National College of Ireland

Text Classification using Graph Based Learning

MSc Research Project
Data Analytics

Soumyadip Dipak Ghosh
Student ID: x1819821

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Soumyadip Dipak Ghosh
Student ID:	x1819821
Programme:	Data Analytics
Year:	2020
Module:	MSc Research Project
Supervisor:	Dr. Catherine Mulwa
Submission Due Date:	28/09/2020
Project Title:	Text Classification using Graph Based Learning
Word Count:	6574
Page Count:	17

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	28th September 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Text Classification using Graph Based Learning

Soumyadip Dipak Ghosh
x1819821

Abstract

Current text classification models based on deep learning increasingly rely on local word occurrence information and sequential semantics. These techniques the information residing long distance semantics and global word occurrence information. Recently, numerous researchers have explored graph neural networks (GNN) to apply on plethora of tasks as graph structures are particularly adept in capturing complex and abstract relations between entities which may prove to be highly useful in natural language processing related tasks. Convolution neural networks (CNN) have been very effective in deep learning for highly structured data like texts and images. Thus studies have recently begun to harness the power of convolution in a manner which would be effective for application on graph-structured data with GNNs. However, very less studies have explored the use of graph convolutional networks (GCN) for the purpose of text classification. This study aims to construct a graph from a corpus of text comprising of documents and words as nodes and use it for text classification using GCN. This will enable the neural network to learn from complex information residing in relationships between document-word and word-word co-occurrences. The approach shows state-of-the-art performance in multi-label classification on two out of the four popular benchmarking corpora used in this work to test our approach.

1 Introduction

Text classification has continued to exist as one of the primary problems in the field of Natural Language Processing (NLP). Numerous real world applications rely on text classification algorithms. These applications include fake news detection, SPAM filtering, document segregation and others (Jindal and Liu; 2007; Aggarwal and Zhai; 2012). Text representation learning is an essential technique in solving any NLP related problem. Widely used traditional methods which have proved very useful in text representation learning include one-hot encoding, bag-of-words, Term Frequency-Inverse Document Frequency (Tf-idf) and n-grams. Recently, text representation learning are employing deep learning techniques including convolutional neural networks (CNN) (Kim; 2014) and more popularly, an implementation of Recurrent Neural Network (RNN) known as Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber; 1997).

These methods rely more upon learning local and sequential information in text (Battaglia et al.; 2018) which can capture features derived from semantics and syntax residing only in local sequences of words. Information provided by global occurrences of words from a corpus is also of great significance and value for purposes like text classification (Peng et al.; 2018). Harnessing the information obtained from semantics which occur non-consecutively and further away from each other might improve text classification.

Recently, graph neural networks and graph embeddings has attracted wide attention and has become an exciting research direction (Battaglia et al.; 2018; Cai et al.; 2017). Graph neural networks have primarily proved to be effective in tasks with data which possesses high relational structure. Such data are commonly found from sources like social networks, biology, chemistry and others. For example, citation networks can reap huge benefits by utilising the relations between research papers in the form of a graph like structure. Another popular example would be the case of a social network as shown in Figure 1, a user has numerous relations with several types of entities. The entities can be but not limited to, another user, groups, pages, games and many others. All these entities are tied up with each other with varying types and levels of relationships.

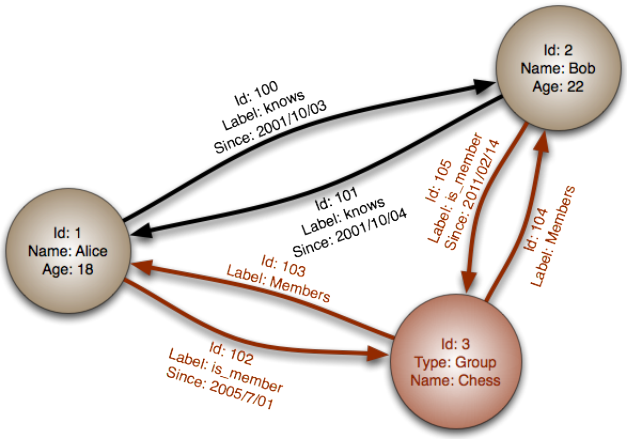

Figure 1: Graph data contains nodes, edges and attributes.

The entities can be modeled as nodes and the relationships as edges. The edges represent the connection between one node and another. An edge can be either directed or undirected. Directional edges are generally used to preserve some sort of hierarchical information which may be present between the nodes. Attributes provide more information about entities and their connections and thus can be present on both nodes and edges. Edges are the key concept as the successfully capture an abstraction which would have been lost otherwise if the data had been cast into a tabular format. These edges play a vital role in harnessing the relationships between the entities.

In this research project, we propose a method for text classification via a GNN based method. The first step is to construct a single large graph to represent the entire corpus of texts. In this case, a piece of text can be a paragraph, a document or a group of sentences, we will refer it as a document in this work. The words and documents will be represented as nodes with each node having their respective attributes.The classification will take place on the basis of document node attributes which will convert a node classification into a text classification problem. This work is primarily inspired from the work of Yao et al. (2019), on how this project constructs the graph and trains it using GCN (Kipf and Welling; 2016). GCN is an effective and simple combination of GNN and CNN to gain high order information of neighbourhoods.

Our implementation involves a masking technique to differentiate between word and document nodes. The edges will be of two types to represent the both document-word and word-word connections. Word co-occurrence information will be calculated using point-wise mutual information (PMI) and will be used to represent word-word edges, similar to that of TextGCN (Yao et al.; 2019). But our model will use document-word edges

calculated using an enhanced version of Term Frequency-Inverse Document Frequency (TF-IDF). This work will also analyse the effect of reduction of word-word edges on the performance of the model to construct a smaller but effective graph as smaller graphs occupy less memory and reduces training time.

