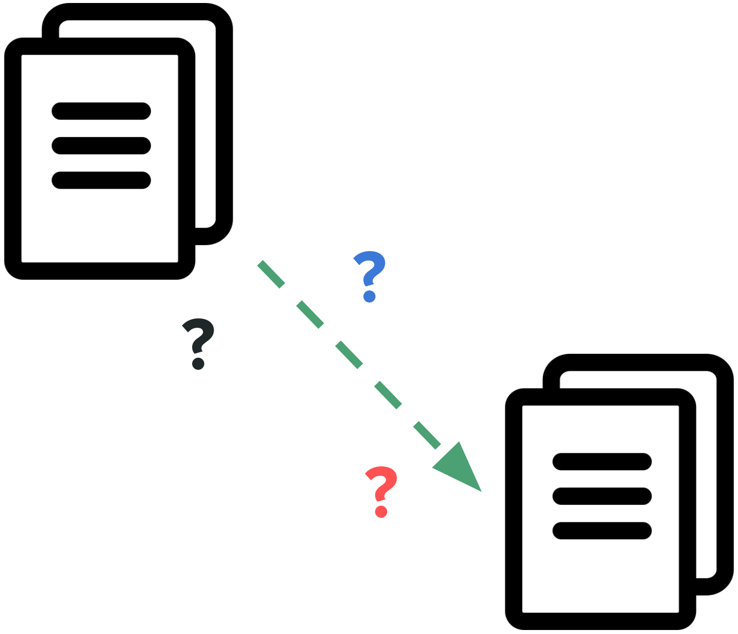
Better Together

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

Christian Clark, Abhinav Gupta, Shashank Srikanth, Suryatej Reddy Vyalla

Project Overview

- Given two academic papers, can we determine if one cited the other?
- There are two current approaches:
  - Text content (NLP) based
  - Graph based
- What would happen if we combine them?



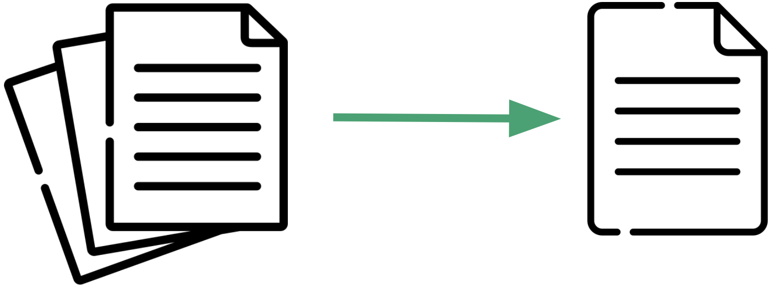
Project Dataset

High-Energy Physics Citation Network

- Collection of **27,770** physics papers on arxiv.org published from January 1993 to April 2003
    - 352,807 citations (edges) between them
- Stored as one text file containing the edge list, one text file containing temporal data, and a text file for each paper containing its abstract and metadata
- Collected by SNAP at Stanford, originally used for 2003 KDD Cup

Dataset Processing

- Overall data was very clean - only 1 erroneous edge existed in the edge list and every paper had associated text data
- We used the abstract text only, no metadata
  - Removed excess text from each file
  - Abstracts were then stored in a single line separated text file rather than 27,000 individual ones

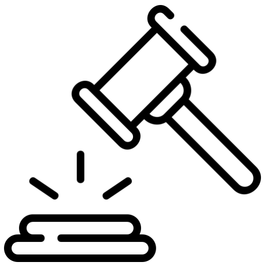


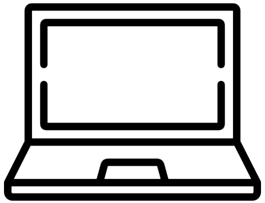
Dataset Statistics

Property	**Value**
Number of edges (citations)	352,807
Number of nodes (papers)	27,770
Average node degree	13
Maximum in-degree	2,414
Minimum in-degree	0
Maximum out-degree	562
Minimum out-degree	1

Why does this matter?

- There are many types of social networks where users have associated text where predicting links between users would be useful
  - Academic networks
  - Social media
  - Legal case precedent
  - Computer program logs





Prediction Task

Predicting citations between papers is an edge prediction task!

We performed the same process for each model:

1. Use the given method to create embeddings for the nodes in the graph
2. Predict edges with these embeddings using a multi-layer perceptron model
   a. Each model was trained using binary cross-entropy loss on positive and negative samples
3. Evaluate the performance of the model by calculating its accuracy and ROC-AUC score

Experimental Details

- Our dataset was split into 90% training, 10% testing
  - This consisted of an equal number of positive and negative samples from the graph
  - The data in each split was created in the same manner for each embedding method with the same random seed

	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

Baseline Models: GraphSAGE

- A graph neural network model that can incorporate information about a node's neighbors in order to create embeddings for said node
  - Users are able to choose the feature vectors input into the model for each node
  - We used the textual embeddings created using our *sentence-transformer* model
- **Problem with this method:** aggregation of many nodes to create embedding may result in an individual node's unique textual features becoming obscured

	Accuracy	**ROC-AUC Score**
GraphSAGE	0.797	0.875

Baseline Models: DeepWalk

- A graph based model that learns contextual information about each node using random walks
- **Problem with this method:** doesn’t include any information about a node’s textual features

