Using Graph Convolutional Neural Networks for NLP tasks

Sanchit Sinha
ss7mu@virginia.edu
University of Virginia

Mohit Sudhakar
ms5sw@virginia.edu
University of Virginia

ABSTRACT

In recent times, Graph Neural Networks are proposed as a paradigm shifting methodology in solving standard Natural Language Processing and Computer Vision tasks. Graph Neural Networks provide a much more generalised method of representing word embeddings, image pixels, etc. Analogous to the standard Deep Learning literature, there have been multiple variants of GNNs proposed - such as Convolutional GNNs, Recurrent GNNs, AutoEncoder GNNs, etc. Similar to using Convolution Networks in text classification, Graph Convolution Networks can also be used in text classification tasks. In this project, we plan to use Graph Convolutional Neural Networks to solve the open problem of Text Classification and Sentiment Analysis. Using a baseline model from Yao et al., we make certain modifications in the model training procedure keeping the architecture similar to the one proposed. Results on benchmark datasets show that the results improve from what are reported.

ACM Reference Format:
Sanchit Sinha and Mohit Sudhakar. 2020. Using Graph Convolutional Neural Networks for NLP tasks. In *Proceedings of ACM Conference (Conference'20).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

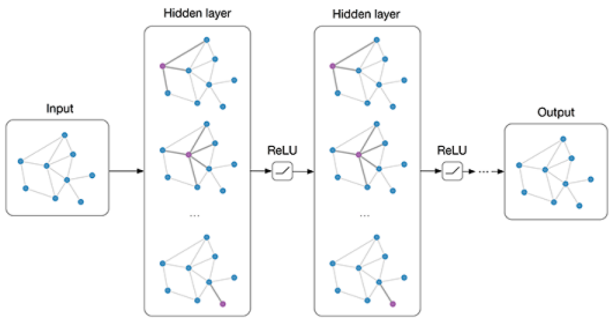
Text classification and sentiment analysis are one of the most widely studied tasks in NLP, with a multitude of literature proposing a number of different systems to solve them. Starting from original probabilistic approaches like Naive-Bayes and Bag of words modelling, the problem can be solved much effectively using deep learning approaches. On this note, well studied deep learning architectures such as CNN, RNN, Bi-LSTM, etc. have been proposed to effectively perform text classification. Although memory networks like RNNs, LSTMs, etc. have shown better results in certain NLP tasks, although text classification still uses CNNs in conjunction with other architectures to get good results. However, in recent times architectures which utilize the graph property of text as their inputs give much better results than conventiona word embeddings or automatic feature extraction. In this project we focus on using Graph Neural Networks - specifically Graph Convolutional Neural Nets for text classification. The efficiency of graph strucutre can be intuitively understood by the fact that graphs have much better understanding of the global corpus rather than only a local area.

As a node of a graph can have potentially arbitrary degree, we can exploit this fact to create very high dimension interactions, getting a much better idea of the neighbourhood of a single word. text classification has multiple usages in complicated applications such as spam filtering, document organization, curation, sentiment classification, etc.

1.1 Converting Text to Graphs

The biggest challenge in using GCN is to actually inputting data in a format which is usable by the GCN. To convert standard text data into graphs, we have to cleverly model word-word and document-document interactions in the input graph. As multiple studies before have shown, the performance of a model is as good as the input features ist receives. Hence, generating high quality features for the model is as important as the model itself. We utilize the graph generation method employed by Yao et al. [1]. Although many better methods might exist to construct graphs, the major aim of this project is not to explore these approaches hence we stick to the baseline approach.

1.2 Graph CNNs



A Graph CNN is similar to a multi layer CNN model which tries to generate an embedding vector based on the neighbourhood properties of a node in a graph. A GCNN is similar to a normal CNN in terms of its structure and architecture, although it differs in multiple fundamental ways such as how it defines convolution, pooling, etc. In recent years, the more generalized nature of GCNN has made them a constant source of research interests. The GCN used in this project is similar to the one proposed by Kipf and Welling , which utilises the weighted averaging properties of embedding layers. With Laplacian normalizations, the feature generation of the layers is much more regular.

1.3 Text Classification

Text classification is the process of assigning tags or categories to text according to its content. Several recent studies showed that the success of deep learning on text classification largely depends on the effectiveness of the word embedding or by extension the input graph. Text classification has multiple applications in a variety of fields such as spam filtering, document organization, curation, sentiment classification, etc.

2 RELATED WORK

- Semi-Supervised Classification with Graph Convolutional Networks [2]: This work was one of the pioneer works in defining Graph convolutions. The averaging property of layers is defined and the normalization regularization is also proposed. We use this paper to define the convolution structure of the network.
- A comprehensive survey on graph neural networks [3]: We studied this paper to get a sense of various convolution processes specific to graph and to get a general idea about their relation to standard deep learning models.
- GraphCNN on text classification: Yao et al. [1] This paper forms the baseline approach which we employ as the benchmark.
- Text Level Graph Neural Network for Text Classification [4]: Another approach using GCN for text classification. The same normalization is used in the paper to improve results.