The rest of the report is organised as follows, Section 2 discusses the types of popular GNNs and what are the key concepts behind them. Section 3 contains information on datasets, the evaluation metrics to be used for model performance evaluation and which other models will be considered for accuracy comparison, This is followed by Section 4, which discusses the novel approach to solve the problem, and what changes have been introduced to improve the pre-existing model and why. Then 5 contains all the granular details about model implementation, various configuration details so that the reader can easily reproduce the results. In Section 6, we compare our model performance with other models as per discussion in Section 3 followed by some discussion on where it fails and why. Report is concluded with future scope in Section 7.

1.1 Research Question

RQ: *How well can Graph Convolutional Networks perform compared to traditional neural networks in text classification?* Different neural network architectures will be compared to assess the performance of our model.

Sub-RQ1: *Can an enhanced scoring mechanism for document-word edges improve the performance of TextGCN?*

Sub-RQ2: *Can removal of insignificant word-word edges improve the performance of the model?* Different reduction proportions will be implemented and their effectiveness will be evaluated.

2 Related Work

Incorporating information from a graph-structure which can be fed to a trainable model is one of the fundamental areas of research in graph learning (Hamilton et al.; 2017b). More recently, numerous researchers are working on projects that aim to learn representations from the structural information of a graph. A basic graph learning problem for node classification can be summarised as following:

- A graph holds a given set of nodes with numeric attributes x_i for each of them.
- Output label y_i for each node has to be predicted.
- The nodes are related to other nodes by weighted edges, represented in the form of an adjacency matrix A.
- The attributes and relationships of related nodes put forward information from the additional context, the key concept is prediction of output y_i for node i.

2.1 Embedding Nodes

Representational learning for graphs or sub-graphs can be achieved by learning from a mapping of points in a lower dimensional space representing node embeddings in a vector space $\mathbb{R}^d$ (Hamilton et al.; 2017b).

The goal is to optimise these mappings so that the geometric relationship in the mapping best reflects the structure of the original graph, so that it can be used as feature inputs to a machine learning model. To extract information of graph structure, summary graph statistics are generally used(e.g., clustering or degrees coefficients) (Bhagat et al.; 2011), kernel functions (Vishwanathan et al.; 2008) and manually engineered features which can be used to measure neighbor structures (Liben-Nowell and Kleinberg; 2007). These features being inherently hand-engineered, have limited ability as they tend to be inflexible and not great for learning processes.

The manner in which the task of representing graph structures is approached is the key difference of representation learning with the traditional approach. Previous works treated it as a pre-processing step, relying on statistical information and features which are manually engineered to draw out structural information. Whereas, representation learning tries it as a machine learning task that emphasises on an approach that is data-driven for optimisation of embeddings to preserve the structure as close as possible to the real graph (Hamilton et al.; 2017b).

2.2 Graph Neural Networks

Along with graph embeddings, graph neural networks (GNN) was proposed as a novel approach by (Scarselli et al.; 2008), especially designed for graph-like structured data. The key concept can be considered as an algorithm responsible for message exchanging between nodes. every node is provided with a random embedding h_i^0, and then GNN accumulates input for every node according to equation 1.

$$\mathbf{h}_i^k = \sum_{v_j \in N(v_i)} h(\mathbf{h}_j, \mathbf{x}_i, \mathbf{x}_j) \tag{1}$$

where h is an arbitrary differentiable function of the form $h : \mathbb{R}^d \times \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$.Until the embeddings converge, eq 1 is applied recursively and h must be a contraction map. Embedding convergence will occur after K iterations, after which the final output will be computed using $\mathbf{z}_{v_i} = g(\mathbf{h}_i^K)$. Here g is an arbitrary differentiable function of the form $g : \mathbb{R}^d \to \mathbb{R}^d$. The work further discussed on various parameters of h and g based on the basic concept of the multi-layer perceptrons (MLPs). However, the algorithm has some limitations, the primary one being convergence of message passing iterations and the requirement of h to be a contraction map.

Li et al. (2015) improves the GNN implementation (Scarselli et al.; 2008) by extending it through addition of Gated Recurrent Units (GRU) paired with back propagation which takes place through time (Cho et al.; 2014). This enables the elimination of the requirement of equation 1 for convergence. GNN framework is modified to use GRUs in a way that node attributes perform initialisation so that intermediate sub-graph embeddings' output can be used.

Li et al. (2015)'s Gated-GNN initialises $\mathbf{h}_i^0$ vectors from node attributes (*i.e.*, $\mathbf{h}_i^0 = \mathbf{x}_i$) to update equations of the form,

$$\mathbf{h}_i^k = \text{GRU}\left(\mathbf{h}_i^{k-1}, \sum_{v_j \in N(v_i)} \mathbf{W}\mathbf{h}_j^{k-1}\right), \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{d \times d}$ denotes a trainable weight matrix. Gilmer et al. (2017)'s implementation brings a new level of abstraction for GNNs, like the form,

$$, \mathbf{h}_i^k = U \left(\mathbf{h}_i^{k-1}, \sum_{v_j \in N(v_i)} q(\mathbf{h}_i^{k-1}, \mathbf{h}_j^{k-1}) \right), \tag{3}$$

where $q : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{d'}$ denotes a differentiable function responsible for performing computations on messages incoming from neighbours and $U : \mathbb{R}^d \times \mathbb{R}^{d'} \to \mathbb{R}^d$ is a differential function to update node embeddings. This approach is widely known as Message Passing Neural Networks (MPNNs) further generalising Li et al. (2015)'s Gated-GNN for example. Gilmer et al. (2017) also discussed on other MPNN variants and implementations (*e.g.*, incorporating edge features) based on molecular property prediction based on their structure.

2.3 Graph Convolutional Networks

Kipf and Welling (2016) have proposed the use of a graph convolution network primarily aimed at solving the node classification problem. The approach basically proposes a neural network that uses x_j input features from all nodes j lying in the local neighbourhood of the node i. The output for the node i is its associated label y_i. The information acquired from the neighbours is combined using graph convolutions. The key concept is applying the convolutions across the entire graph, to accumulate features from the relevant neighbourhood for each node.

