Better Together: Combining Textual and Graph Embeddings for Directed Edge Prediction in Citation Networks

Christian Clark
Georgia Institute of Technology
christianclark@gatech.edu

Abhinav Gupta
Georgia Institute of Technology
agupta931@gatech.edu

Shashank Srikanth
Georgia Institute of Technology
ssrikanth39@gatech.edu

Suryatej Reddy Vyalla
Georgia Institute of Technology
svyalla3@gatech.edu

ABSTRACT

Text data is part of many popular graph datasets; however, most existing graph embedding techniques do not utilize this text effectively for link prediction. In this work, we put forth a new approach called GAT-CAT (Graph And Text conCATenateded embeddings) that combines the sentence embedding and node embedding data to perform directed edge-prediction on citation networks. Through experiments on the High Energy Physics Theory citation network dataset, we show that our proposed model GAT-CAT achieves a ROC-AUC score of over 0.932 and an accuracy of over 0.877 on the dataset, a 3.10% increase in accuracy over the best-performing text-embedding method. Our code is available here: https://github.com/BonJovi1/6240-Better-Together

KEYWORDS

link prediction, social networks, graphs

ACM Reference Format:
Christian Clark, Abhinav Gupta, Shashank Srikanth, and Suryatej Reddy Vyalla. 2018. Better Together: Combining Textual and Graph Embeddings for Directed Edge Prediction in Citation Networks. In *Proceedings of (CSE 6240: Web Search and Text Mining).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Link predictions on citation networks have been studied extensively in the Graph Learning and Natural Language Processing (NLP) literature. Pure NLP approaches utilize a Siamese network to identify semantically similar publications, and graph based approaches utilize the local and global connectivity in the citation-network to make the predictions [14]. *Aim:* In our specific case, we plan to test a concatenation of these two types of embeddings on a citation network dataset that includes text data in the form of the paper metadata. Using this concatenated embedding, we will predict edges (citations) in the directed graph from one paper to another.

Textual embeddings allow users to capture the content and meaning of individual documents for use in machine learning. Graph embeddings capture the relationships among and between groups of data. *Challenges:* While these both have value individually, neither fully captures the information both within and and surrounding a given piece of data. In this work, we augment graph embeddings with language embeddings from natural language processing techniques through concatenation for enhanced neural network performance on edge prediction tasks.

Impact: Creating embeddings that improve edge prediction between nodes that can be described with text data would be valuable for a variety of tasks and subject areas. Similarly to a scientific citation network, the use of combined language and graph embedding techniques could be applied across legal and medical documents to discover court case precedence or related medical research. Textual embeddings from users' social media posts and their relationships within the social network (friendships, liking, and reposting) could allow for the improved predictions of future relationships or interactions with other users. Applications are even possible in the cybersecurity domain, where programs' textual log files could be used to predict links between users or users and programs.

2 LITERATURE SURVEY

Given a social network, we would like to predict whether nodes which are currently not connected would interact in the future. It has many applications such as movie recommendations and knowledge graph completion. There are three major ways of performing this task that have been proposed in the literature.

1) Heuristic based methods: A simple yet effective way for link prediction are heuristic methods. One popular method if the common neighbors [10] approach, which computes the number of common neighbors between two nodes as the similarity score of a link between the nodes. Nodes with higher scoring are more likely to have a link. For instance, in the citations dataset, paper A and paper B may have many citations in common (that is, both are connected to many common nodes) and this would result in a high heuristic score. Some heuristics involve the one-hop neighbors of two target nodes to calculate this score, such as common neighbors (CN) and preferential attachment (PA) [10]. Other popular methods such as Adamic-Adar (AA) [1] are second-order heuristics, as they are calculated from up to two-hop neighborhood of the target nodes. There are also some high-order heuristics which require knowing the entire network, most popularly the rooted PageRank algorithm [11] and SimRank (SR) [7]. However, the main shortcoming is that

CSE 6240: Web Search and Text Mining, Spring 2022, Georgia Institute of Technology

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

heuristic methods have strong assumptions on when links may exist. For instance, the common neighbor heuristic assumes that two nodes are more likely to connect if they have many common neighbors, which is not the case in many real-life networks. Hence, we shall not be considering them for our task.

