiner: Extraction and Mining of Academic Social Networks. In *Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (SIGKDD'2008). Pp.990-998.
8. T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean. "Distributed Representations of Words and Phrases and their Compositionality" in *Proceedings of NIPS, 2013*, https://arxiv.org/pdf/1310.4546.pdf