A GCN with a multi-layer propagation rule can be defined as follows:

$$H^{l+1} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H_{(l)} W_{(l)}) \tag{4}$$

where $\tilde{A} = A + I_N$ denotes the adjacency matrix of an undirected graph $\mathcal{G}$ and I_N is the identity matrix which is added to the original matrix to represent self-loops. $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is the degree matrix and $W^{(l)}$ is a weight matrix specific for layer l. $\sigma(.)$ is used to denote an activation function and activation matrix is denoted by $H^{(l)} \in \mathbb{R}^{N \times D}$ in the l^{th} layer.

Convolutions in neural nets can be easily applied on 1d, 2d or 3d tensors. It is similar to graphs, in a way that a node has uniform connectivity with its neighbouring nodes. Defferrard et al. (2016) provided a spectral take on the same by using it in the Fourier domain with Chebyshev filter. Spectral transforms generalise graph convolutions therefore can define as point-wise multiplication of signal spectral.

Exact convolution computation across a graph often becomes a computationally expensive and intensive process due to involvement of matrix diagonalisation. Kipf and Welling (2016) use only first order approximation of polynomials from Chebyshev filters to limit the amount of convolutional kernels. This quickens the computation and to prove it some popular datasets of the graph-domain were used to provide competitive results. Feature representations from graphs are learnt which is useful for node classification problems.

2.4 GraphSAGE

Inductive learning capability is important for production machine-learning systems as they require high-throughput and also possess the ability to work on changing graphs

and unseen nodes(Hamilton et al.; 2017a). The name GraphSAGE is extracted from the operation of sample and aggregation methodology highly useful for node embedding in an inductive manner by making use of node features to construct an embedding function which generalises on unseen nodes.

It trains a aggregate function group information from a node's neighbourhood features. The distinguishable factor lies in the manner it performs information aggregation by taking into account several factors like different hop counts, search depths. A graph-based loss function is used for output representations, $\mathbf{z}_u, \forall u \in \mathcal{V}$. Then the aggregator function's parameters and weight matrices are tuned, $\mathbf{W}^k, \forall k \in \{1, ..., K\}$ via stochastic gradient descent. The loss function tries to achieve similar representations for nodes and disparate representations for distinct nodes. The loss function is given by equation 5.

$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log(\sigma(\mathbf{z}_u \mathbf{z}_v)) - Q.\mathbb{E}_{v_n \sim P_n(v)} \log(\sigma(-\mathbf{z}_u \mathbf{z}_{v_n})), \tag{5}$$

where v is a node and u is a node that occurs on fixed length random walk.σ here is the sigmoid function. P_n denotes a negative sampling distribution whereas Q is the number of negative samples.$\mathbf{z}_u$ is the representation generated from the features of the neighbouring nodes.

Node classification results on Citation[1] and Reddit[2] data were provided and evaluated. Performance of GraphSAGE was found superior to that of GCN for the above tasks.

For evaluation, the authors compared four algorithm variants, differentiated by the aggregator functions being used. The aggregator functions used were Mean, LSTM based on Hochreiter and Schmidhuber (1997) and Max pooling.Convolution aggregator similar to the GCN framework (Kipf and Welling; 2016) was the fourth variant. Implementation was done using TensorFlow[3] (Abadi et al.; 2016) with the Adam optimizer (Kingma and Ba; 2014). The authors tested supervised variant using 5 as loss function and unsupervised variant using cross-entropy loss to show the algorithm's performance on unseen nodes.

2.5 FastGCN

A major drawback in GCN is the inclusion of neighbourhoods across layers requiring expensive recursive computations. Scalability for large and dense graphs become a bottleneck as large amount of data is involved even in mini-batch training.

These challenges were addresed by Chen et al. (2018), they propose a different approach by interpreting embedding functions as integral transforms under probabilistic measures. This forms the basis of inductive learning by using loss formulation and stochastic gradient.The loss in each convolution layer is calculated by interpreting it as integrals with respect to embedding functions. Monte Carlo approximation are used for integral evaluation to define the sample loss and gradient. They also state that approximation variance can be decreased by altering sample distribution. Chen et al. (2018)'s is primarily based on GraphSAGE (Hamilton et al.; 2017a) which learns representation by aggregating information from neighbourhoods. Chen et al. (2018) also recognise the problem of memory bottleneck in GCNs and thus propose an ad-hoc sampling methodology to restrict the size of neighbourhoods.

[1]http://snap.stanford.edu/graphsage/ppi.zip
[2]http://snap.stanford.edu/graphsage/reddit.zip
[3]https://www.tensorflow.org/

3 Methodology

3.1 Datasets

To assess the performance of our model, we have selected four widely used datasets. These datasets have been repeatedly used by researchers for node and text classification purposes in numerous deep learning based works. These datasets are well suited to serve as benchmark corpora.

- **R8 and R52 data**[4] - This is a collection of news articles from the Reuters 21578 data. There are 8 categories of articles in R8 and is split into 5,485 training and 2,189 test documents. R52 has 6,532 and 2,568 training and test documents respectively and has 52 categories.

- **Ohsumed corpus**[5] - is a bibliographic dataset of medical literature maintained by National Library of Medicine. Each piece of literature is associated with one or more categories from the 23 cardiovascular disease categories.The first set containing 20,000 abstracts are selected for consistency. As the research will focus on text classification which is single-label, documents containing multiple labels will be excluded.

- **MR**[6] - This datasets contains movie reviews for binary sentiment classification.. Each review holds only one sentiment (Pang and Lee; 2005). There are total 10,662 reviews with equal number of positive and negative reviews. For consistency, the train-test split is kept as in Tang et al. (2015)[7].

The complete statistics of the pre-processed data can be found in Table 1.

