CS550 Project Report

Group : True Positives

October 2024

Group Members

1. 12240420 Bhavya Jain
2. 12240880 Kriti Arora
3. 12240740 Kaki Venkata Vaneesha
4. 12240950 Mallarapu Hema Varshini

The GitHub link for our project can be found here

1 Project Overview

Our project focuses on **document classification** by leveraging both text and graph embeddings. We have explored various models from the literature and are combining their strengths to create a novel approach. Below is a summary of the methodologies and models involved in our project.

1.1 Background and Motivation

In our research, we identified several models that approach document classification through different embeddings:

- **TextGCN Model**: We found a paper [1] in which TextGCN creates a graph of words from the documents and treats documents as nodes in the graph. This graph is used to generate text embeddings for classification tasks. Each node (document) gets an embedding based on word co-occurrences in the graph, effectively capturing semantic relationships in the text.
- **GNN on Citation Graphs**: Another paper [2] presented models that use Graph Neural Networks (GNNs) on citation graphs, where the documents (nodes) are linked based on citation relationships. The GNN model, alongside graph-based algorithms like Node2Vec, is trained to produce embeddings for documents. These embeddings are used to classify documents based on their citation network. This paper has benchmark graph datasets.
- **Combining Text and Graph Embeddings**: Inspired by a third paper [3], which combined text embeddings (e.g., Doc2Vec) and graph embeddings from citation graphs, we aimed to integrate both types of embeddings into a unified classification model. This combination can potentially capture both the semantic meaning from the text and the structural relationships from the citation network.

[1]https://doi.org/10.48550/arXiv.1809.05679
[2]https://doi.org/10.48550/arXiv.2005.00687
[3]https://doi.org/10.48550/arXiv.2209.12474

1.2 Our Approach

We developed four models for document classification:

1. **TextGCN**: This model generates embeddings by creating a word-document graph from the dataset. The model treats words and documents as nodes, and edges represent co-occurrences between words in the same document. The embeddings produced are trained for the classification task.

2. **GNN (e.g., Node2Vec)**: This model focuses on citation graphs. Each document is a node, and citations between documents form edges. We use GNN models like Node2Vec to generate graph embeddings. These embeddings capture structural patterns, such as the citation relationships between papers.

3. **Combined Model**: The third approach combines the embeddings from the TextGCN and GNN models. The integration could be done through concatenation or summation of the embeddings, or by employing ensemble learning. This model seeks to leverage both textual and structural information for classification.

4. **BM25**: This model is used as a retrieval-based classifier. For a given query text, BM25 retrieves the top 5 most similar documents. The labels of these top 5 documents are used for a voting-based classification of the query. Essentially, the predicted class is determined by the majority class among the top retrieved documents.

1.3 Novelty

The novelty of our project lies in the **combination of graph embeddings from citation networks and text embeddings** from documents for classification. While previous works have explored text-based classification or citation-based classification independently, we combine both sources of information, which allows us to capture both semantic and structural patterns.

1.4 Dataset

A key challenge was identifying a suitable dataset that contains:

- **Text data**: Document content for text-based models.
- **Citation graphs**: Citation relationships between documents for graph-based models.
- **Labels**: Ground truth for supervised classification. Given is the link for the labels mapping to actual labels. [4]

Our dataset satisfies these criteria, allowing us to experiment with and compare the performance of different models.

Progress

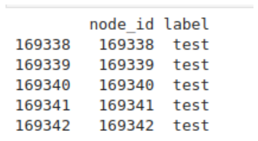
Data Preparation: We have collected data from here (citation graph) and here[5] (raw data). The dataset included:

- The ogbn-arxiv dataset is a directed graph, representing the citation network between all Computer Science (CS) AR X IV papers indexed by MAG.
- The tsv file that maps paper IDs into their titles and abstracts are available here. There are three columns: paperid $\hat{t}$itle $\hat{a}$bstract.

and did the following preprocessing to the data

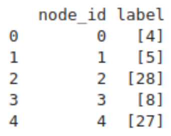
[4]https://arxiv.org/archive/cs
[5]https://doi.org/10.48550/arXiv.2005.00687

- Preparing the input in the format of datasets provided by TextGCN. [6]
- Cleaning the text: Stopword Removal, punctuations, using regular expressions etc.
- Mapping the nodes to the paper indexes, combining the two data sets together