2) Graph Neural Networks: Node2vec [3] is a popular graph embedding method that encodes nodes into rich low dimensional vectors. Then a multi-layer perceptron (MLP) predictor can be applied that uses the combination of the original node features and the Node2vec output vectors as input. GraphSAGE [4] is a widely used graph neural network that uses sample and aggregation tricks to support inductive learning for unseen nodes. DeepWalk [12] is another technique used to generate node embeddings and is useful in computing node similarity. We talk more about GraphSAGE and DeepWalk in Section 5. DeepWalk doesn't take into account the local neighborhood of the nodes and although we can use text embeddings in GraphSAGE, we believe we can get richer representations by concatenting raw text with GraphSAGE embeddings (inspired by ResNet [5]).

3) State of the Art: Zhang and Chen [17] proposed SEAL [17], which could learn heuristics from from local subgraphs using a GNN by extracting a local subgraph around each target link. Zhang and Chen also put forth the Weisfeiler-Lehman Neural Machine (WLNM) [16] where they extract local enclosing subgraphs around links as the training data, and use a fully-connected neural network to learn existence of links. Ai et al [2] improved upon the results of SEAL by using a Structure Enhanced Graph neural network (SEG) for link prediction. In this work, we shall not be using SEAL which is out of the scope of this project.

3 DATASET DESCRIPTION

We will perform edge prediction on the High Energy Physics Theory Collaboration Network dataset. *Data Source:* This dataset, collected and originally processed for the 2003 KDD Cup [6], contains information about scientific collaboration between authors on arxiv.org's High Energy Physics category between January 1993 and April 2003. A cleaned version of the dataset containing metadata for each individual paper, a list of directed edges, and temporal information for when each paper was published, is hosted online by the Stanford Network Analysis Project [8]. Each node in the dataset graph represents a paper on the site, while the directed edges indicate one paper has cited another. For some basic statistics about this dataset see table 1 below.

Property Type	Value
Number of edges	352807
Number of nodes	27770
Average node degree	13
Maximum node in degree	2414
Minimum node in degree	0
Maximum node out degree	562
Minimum node out degree	1

Table 1: Statistics for the arxiv.org High Energy Physics Dataset.

Data preprocessing: All of the files in this dataset are text files. Both the edge list and temporal data are a single file, and there is an individual text file for each paper (node) in .abs format containing its metadata which was turned into a textual embedding for our edge prediction task. While the edge list is already in a usable format for our prediction task, the metadata required additional processing. We preprocessed the metadata by extracting the relevant information for each node (abstract) from their individual text file and writing this data to a single, tab delimited text file for the full dataset.

4 EXPERIMENTAL SETTINGS

As discussed earlier, our goal is to predict if a given Paper A will cite Paper B, a link-prediction (classification) task. In order to compare performance across models, we collected two primary metrics: i) Accuracy, ii) Precision, iii) Recall, iv) F1 Score, and v) ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) Score.

All of the experiments for this project were run in an identical environment to ensure our results were valid. We used the same training and testing data split (90-10) with the same random seed for each model and the same set of positive and negative samples created from our graph. We used an equal number of positive and negative edges (half of the edges were of each type) for our model training and testing. Hyperparameters that maximized performance were chosen through cross-validation on our training set. The optimal parameters for each model tested in this experiment are listed in table 2 below.

	DeepWalk, Sentence-Transformers, Combined	GraphSAGE
Training Epochs	30	100
Learning Rate	0.0001	0.01
Optimizer	ADAM	ADAM
Loss Function	Binary cross-entropy loss	Binary cross-entropy loss

Table 2: Optimal hyperparameters for experimental models

These models were trained using the default settings for GPUs on Google Colab.

5 METHODS

The task of citation prediction can be performed using approaches that rely completely on i) Natural Language Processing (NLP), and ii) Graph Neural Networks. Pure NLP approaches leverage the information encoded in the abstract of the two papers (Paper A and Paper B) to create a combined embedding that is used to make the prediction. GNN based approaches utilize the local/global structure of the graph and the node's neighborhood to make these predictions. We tested both language and graph based embedding models, then compared their performance with our combined approach of concatenated graph and language embeddings.

For this project, we utilized three existing methods to generate node embeddings on our dataset: two graph-based methods, DeepWalk and GraphSAGE, and one text-based, a Sentence-Transformer model. These were then compared to our newly proposed method

of concatenation that utilizes both textual and graph embeddings.