Table 1: Summary attributes of datasets

Dataset	Total	Train	Test	Vocab	#Classes	Avg Length	#Nodes/Edges
R8	7,674	5,485	2,189	7,688	8	65.72	15,362/3.165M
R52	9,100	6,532	2,568	8,892	52	69.82	17,992/3.574M
Ohsumed	7,400	3,357	4,043	14,157	23	135.82	21,557/7.456M
MR	10,662	7,108	3,554	18,764	2	20.39	29,426/1.701M

3.2 Evaluation Metrics

TP_t, FP_t, FN_t denotes the true-positives, false-positives and false-negatives respectively. t denotes $t-th$ label in label set $\mathcal{L}$.

- **Macro-F1** - This F_1 score is calculated by averaging F1 scores across all class labels. Macro-F1 score gives equal weightage to each label. Mathematically, it can be defined as $P_t = \frac{TP_t}{TP_t+FP_t}$, $R_t = \frac{TP_t}{TP_t+FN_t}$. Thus $Macro - F_1 = \frac{1}{\mathcal{L}} \sum_{t \in \mathcal{L}} \frac{2P_tR_t}{P_t+R_t}$.

[4]https://www.cs.umb.edu/∼smimarog/textmining/datasets/
[5]http://disi.unitn.it/moschitti/corpora.htm
[6]http://www.cs.cornell.edu/people/pabo/movie-review-data/
[7]https://github.com/mnqu/PTE/tree/master/data/mr

- **Weighted Average F1** - calculates weighted precision and recall scores which is then used to calculated the final F_1 score. $P = \frac{\sum_{t \in \mathcal{L}} TP_t}{TP_t + FP_t}$, $R = \frac{\sum_{t \in \mathcal{L}} TP_t}{TP_t + FN_t}$, hence Weighted Average -$F_1 = \frac{2PR}{P+R}$

- **Micro-F1** - is another F_1 metric which considers overall precision and recall for all the class labels. The micro-averaged F_1 is $P = \frac{\sum_{t \in \mathcal{L}} TP_t}{\sum_{t \in \mathcal{L}} TP_t + FP_t}$, $R = \frac{\sum_{t \in \mathcal{L}} TP_t}{\sum_{t \in \mathcal{L}} TP_t + FN_t}$. Thus $Micro - F_1 = \frac{2PR}{P+R}$.

3.3 Comparison Methods

- **TF-IDF+LR**: TF-IDF is a traditional algorithm but very effective in scoring the importance of a word for a document in a corpus. Logistic Regression is used on top of the classifier after applying TF-IDF. This technique is very simple, computationally inexpensive but very effective nonetheless. It is the most common benchmarking algorithm for text classification related tasks.

- **CNN based model**: The implementation of CNN for sentence classification proposed by Kim (2014) will be used for comparing the results with th results of our model. Kim (2014) uses 4 CNN variants for sentence classification. CNN-rand is the method in which random weights are initialised and modification takes place during training. CNN-static on th e other hand uses the widely-used pre-trained word embedding word2vec (Mikolov et al.; 2013). The embeddings from word2vec are left unchanged and rest of the model parameters are tuned during training. CNN-non-static also follows the same approach as CNN-static but it also fine tunes the word vectors during training. For our evaluation, we have considered CNN-rand and CNN-non-static because of their superior performance.

- **LSTM based model**:LSTM rose to popularity because of its inherent nature to preserves information from inputs that has already passed through the hidden state.Liu et al. (2016) applied three variants of LSTM for text classification. For the purpose of comparison, We use the LSTM model defined in Liu et al. (2016) which uses the last hidden state as the whole text representation. The model with pre-trained embeddings has been used as it had the best performance among all the LSTM based models.

- **Bi-LSTM based model**: Unidirectional LSTM has the power to only use information which has passed through its network. Bi-directional LSTMs, have become immensely popular in complex NLP related tasks which require high orderly accuracy like translation and sentiment analysis. This is because of its unique ability to use information from both past and future. Zhou et al. (2016) performed text classification with Bi-LSTM stacked with 2D max pooling enabling capture of more textual features on both time and feature vector domain. It has been used as one of our benchmarking models due to its immense performance improvements over LSTM-based text classification.

- **fastText**:is a very efficient and simple algorithm for thetext classification process (Joulin et al.; 2016). It grew in popularity it is quicker for training many orders of magnitude. Evaluation is also very fast even while retaining competitive performance with sophisticated algorithms and deep learning techniques.The authors have

stated that fastText has the ability to train on a billion words in an approximately ten minutes time-frame and classify half a million sentences that too with approximately 312K classes on standard home computer with multi-core CPUs in a minute. fastText use a very efficient combination of technique comprising of combining bag-of-words and n-grams to retain useful partial information from local sequences of words. Hierarchical softmax is used as the classifier. We use the bag-of-words model for fastText for evaluation as it has superior performance than bi-grams for our datasets.

- **Graph-CNN**: CNN when applied on graph-structured data are generally termed as Graph-CNNs. Most graph-CNNs use a similar approach on graphs by applying filters in the spectral domain. Defferrard et al. (2016) use Chebyshev filters, which was the first work to apply GCNs for the purpose of text classification. The performance result was better than the traditional CNN approaches. Other filter widely used are Spline(Bruna et al.; 2013) and Fourier filters(Henaff et al.; 2015). We only use Graph-CNN using Chebyshev filters for evaluation as it has proved to have the best performance among all of them.

- **Text-GCN**:This is one of the most popular works of using GCN for the purpose of text classificationYao et al. (2019) and this work and can be identified as an extension of their work hence we use it for bench-marking the performance of our model.

4 Design Specification

A heterogeneous graph structure is constructed which comprises of word and document nodes. This enables us to capture information on global word co-occurrences throughout the corpus as shown in Figure 2. The total number of nodes in the graph $|V|$ will be equal to the number of documents added to the total number of unique words, also called as vocabulary which was the approach taken in the work by Yao et al. (2019).