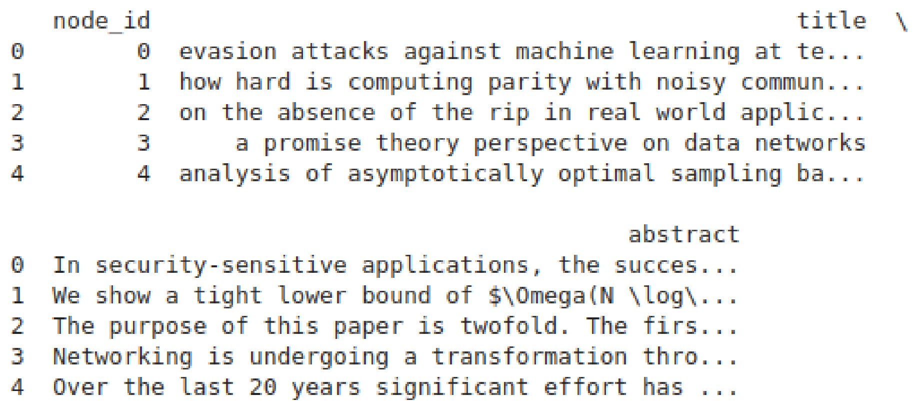
```
        node_id label
169338   169338  test
169339   169339  test
169340   169340  test
169341   169341  test
169342   169342  test
```

Figure 1: mapping from node id to train/test label



```
   node_id label
0        0   [4]
1        1   [5]
2        2  [28]
3        3   [8]
4        4  [27]
```

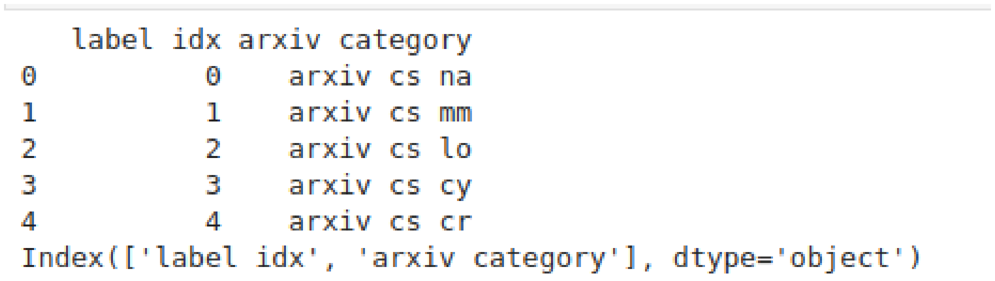
Figure 2: mapping from node id to class label



```
   node_id                                              title  \
0        0  evasion attacks against machine learning at te...
1        1  how hard is computing parity with noisy commun...
2        2  on the absence of the rip in real world applic...
3        3      a promise theory perspective on data networks
4        4  analysis of asymptotically optimal sampling ba...

                                            abstract
0  In security-sensitive applications, the succes...
1  We show a tight lower bound of $\Omega(N \log\...
2  The purpose of this paper is twofold. The firs...
3  Networking is undergoing a transformation thro...
4  Over the last 20 years significant effort has ...
```

Figure 3: mapping from node id to title and abstract



```
   label idx arxiv category
0          0    arxiv cs na
1          1    arxiv cs mm
2          2    arxiv cs lo
3          3    arxiv cs cy
4          4    arxiv cs cr
Index(['label idx', 'arxiv category'], dtype='object')
```

Figure 4: mapping from label id to arxiv category

Modeling Progress: We have implemented several models, including:

[6]https://doi.org/10.48550/arXiv.1809.05679

- **GNN** - Using the citation graph from the dataset, we implemented GNN to classify a small set of data. Our total dataset was around 1.5Lakh documents. We used a reduced set of nodes and documents to test run the GNN code for 20 epochs and 5 runs. This reduced dataset was divided into train and test data, and the findings are mentioned in the table below.
- **Node2Vec** - Node2Vec uses a three-layer neural network to learn node representations by predicting the context nodes for each target node. The negative sampling strategy significantly reduces computational complexity by focusing on a small subset of context and negative samples. This method effectively captures the local and global structural properties of the graph, producing high-quality node embeddings.

Results: The GCN model achieved the following accuracy.

Table 1: GCN Performance Results

Set	**Accuracy (%)**	**Train Loss**	**Valid Loss**
Train	53.00	0.0187	0.25
Valid	16.00	0.0448	0.40
Test	31.00	0.0559	0.41

Table 2: Training, Validation, and Test Accuracies and Loss of GNN per Epoch for Run 01 and Run 05