5.1 DeepWalk

Deepwalk is a deep learning technique, originally proposed by Perozzi et al. [12], that generates embeddings for vertices/nodes in a graph network such that two nodes have similar embeddings if they are similar in the network. DeepWalk defines similarity in the network based on co-occurrence in random walk, i.e, two nodes u and v are similar if they have a high probability of co-occurring in random walks. The Deepwalk algorithm has two components:

- Random Walk Generator: Randomly sample a vertex u from the Graph and generate fixed length random walks originating from u.
- Update Procedure: Deepwalk uses the SkipGram technique [9] to update embeddings of a vertex given a random walk generated from it. To speed up the training procedure, Deepwalk uses Hierarchical Softmax. Stochastic Gradient Descent (SGD) is used for training.

Attribute	Value
d(dimension of embedding)	128
t (length of random walk)	40
gamma (number of random walks)	80
w (window size)	10
Link to Code	Deep Walk GitHub

Table 3: Details for original implementation of DeepWalk

Reason for choosing method: We chose this method of creating graph based embedding because our task may benefit from the notion of similarity in DeepWalk. In random walks, if two papers co-occur together more, then they should have more similar embeddings.

5.2 GraphSAGE

GraphSAGE [4] is a generalized graph convolution network (GCN) technique. It generates embeddings for vertices/nodes in a graph by sampling and aggregating features from a node's local neighborhood. It is able to learn the structure of each node's neighborhood as well as the distribution of node features in said neighborhood, making it more robust.

GraphSage trains aggregator functions (mean, pool, LSTM) that aggregate feature information from the local neighborhood of a node. For training the functions in an unsupervised fashion, a graph based loss (binary cross-entropy) is used along with negative sampling. The embeddings that are used in this loss function are generated using the aggregated embeddings of a node's neighbors.

Reason for choosing method: These methods can use textual features of nodes to create aggregated embeddings. Since we want to use text of the abstract of the paper as a feature, we believe this technique would perform well on our task.

5.3 Sentence-Transformers

Sentence-transformers are a type of natural language processing model created by modifying a pretrained transformer network (BERT, for example) to use Siamese and triplet network structures in order to derive semantically meaningful sentence embeddings [14]. The specific sentence-transformer model we used, all-MiniLM-L6-v2 [13], is a natural language processing model from the Python sentence-transformers library trained to maximize the difference in embeddings between paired contrasting sentences. Using these pretrained weights, the model encodes a given sentence or short paragraph as a multi-dimensional feature vector.

Attribute	Value
Non-Linearity)	Rectified Linear Units (ReLU)
K (number of aggregators)	2
Sample Sizes	$S_1 = 25$ and $S_2 = 10$
Best Aggregators	LSTM and Pooling
Link to Code	GraphSAGE GitHub

Table 4: Details for the original implementation of GraphSAGE

Attribute	Value
Embedding Size	384
Training Seq Length	128
Optimizer	AdamW
Link to Code	HuggingFace Repository

Table 5: Details for original implementation of all-MiniLM-L6-v2

5.4 Combined Graph and Text Features

Our combined embedding helps to capture the features included in both a pure NLP approach and a pure graph based approach. We first generate an independent set of textual embeddings using the all-MiniLM-L6-v2 sentence-transformer model as well as a set of graph embeddings using DeepWalk. The two embeddings for each node in the dataset are then concatenated together before being used as inputs to a two layer multi-layer perceptron model, as indicated in the diagram below.

6 EXPERIMENTS

We used the three existing methods for embedding generation and one new method of concatenating embeddings as described in the previous section to predict positive and negative edges in the High Energy Physics Theory Collaboration Network. The positive edges refer to the edges that exist in the original dataset, and the negative edges refers to the edges in the reversed graph. The final dataset consisted of an equal number of positive and negative edges, alleviating the problem of dataset imbalance. After training each of these models on our training set and selecting the best performing set of hyperparameters, we used each of these to predict edges on a held out test set containing 10% of our data. Our experimental results can be seen at the end of this section in table 10.

6.1 Baseline 1: GraphSAGE

As mentioned in section 5.2, GraphSAGE can be trained by passing in node embeddings to the model as input. These node embeddings can represent particular features or simply be initialized as