However, there are two drawbacks in their approach. It will give rise to a large number of edges due to its unrestrained nature on edge construction and thus the model proposed by Yao et al. (2019) will suffer from high memory consumption. The second drawback is use of TF-IDF score as, we cannot adjust TF-IDF score according to average length of texts or the term-frequencies of words in a corpus.

This work proposes the use of the Lucene[8] scoring system which is an enhanced version of the conventional TF-IDF scoring system, which we will refer to as L-TF-IDF in this project. Once a document is saturated with occurrences of a particular word, the impact of occurrences on the score can be controlled by the value of k. This enables us to control the contribution of term-frequency (TF) in a tunable way. This means that if we increase the value of k, the value of $\mathrm{TF}/\mathrm{TF} + k$ decreases.

But if a document is shorter than the average length, then saturation of TF should lesser than those with longer than average lengths. This tunability is achieved by introducing another parameter b. As the value of b increases from 0 to 1, the score becomes more sensitive to the length of the document. To summarise, k is the knob that controls the term saturation curve, and b controls the contribution of document length to the

[8]https://lucene.apache.org/core/

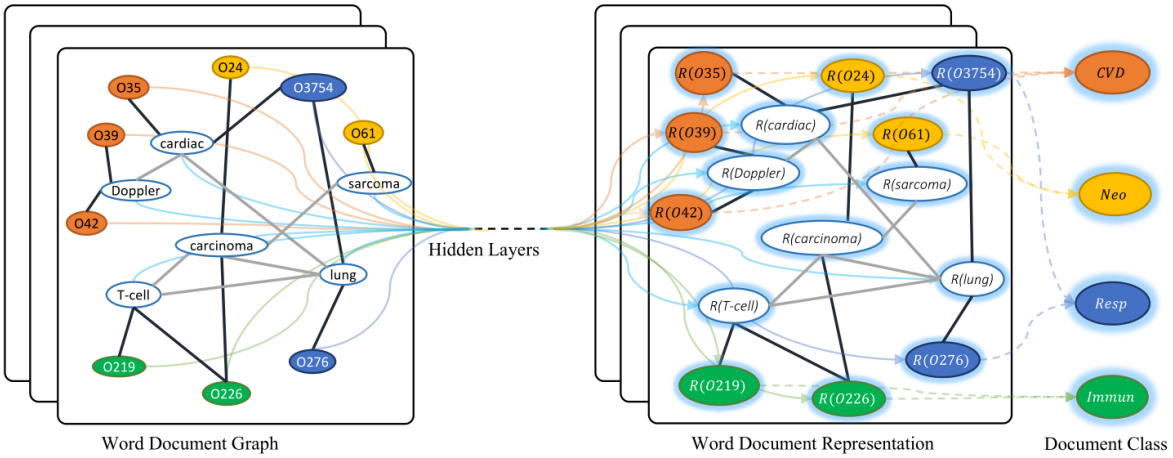

Figure 2: Example represents the Ohsumed corpus. 'O' denotes nodes containing documents and the rest are words. Thin grey edges represent word-word edges and the black bold edgees represent document-word edges. Just for example, 'CVD', 'Neo', 'Resp' and 'Immun' are document labels for classification. (Only 4 class labels shown due to space limitation)

score. To calculate the the value of L-TF-IDF we use the equations 6-8.

$$\text{L-IDF}(w) = \log(1 + (N - n + 0.5)/(n + 0.5) \tag{6}$$

$$\text{L-TF}(w) = f/(f + k(1 - b + (b \times (l/L)) \tag{7}$$

$$\text{L-TF-IDF}(w) = \text{L-TF}(w) \times \text{L-IDF}(w) \tag{8}$$

where, N is the total number of documents and n is the number of documents where word w is present. f denotes the frequency of word w in a document, l is the document length and L is the average document length of the corpus. k and b are the tunable parameters as discussed earlier.

Global word co-occurrence is calculated using a fixed size window which slides through all the documents(Yao et al.; 2019). Point-wise mutual information (PMI) will be used to calculate edge weights between word nodes as they have better performance than just using co-occurrence count. The PMI for a pair of word i, j is calculated as:

$$\text{PMI}(i, j) = \log \frac{p(i, j)}{p(i)p(j)} \tag{9}$$

$$p(i, j) = \frac{\#W(i, j)}{\#W} \tag{10}$$

$$p(i) = \frac{\#W(i)}{\#W} \tag{11}$$

where, the count of sliding windows in a corpus for word i is given by $\#W(i)$. $\#W(i, j)$ is the count of sliding windows in a corpus for both word i and j occurring in a window. The total number of windows is denoted by $\#W$. Thus the edge weight between node

i, j can be calculated as:

$$A_{ij} = \begin{cases} \text{PMI}(i,j) = \log \frac{p(i,j)}{p(i)p(j)} & i, j \text{ are words, PMI}(i,j) > 0 \\ \text{L-TF-IDF}_{ij} & i \text{ is document, } j \text{ is word} \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

High semantic correlation between words is indicated by high PMI value, whereas a negative value indicates negative correlation so no edge will be formed. After constructing the text graph, an adjacency matrix will be created which will be used as an input to a two layer GCN as in (Kipf and Welling; 2016). The node (word/document) embeddings from second layer will have the same length as labels set which will be fed to the 'softmax' classifier:

$$Z = \text{softmax}(\tilde{A} \quad \text{ReLU}(\tilde{A}XW_0)W_1) \tag{13}$$

where $\tilde{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ represents the normalised symmetric adjacency matrix. W_0 and W_1 are the weight matrix for first and second layer respectively. Now, we can see that it becomes similar to Equation 4. In Equation 13, $E_1\tilde{A}XW_0$ denotes the first layer of nodes with ReLU as the activation function. ReLU stands for Rectified Linear Unit, the function makes the output equal to the input if it is greater than 0 or else 0. The output from the first layer is fed to the second layer, defined as $E_2 = \tilde{A} \quad \text{ReLU}(\tilde{A}XW_0)W_1$ and then activated using Softmax function. Softmax takes a vector input of some length and gives a vector of same length as the output containing the normality probability distribution proportional to the exponential of the numbers in the input vector.