	Accuracy	**ROC-AUC Score**
DeepWalk	0.807	0.877

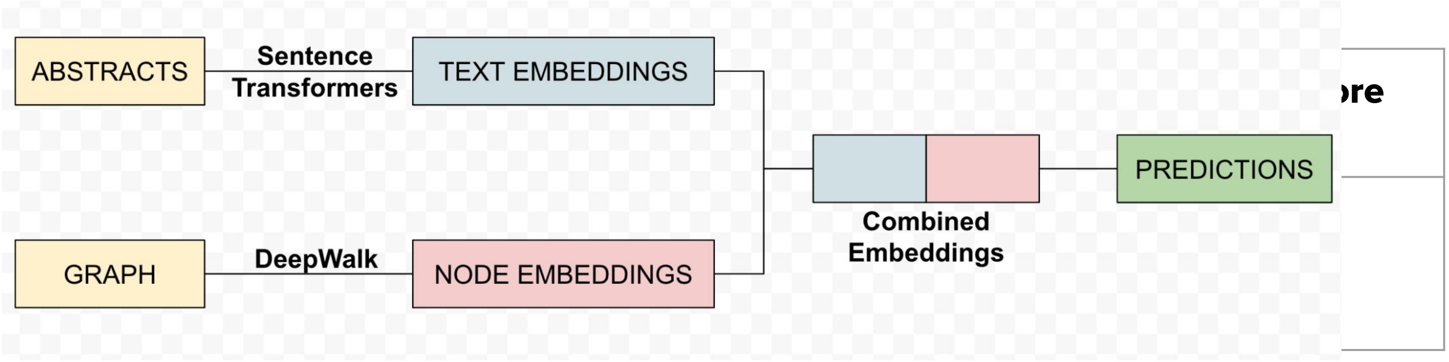
Baseline Models: Sentence-Transformer

- A modified version of BERT/RoBERTa sentence embedding models that reduces computation by calculating similarity on subset of embeddings
  - The specific model we used **all-MiniLM-L6-v2** was trained using a contrastive objective (maximizing the difference in embeddings between unlike sentences)
- **Problem with this method:** doesn't utilize the contextual information contained within the graph

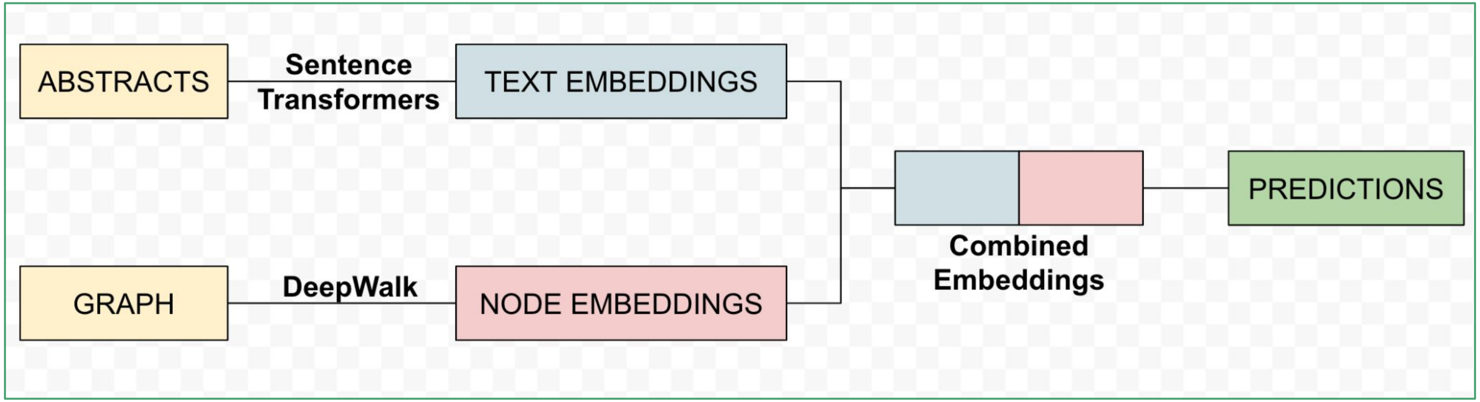
	Accuracy	**ROC-AUC Score**
Sentence-Transformer	0.846	0.921

Our Combined Approach

- We concatenated embeddings from both model types (graph and textual) before inputting them into the multi-layer perceptron model
- Combining both allows us to obtain information about a node's context in the graph while also retaining its *individual* language features



Our Combined Approach



	Accuracy	**ROC-AUC Score**
Combined Graph-NLP (DeepWalk + Sentence-Transformers)	0.877	0.932

Model Comparison

Embedding Method	**Accuracy**	**ROC-AUC Score**
GraphSAGE (w/o text embeddings)	0.751	0.663
DeepWalk	0.807	0.877
Sentence-Transformers	0.846	0.921
GraphSAGE	0.797	0.875
Combined Graph-NLP	**0.877**	**0.932**

Conclusion

- Existing graph and language methods are fairly effective for edge prediction in network datasets that include associated text
    - Many applications, from online social networks to network security

- However, combining these methods can improve their predictive abilities and is very simple!

Future Work

- As new, better performing textual and graph embedding methods are developed further performance improvements may be possible by using those to create concatenated embeddings
- Try GraphSAGE or other GCN on node and text embeddings concatenated with each other, which would take place **before** the embeddings are learnt.
  - Evaluate End-to-End learning vs Unsupervised learning methods
- Incorporate temporal features into the node embedding algorithms