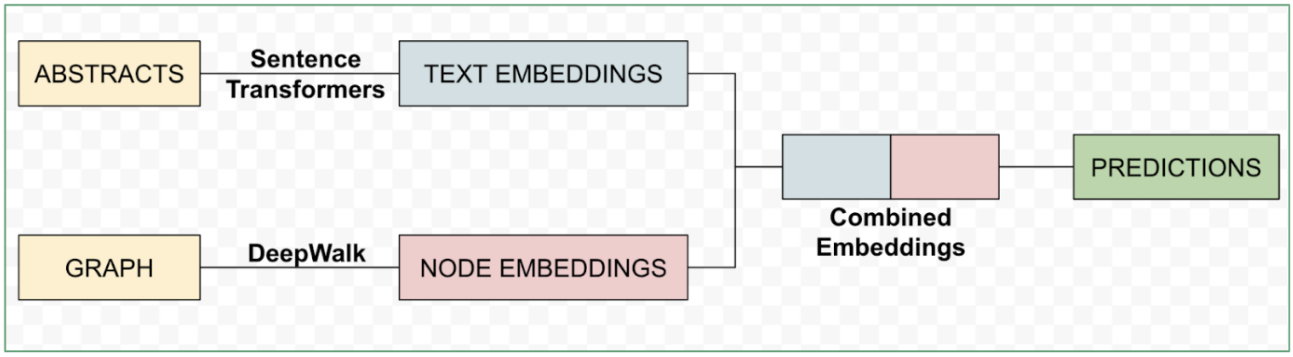

Figure 1: Model architecture for our proposed GAT-CAT model. Our model takes the sentence embeddings as well as the node embeddings as inputs, concatenates them to get a single node-embedding. For any two nodes u and v, we concatenate the combined node of both these nodes, and pass this as input to a two-layer MLP network to predict if they are connected by an edge.

a uniform value (for example, all ones). We trained two sets of GraphSAGE models using different types of features:

(1) GraphSAGE Node - Intializing node embeddings as a vector of ones
(2) GraphSAGE Text - Intializing node embeddings as the text embeddings of each paper abstract generated by our Sentence-Transformer model in section 6.3

Model	Accuracy	ROC-AUC	Prec	Recall	F1
Text	0.797	0.875	0.776	0.836	0.805
Node	0.664	0.752	0.621	0.842	0.714

Table 6: Experimental results: We evaluate the performance of our 2-layer MLP model after training with GraphSAGE embeddings (both Text and Node)

6.2 Baseline 2: DeepWalk

As mentioned in section 5.1, Deepwalk [12] is a graph based model that learns contextual information about each node using random walks. Hence, two nodes would have similar embeddings if they are similar in the network. In random walks, if two papers co-occur together often, then they would have similar embeddings and hence, we selected this method as a baseline for our experiments. We used DeepWalk to generate node embeddings of our data and then trained our multi-layer perceptron on these embeddings. We evaluated the network against the test set using various metrics, as shown in table 7

Accuracy	ROC-AUC	Precision	Recall	F1
0.807	0.877	0.802	0.814	0.866

Table 7: Experimental results: We evaluate the performance of our 2-layer MLP model after training with node embeddings generated by DeepWalk [12]

6.3 Baseline 3: Sentence-Transformers

We used a pretrained Sentence-Transformer model, all-MiniLM-L6-v2 [13], for this edge prediction task. To obtain a sentence embedding for each paper (node) in our graph, we directly input the raw abstract text for each paper into the Sentence-Transformer model to obtain a 384-dimensional sentence embedding vector for that paper. These were then passed through two fully connected MLP layers with a ReLU nonlinearity in between to obtain our positive and negative edge predictions. The results for this baseline Sentence-Transformer model can be seen below in Table 8.

Accuracy	ROC-AUC	Precision	Recall	F1
0.846	0.921	0.811	0.901	0.853

Table 8: Experimental results: We evaluate the performance of our 2-layer MLP model after training with textual embeddings obtained from a sentence tranformer model [15]

.

6.4 Proposed Method: GAT-CAT

The architecture for our proposed method is described in Figure 1. We generate the sentence embeddings using the same sentence-transformer [15] models that we used in our text-based baseline. The node embeddings themselves were generated using DeepWalk [12], one of our other baselines. We generate the text and node embeddings for each node in our graph, and then concatenate them together to get a single combined embeddings. The Graph and Text Embedding ConCATenation (GAT-CAT) model consists of two MLP layers with a ReLU activation unit, and uses the combined node embeddings of node u and node v as inputs. Let E_u and E_v be the combined node embeddings of nodes u and v respectively. Then the network can be represented by the following equations:

$$o_1 = \sigma(W_1[E_u, E_v] + b_1)$$
$$o_2 = W_2 o_1 + b_2$$

The DeepWalk node embeddings were 128 and the Text embedding dimensions were 384, giving the total embedding size of 512. Simply concatenating the combined node embeddings of nodes u and v works because our model is being trained on the task of directed edge prediction. In the case of undirected edge prediction, we will have to use others methods of combining the node embeddings.