To perform text classification, the GCN will allow message passing between nodes which are two step away since it is a two-layer GCN. Even though there are no edges connecting document nodes to each other, the two-layer GCN will enable information exchange between document node pairs. Cross-entropy error over all document nodes will be used as the loss function:

$$\mathcal{L} = -\sum_{d \in \mathcal{Y}_D} \sum_{f=1}^{F} Y_{df} \log_e Z_{df} \tag{14}$$

where $\mathcal{Y}_D$ denotes the set of document indices and F is the number of output features which will be equal to the number of class labels.

5 Implementation

Before constructing the graph, all the texts have to be preprocessed. For consistency, data cleaning and tokenisation is carried out as Kim (2014). Stopwords are removed as per definition in NLTK[9]. Low frequency words with frequency less than 5 are removed except in MR as the documents are very short. This is because these datasets are widely used to get benchmarking scores and assess a models performance. We want to maintain the same number of words and vocabulary so that performance comparison is as relevant and admissible as possible. This also helps us to use the performance scores of different models, which have been generated in other works Validation set is extracted by random

[9]http://www.nltk.org/

Table 2: Accuracy % comparison. All models have been run 10 times and the values represent mean ± standard deviation. Results of some models have been taken directly from Yao et al. (2019).

Model	**R8**	**R52**	**Ohsumed**	**MR**
TF-IDF + LR	93.74 ± 0.00	86.95 ± 0.00	54.66 ± 0.00	74.59 ± 0.00
CNN	94.02 ± 0.57	85.37 ± 0.47	43.87 ± 0.01	74.98 ± 0.70
CNN-non-static	95.71 ± 0.52	87.59 ± 0.48	58.44 ± 1.06	**77.75 ± 0.72**
LSTM (pretrain)	93.68 ± 0.82	85.54 ± 1.13	41.13 ± 1.17	75.06 ± 0.44
Bi-LSTM (pretrain)	96.31 ±0.33	90.54 ± 0.91	49.27 ± 1.07	77.68 ± 0.86
Graph-CNN-C	96.09 ± 0.19	90.48 ± 0.86	51.10 ± 1.50	77.33 ± 0.89
fastText	96.13 ± 0.21	92.81 ± 0.09	57.70 ± 0.49	75.14 ± 0.20
Text-GCN	97.07 ± 0.10	93.56 ± 0.18	**68.36 ± 0.56**	76.74 ± 0.20
(Soumyadip, 2020)	**97.65 ± 0.05**	**94.02 ± 0.11**	67.73 ± 0.35	76.61 ± 0.27

Table 3: Mean F1 scores after running model on each dataset 10 times.

Dataset	Weighted-Average-F1	Macro-F1	Micro-F1
R8	0.9763	0.9407	0.9765
R52	0.9335	0.6881	0.9393
Ohsumed	0.6697	0.6035	0.6735
MR	75.99	75.99	75.99

selection of 10% of the training set which is also a widely used convention followed for our datasets.

The window size for PMI calculation is set to 20 and we found that varying window size did not cause significant performance variation unless set below 10. The value of k and b are set at 1.2 and 0.75 respectively for L-TF-IDF which is the default values in Lucene. The hidden layer size for the first convolution layer is set to 200. Other parameters were also tuned and learning rate was kept at 0.02, dropout rate was set at 0.5 and L_2 loss weight as 0. The maximum number of epochs have been kept at 300 with early stopping, if there is no decrease in validation loss for 10 consecutive epochs. For baseline models, accuracy figures have been taken from Yao et al. (2019).

We also drop word-word nodes so that the model can only focus on important word co-occurrence information. We run our models with different node dropout rates to test if it has any significant effect on performance. We test our model with 5 node dropout rates ranging from 0.25 to 0.95. If the value is set at 0.75, it means that only word-word edges with values for more than 75 percentile will be retained, rest will be dropped.

6 Evaluation and Discussion

Our model achieves competitive performance and outperforms TextGCN on three datasets except Ohsumed. While we do not get the best performance on MR, but we get a marginally better result then TextGCN. Table 3 lists all the different types of F1 socre for all datasets, since MR is a case of binary classification, the F1 scores are same. Table 2 lists the performance of our model when compared with other models as

mentioned in Section 3.3. However, our model fails to outperform CNN based models on MR. This might be due to the failure in retaining word order information in graphs which are important for sentiment classification. Another reason might be the presence of fewer edges in MR than other text graphs as documents are small giving rise to lesser number of sliding windows. The two main reasons why it performs better than TextGCN are :

- Use of L-TF-IDF is a better scoring mechanism for scoring document-word edges than the traditional TF-IDF. It takes both word frequency saturation and document length into consideration for generating the scores.
- Reducing redundant edges between words as low PMI scores do not contribute to the model significantly hence the model does not generalise well. Retaining word-word edges helps the model to focus only on important word co-occurrences.