3 METHODOLOGY

3.1 Preprocessing

We have followed the pre-processing steps considered by the authors, which are:

- Decoding the content into Latin1-encoding. This ensures there are no punctuation and special symbols in the corpus.
- Words only with frequency > 5 and no stop words are considered. However, stop words and highly frequent words are still considered in the MR dataset. The intuition behind this is that a lot of Movie reviews have similar words in them which are important in making the classification decision, hence they are not removed.

3.2 Convert text data to graph structured data

We have used Yao et al's [1] method to convert the data into graph structured format. The dataset 20-Newsgroups consists of 20 groups of news articles on various topics, such as science, politics, computers, automobiles and sports.

3.2.1 *Nodes.*
Nodes in graph are represented by both words and documents. Hence, the graph is heterogeneous with respect to words an documents. The total number of nodes in the graph is equal to the sum of unique words in the vocabulary and the number of documents in the corpus.

3.2.2 *Edges.*
Edges between words are created by taking a sliding window of nearby words. The default sliding window is 20. Edges between words and documents are formed if the word is present in the document. The intuition behind the sliding window is using it as a 1xn convolution operation, where every window acts as a convolution filter of size (1,n). Similar to the cases of images, the receptive field will be larger for larger size of the filter.

3.2.3 *Weights.*
Weights between word-word edges are added by calculating the pointwise mutual information (PMI) between words.

$$A_{ij} = \begin{cases} \text{PMI}(i,j) & i,j \text{ are words, PMI}(i,j) > 0 \\ \text{TF-IDF}_{ij} & i \text{ is document, } j \text{ is word} \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases}$$

Figure 1: Function used to construct the adjacency matrix. Source [1]

Positive PMI values are taken and added as weights. Negative PMI values are not added as edges. Weights between word-document edges are added by calculating the term frequency-inverse document frequency (TF-IDF) between words and documents.

The initial input matrix A is formed by the function detailed in Figure 1. As can be seen, the edges between two documents have no value, that is no relation is modeled between them.

3.3 Baseline model

We have implemented Kipf and Welling's[2] graph convolutional network on this generated graph dataset. The objective function for GCN is:

$$f(H^{(l)},\ A) = \sigma(\hat{D}^{-\frac{1}{2}}\ \hat{A}\ \hat{D}^{-\frac{1}{2}}\ H^{(l)}\ W^{(l)}) \tag{1}$$

where $\hat{A} = A + I$, I is the identity matrix and $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$. Here $\hat{D}^{-\frac{1}{2}}$ acts as the normalizing factor.

4 DATASETS USED

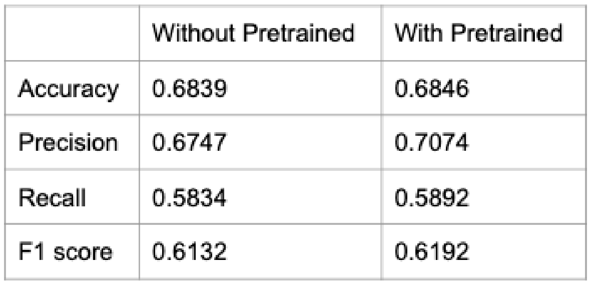
The following datasets were used for performance evaluation:

- The 20NG dataset: 18,846 documents (news reports) evenly categorized into 20 different categories. Training set (11,314 docs) and test set (7,532 docs).
- The Ohsumed23 corpus: From the MEDLINE database (medical literature - cardiovascular diseases). Training set (3357 abstracts) and test set (4,043). R52 and R83 : subsets of the Reuters 21578 dataset. (news reports)
- MR: one liner movie review dataset - binary sentiment classification. Same train/test split as in Tang, Qu, and Mei 2015

5 EVALUATION AND EXPERIMENTS

5.1 Experiment-0

(1) We have reproduced the results of the Yao et al paper and got an accuracy of 86.15% on the 20NG dataset which is same as that of the paper [Note: SOTA on 20NG is 88%] and 76% on MR dataset [Note: SOTA on MR is 78%].
(2) We have experimented by increasing the sliding window size and found that it increases accuracy, by increasing the context for a word and making the graph more dense. The intuition behind this is to study the effect of considering a higher order neighbourhood in the formation of the graph. As the accuracy increased, it implies that modelling relations with more words increases the performance and is indeed



	Without Pretrained	With Pretrained
Accuracy	0.6839	0.6846
Precision	0.6747	0.7074
Recall	0.5834	0.5892
F1 score	0.6132	0.6192

Figure 2: Comparison of GCN performance with and without pretrained Word embeddings

useful. To this note, we plan to use better word-word relationship modelling techniques as mentioned in (2) of Future Work.

(3) We have also tried using cosine similarity as similarity measure between word vectors and it improves accuracy by 1 percent point. This further corroborates our intuition of using better similarity scores and interactions.