Accuracy	ROC-AUC	Precision	Recall	F1
0.877	0.932	0.852	0.912	0.881

Table 9: Experimental results for our combined GAT-CAT Model on the Citations Dataset

We can see that our proposed GAT-CAT model significantly outperforms all the proposed baselines and achieved and AUC-ROC score of 0.932 and an accuracy score of 0.877. We compare these metric scores against all the other baselines in Table 10.

Model	Accuracy	ROC-AUC
GraphSAGE - Node	0.751	0.663
GraphSAGE - Text	0.797	0.875
DeepWalk	0.807	0.877
Sentence-Transformers	0.846	0.921
GAT-CAT	**0.877**	**0.932**

Table 10: Experimental results: We evaluate our proposed GAT-CAT model against all the other baselines and report a 3.10% increase in accuracy over the best-performing text-embedding method.

7 CONCLUSION

Our combined Graph and Textual Concatenated embedding model performed very well on this edge prediction task, having the highest accuracy and ROC-AUC score of all the models we tested. However, while concatenation is a simple process, it requires calculating and storing both textual and graphic embeddings, meaning it takes a greater amount of computation power and storage than any of the other methods studied in our experiment. While this is a potential shortcoming of this model, the embedding generation step of our model testing was much shorter than the training of the multi-layer perceptron model used for predictions, and the increased accuracy that arises from concatenating existing embeddings from quickly calculated methods like DeepWalk and Sentence-Transformers is likely worth a slight increase in cost for most edge prediction tasks.

Concatenation of existing graph and textual embeddings is an easy to implement strategy and appears to have enhanced predictive abilities beyond either type of embedding on its own. However, in this experiment we only tested a limited number of existing embedding techniques (DeepWalk and Sentence-Transformers) in our concatenated embeddings. Using more recent graph or language embedding methods may allow us to improve performance even further on edge prediction tasks with few changes to this model. Additionally, we may be able to improve the predictive abilities of Graph Convolutional Networks (GCNs) or GraphSAGE models by using these concatenated embeddings for model training because of their inclusion of both textual and graph information. There are many possibilities for enhanced performance using this technique that are yet to be realized.

8 DIVISION OF LABOR

All team members have contributed equal amounts of efforts, from researching topics, finding datasets and writing the report.

REFERENCES

[1] Lada A Adamic and Eytan Adar. 2003. Friends and neighbors on the Web. *Social Networks* 25, 3 (2003), 211–230. https://doi.org/10.1016/S0378-8733(03)00009-1

[2] Baole Ai, Zhou Qin, Wenting Shen, and Yong Li. 2022. Structure Enhanced Graph Neural Networks for Link Prediction. https://doi.org/10.48550/ARXIV.2201.05293

[3] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. https://doi.org/10.48550/ARXIV.1607.00653

[4] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[6] J. M. Kleinberg J. Gehrke, P. Ginsparg. 2003. *Overview of the 2003 KDD Cup*. https://www.cs.cornell.edu/home/kleinber/kddcup2003.pdf

[7] Glen Jeh and Jennifer Widom. 2002. SimRank: A Measure of Structural-Context Similarity. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Edmonton, Alberta, Canada) *(KDD '02)*. Association for Computing Machinery, New York, NY, USA, 538–543. https://doi.org/10.1145/775047.775126

[8] Jure Leskovec. 2003. *High-energy physics theory citation network*. https://snap.stanford.edu/data/cit-HepTh.html

[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. https://doi.org/10.48550/ARXIV.1301.3781

[10] M. E. J. Newman. 2001. Clustering and preferential attachment in growing networks. *Physical Review E* 64, 2 (jul 2001). https://doi.org/10.1103/physreve.64.025102

[11] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank Citation Ranking: Bringing Order to the Web*. Technical Report 1999-66. Stanford InfoLab. http://ilpubs.stanford.edu:8090/422/ Previous number = SIDL-WP-1999-0120.

[12] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. https://doi.org/10.1145/2623330.2623732

[13] Nils Reimers. 2021. all-MiniLM-L6-v2. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[14] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *CoRR* abs/1908.10084 (2019). arXiv:1908.10084 http://arxiv.org/abs/1908.10084

[15] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. https://arxiv.org/abs/1908.10084

[16] Muhan Zhang and Yixin Chen. 2017. Weisfeiler-Lehman Neural Machine for Link Prediction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Halifax, NS, Canada) *(KDD '17)*. Association for Computing Machinery, New York, NY, USA, 575–583. https://doi.org/10.1145/3097983.3097996

[17] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. https://doi.org/10.48550/ARXIV.1802.09691