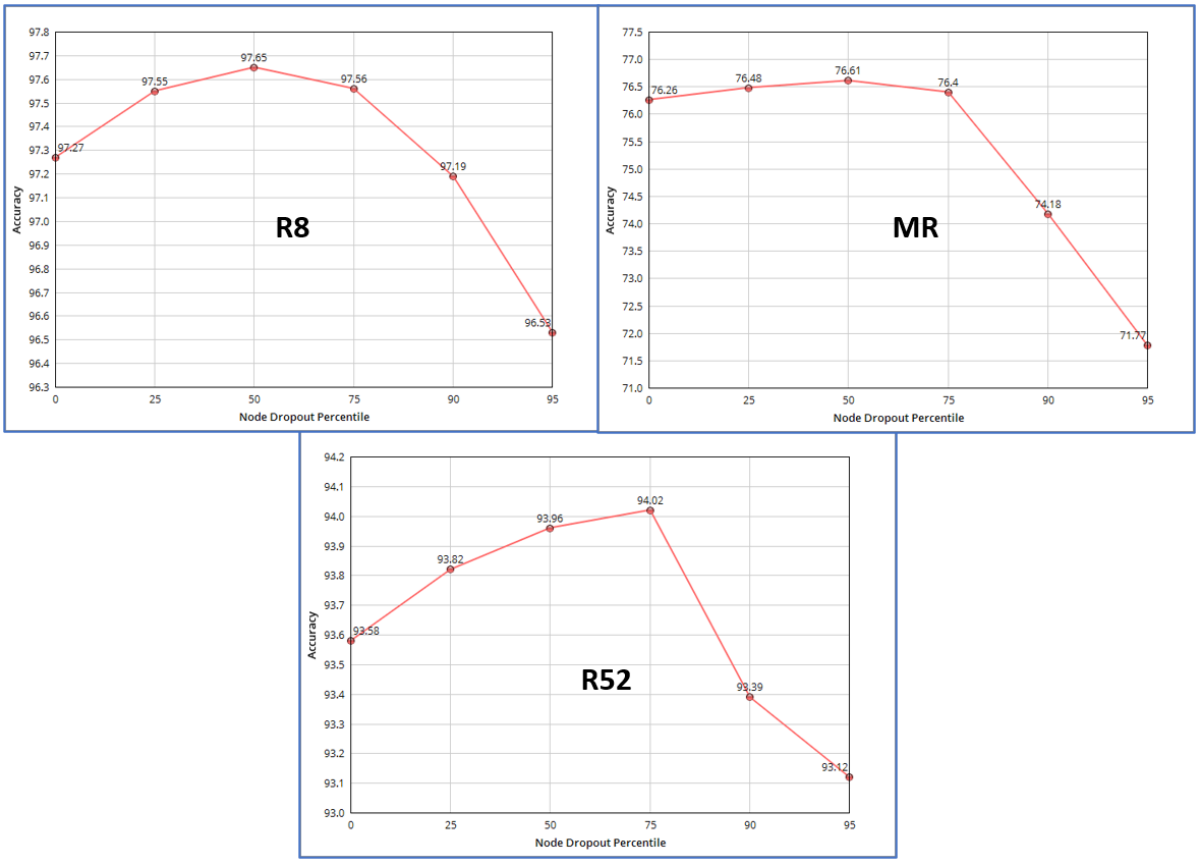

Figure 3: Test Accuracy with different Node Dropout Rates

We also find that reducing the number of word-word edges have a positive impact on model accuracy. Figure 3 depicts the performance on three datasets as we drop the edges with low PMI values, accuracy increases gradually and then falls significantly. The general trend we observed that our model gives the best performance when we retain the top 50-75% of word-word edges. This has two advantages, one is that the size of the graph reduces significantly so generalises better, secondly model convergence takes place in lesser number of epochs. The average number of epochs with different dropout rates can be seen in Figure 4. The mean number of epochs have been reported after running

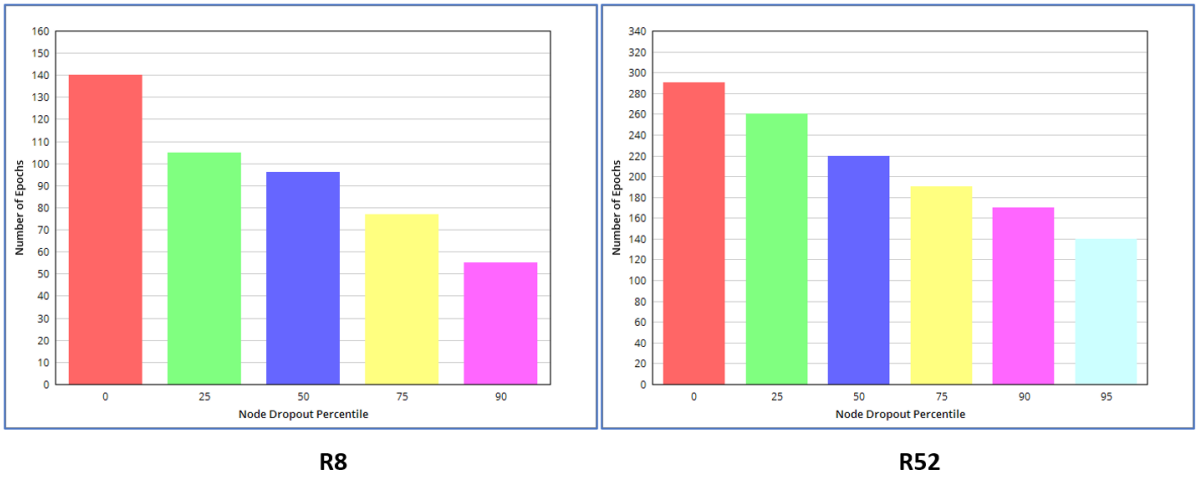

Figure 4: Change in number of Training Epochs with different Node Dropout Rates

them at least 10 times. We can see that graphs with lesser word-word edges will give better performance and will converge faster.

7 Conclusion and Future Work

By studying the result from the experiments, our model can achieve very good text classification results and thus learn predictive word and document embeddings. But this approach has a major limitation as the model is transductive in nature, and will not be able to generate prediction on unseen nodes.

In this study, we improve upon the previously proposed TextGCN, by building document-word edges with a novel scoring mechanism which can be tuned as per the nature of the corpus. Our approach also reduces graph size while increasing accuracy and reducing training time, enabling the use of larger corpus of texts for training.