Run	**Epoch**	**Loss**	**Train Accuracy (%)**	**Valid Accuracy (%)**	**Test Accuracy (%)**
01	01	4.2399	15.09	6.25	9.68
01	02	1.6438	20.75	6.25	12.90
01	03	0.8556	28.30	25.00	32.26
01	04	0.5426	33.96	6.25	12.90
01	05	0.3570	39.62	12.50	6.45
01	06	0.2887	64.15	12.50	9.68
01	07	0.1718	75.47	25.00	16.13
01	08	0.1520	81.13	31.25	25.81
01	09	0.1095	81.13	25.00	29.03
01	10	0.0927	81.13	25.00	35.48
01	11	0.0559	83.02	25.00	32.26
01	12	0.0448	83.02	25.00	32.26
01	13	0.0409	83.02	25.00	29.03
01	14	0.0406	84.91	25.00	25.81
01	15	0.0388	86.79	18.75	29.03
01	16	0.0259	88.68	25.00	41.94
01	17	0.0190	92.45	25.00	41.94
01	18	0.0141	96.23	25.00	38.71
01	19	0.0187	96.23	25.00	38.71
01	20	0.0089	96.23	31.25	38.71
05	01	4.0677	16.98	6.25	6.45
05	02	1.5656	26.42	12.50	12.90
05	03	0.8360	28.30	18.75	29.03
05	04	0.5103	32.08	25.00	29.03
05	05	0.3062	37.74	18.75	32.26
05	06	0.2061	45.28	18.75	29.03
05	07	0.1414	50.94	18.75	29.03
05	08	0.0940	56.60	12.50	29.03
05	09	0.0906	66.04	12.50	25.81
05	10	0.0499	67.92	6.25	29.03
05	11	0.0492	71.70	6.25	22.58
05	12	0.0371	75.47	12.50	25.81
05	13	0.0338	79.25	12.50	25.81
05	14	0.0368	81.13	6.25	32.26
05	15	0.0733	88.68	12.50	29.03
05	16	0.0299	90.57	12.50	25.81
05	17	0.0124	90.57	12.50	25.81
05	18	0.0128	92.45	12.50	25.81
05	19	0.0096	94.34	18.75	25.81
05	20	0.0074	96.23	18.75	25.81

Next Steps

Planned Improvements: The following steps outline our planned improvements:

- **Training on Full Dataset:** We will train the entire dataset using TextGCN to generate embeddings based on the textual data.
- **Combining Embeddings:** The next stage involves combining the TextGCN embeddings with the GNN embeddings from the citation graph an then predicting the class using the embeddings . We can combine using concatenation , neural networks, averaging etc.

- **BM25 method** We will also integrate the BM25 method, as described earlier, for document retrieval and then using voting for class prediction.
- **Ensemble Learning:** Instead of directly concatenating the embeddings, we aim to explore ensemble models (such as XGBoost) to improve performance by better leveraging the strengths of different models.
- **Dimension Alignment:** If the embeddings generated from the models have different dimensions, we will employ a neural network to align them to the same dimensional space for more effective integration.
- **Cross-Validation:** We plan to implement cross-validation techniques for more robust model evaluation and to ensure that our model generalizes well across various subsets of the data.
- **Hyperparameter Tuning:** Finally, we will continue to fine-tune hyperparameters for all models to optimize their performance further.

Further Validation: We will validate the models on a holdout test set to ensure generalization. Additional metrics such as F1-score and AUC will be used to evaluate the model performance.

Individual Contribution

Kriti Arora

- Input Graph Preparation: - Carefully structured the input graphs to ensure compatibility with GNN frameworks. Ensured accurate representation of relationships within the dataset.
- Research on Classification Techniques: - Conducted extensive research on various GNN classification techniques. Analyzed strengths and weaknesses of different methods and their applicability to various graph data types.
- Analysis of GNN Output: - Evaluated the performance of GNN classification using metrics. Conducted error analysis to identify misclassification patterns, providing insights for model refinement.

Bhavya Jain

- Conceived the methodology for our approach, combining text and graph-based embeddings.
- Prepared the input data in the required format for TextGCN, ensuring proper handling of the dataset structure.
- Cleaned and preprocessed the raw textual data to improve quality and consistency for graph formation.
- Constructed the word-document graph for TextGCN, mapping relationships between words and documents.
- Developed a sample Python script to implement the BM25 algorithm for document retrieval.

Kaki Venkata Vaneesha

1. Implementation of Node2Vec Algorithm: This code loads the Node2vec algorithm using PyTorch Geometric. It has taken the node embeddings from the graph, which generate a random walk, allowing the model to infer relationships and structural information from this data.

2. Training Configuration: The following script requires several hyperparameters which are fed in through the command line while running, such as embedding dimension, walk length, context size, and a chosen learning rate. This further gives users room to customize the training process according to their respective dataset and needs.

3. Saving the Embedding: After training the model, the script saves the learnt node embeddings to a file called (embedding.pt) using PyTorch's torch.save() function. This allows the learnt embeddings for further use in downstream tasks such as node classification or clustering.

Mallarapu Hema Varshini

1. Model Development: Implemented a configurable multi-layer perceptron (MLP) architecture for processing node features in graph neural networks.

2. Training and Evaluation Framework: Created robust training and evaluation functions, enabling effective model training and performance assessment using accuracy metrics.

3. Node Feature Integration: Added functionality to incorporate external node embeddings, enhancing the model's ability to utilize additional information for improved prediction performance.

Conclusion

In summary, we have made significant progress in data preprocessing and initial modeling efforts. We have overcome challenges such as class imbalance and overfitting, and we plan to focus on advanced modeling techniques and final validation in the next phase.