5.2 Experiment-1 - Word Embeddings

We have explored the effect of using learned word embeddings rather than learning node embeddings on the fly. For this we used GloVe word2vec embeddings as they are generally used to capture word representations effectively. Although there are better representations like BERT that have come out recently we have decided to start our research with GloVe word2vec.
We have run experiments on the Ohsumed dataset which contains medical information. The experiment was run using the following hyperparameters:

Epochs $\longrightarrow$ 100
Word Embedding Dimension $\longrightarrow$ 300
GCN Layers $\longrightarrow$ 2
Optimizer $\longrightarrow$ *Adam*
Learning Rate $\longrightarrow$ 0.02
Dropout $\longrightarrow$ 0.5

Results from using pretrained word embeddings such as GloVe vectors show similar values in accuracy and recall and an increase in precision. This could be attributed to better learning due to new semantic information gained from using pretrained word representations over one-hot encoding. Note that these results are lower than state-of-the-art results because these are run only for 100 epochs, whereas in the original paper, they have run for more than 300 epochs. As shown in Figure 2, we can conclude that word embeddings do not degrade performance. This gives us confidence that better word embeddings along with the representations learned by the graph may lead to better prediction performance in text classification. We also note that higher dimensional embeddings may work better as they capture more diverse information, which may help in further improving performance.

5.3 Experiment-2

In the second experiment, we tried using a well known graph convolution method to generate embeddings from the adjacency matrix. The Chebyshev convolution is similar to the one proposed in [2] but instead of using the whole adjacenecy matrix, it employs on a reduced latent space which are defined using the eigenvectors of the matrix. We used the Chebyshev convolution upto 3 eigenvalues and used the reduced latent space to perform the convolutions. The experimental results on the MR dataset are presented in Figure 4. As it is evident the performance is not as good as Kipf and Welling's convolution operation implying that there might not be a good embedding in the latent space to work with. The curves are also showing classic signs of overfiting implying that some kind of regularization is needed to obtain better results. Below equations show the operation of the Chebyshev's convolution. The major factor in this convolution is that the calculation of the matrices takes a lot more computation time than the standard operations.

$$\begin{aligned}\mathbf{Z}^{(1)} &= \mathbf{X}\\ \mathbf{Z}^{(2)} &= \hat{\mathbf{L}} \cdot \mathbf{X}\\ \mathbf{Z}^{(k)} &= 2 \cdot \hat{\mathbf{L}} \cdot \mathbf{Z}^{(k-1)} - \mathbf{Z}^{(k-2)}\end{aligned}$$

6 CONCLUSION

(1) Various experiments and studies have demonstrated the usefulness and efficacy of GraphCNNs on NLP problems.
(2) Models are less data-hungry and are more robust to hyper-parameter setting
(3) Biggest drawback is that there is no sequence information. The global inference and relationship might be useful for text classification, but it is not generalisable to many other downstream tasks, where ordering of words is important
(4) Training documents are also used in the graph construction. There is no document-document interaction scores, which might be not very useful in classification but becomes useful in retrieval, seq2seq, etc. Also, prediction on unseen documents is not very easy. The whole graph needs to be rebuilt

REFERENCES

[1] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 7370–7377, 2019.
[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
[3] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *arXiv preprint arXiv:1901.00596*, 2019.
[4] L. Huang, D. Ma, S. Li, X. Zhang, and H. WANG, "Text level graph neural network for text classification," *arXiv preprint arXiv:1910.02356*, 2019.

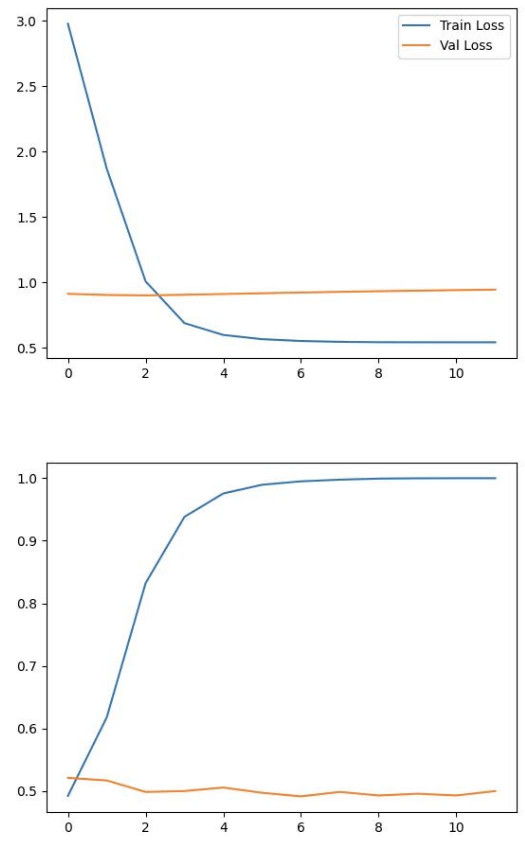


Figure 4: The loss and accuracy curves for the Chebyshev convolution operation.