This provides the flexibility to use the model on documents of varying lengths and thus can also be used on smaller textual units like paragraphs, abstracts and summaries as well as larger texts like research papers and many others. However, over-smoothing of GCN limits the abstraction ability which is required for deeper networks, so increasing stacks of layer will not have any significant benefit. An interesting future direction will be application of other GNN models for text classification with inductive abilities. However, this approach is not limited only to text classification, it can also be applied to other NLP related tasks in the future.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I. J., Harp, A., Irving, G., Isard, M., Jia, Y., Józefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D. G., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P. A., Vanhoucke, V., Vasudevan, V., Viégas, F. B., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems, *CoRR*

abs/1603.04467.
URL: *http://arxiv.org/abs/1603.04467*

Aggarwal, C. C. and Zhai, C. (2012). *A Survey of Text Classification Algorithms*, Springer US, Boston, MA, pp. 163–222.
URL: *https://doi.org/10.1007/978-1-4614-3223-4*$_6$

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y. and Pascanu, R. (2018). Relational inductive biases, deep learning, and graph networks.

Bhagat, S., Cormode, G. and Muthukrishnan, S. (2011). Node classification in social networks, *CoRR* **abs/1101.3291**.
URL: *http://arxiv.org/abs/1101.3291*

Bruna, J., Zaremba, W., Szlam, A. and LeCun, Y. (2013). Spectral networks and locally connected networks on graphs, *arXiv preprint arXiv:1312.6203* .

Cai, H., Zheng, V. W. and Chang, K. C. (2017). A comprehensive survey of graph embedding: Problems, techniques and applications, *CoRR* **abs/1709.07604**.
URL: *http://arxiv.org/abs/1709.07604*

Chen, J., Ma, T. and Xiao, C. (2018). Fastgcn: Fast learning with graph convolutional networks via importance sampling, *CoRR* **abs/1801.10247**.
URL: *http://arxiv.org/abs/1801.10247*

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H. and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation, *CoRR* **abs/1406.1078**.
URL: *http://arxiv.org/abs/1406.1078*

Defferrard, M., Bresson, X. and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering, *CoRR* **abs/1606.09375**.
URL: *http://arxiv.org/abs/1606.09375*

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. and Dahl, G. E. (2017). Neural message passing for quantum chemistry, *CoRR* **abs/1704.01212**.
URL: *http://arxiv.org/abs/1704.01212*

Hamilton, W. L., Ying, R. and Leskovec, J. (2017a). Inductive representation learning on large graphs, *CoRR* **abs/1706.02216**.
URL: *http://arxiv.org/abs/1706.02216*

Hamilton, W. L., Ying, R. and Leskovec, J. (2017b). Representation learning on graphs: Methods and applications, *CoRR* **abs/1709.05584**.
URL: *http://arxiv.org/abs/1709.05584*

Henaff, M., Bruna, J. and LeCun, Y. (2015). Deep convolutional networks on graph-structured data, *arXiv preprint arXiv:1506.05163* .

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory, *Neural Comput.* **9**(8): 1735–1780.
URL: *https://doi.org/10.1162/neco.1997.9.8.1735*

Jindal, N. and Liu, B. (2007). Review spam detection, *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, Association for Computing Machinery, New York, NY, USA, p. 1189–1190.
URL: *https://doi.org/10.1145/1242572.1242759*

Joulin, A., Grave, E., Bojanowski, P. and Mikolov, T. (2016). Bag of tricks for efficient text classification, *CoRR* **abs/1607.01759**.
URL: *http://arxiv.org/abs/1607.01759*

Kim, Y. (2014). Convolutional neural networks for sentence classification, *CoRR* **abs/1408.5882**.
URL: *http://arxiv.org/abs/1408.5882*

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* .

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks, *CoRR* **abs/1609.02907**.
URL: *http://arxiv.org/abs/1609.02907*

Li, Y., Tarlow, D., Brockschmidt, M. and Zemel, R. (2015). Gated graph sequence neural networks, *arXiv preprint arXiv:1511.05493* .

Liben-Nowell, D. and Kleinberg, J. (2007). The link-prediction problem for social networks, *J. Am. Soc. Inf. Sci. Technol.* **58**(7): 1019–1031.

Liu, P., Qiu, X. and Huang, X. (2016). Recurrent neural network for text classification with multi-task learning, *CoRR* **abs/1605.05101**.
URL: *http://arxiv.org/abs/1605.05101*

Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013). Efficient estimation of word representations in vector space, *in* Y. Bengio and Y. LeCun (eds), *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.
URL: *http://arxiv.org/abs/1301.3781*

Pang, B. and Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales, *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, Association for Computational Linguistics, USA, p. 115–124.
URL: *https://doi.org/10.3115/1219840.1219855*

Peng, H., Li, J., He, Y., Liu, Y., Bao, M., Wang, L., Song, Y. and Yang, Q. (2018). Large-scale hierarchical text classification with recursively regularized deep graph-cnn, *Proceedings of the 2018 World Wide Web Conference*, WWW '18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, p. 1063–1072.
URL: *https://doi.org/10.1145/3178876.3186005*

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. and Monfardini, G. (2008). The graph neural network model, *IEEE Transactions on Neural Networks* **20**(1): 61–80.

Tang, J., Qu, M. and Mei, Q. (2015). Pte, *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15* . **URL:** *http://dx.doi.org/10.1145/2783258.2783307*

Vishwanathan, S. V. N., Borgwardt, K. M., Kondor, I. R. and Schraudolph, N. N. (2008). Graph kernels, *CoRR* **abs/0807.0093**. **URL:** *http://arxiv.org/abs/0807.0093*

Yao, L., Mao, C. and Luo, Y. (2019). Graph convolutional networks for text classification, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, pp. 7370–7377.

Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H. and Xu, B. (2016). Text classification improved by integrating bidirectional lstm with two-dimensional max pooling, *arXiv preprint arXiv:1611.06639* .