Text Classification Using Document-Relational Graph Convolutional Networks

CHONGYI LIU\textsuperscript{1}, XIANGYU WANG\textsuperscript{2,3}, AND HONGLEI XU\textsuperscript{1}, (Senior Member, IEEE)

\textsuperscript{1}School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Perth, WA 6845, Australia
\textsuperscript{2}School of Civil Engineering and Architecture, East China Jiaotong University, Nanchang 330013, China
\textsuperscript{3}Australian Joint Research Centre for Building Information Modeling, Curtin University, Perth, WA 6102, Australia

Corresponding author: Xiangyu Wang (xiangyu.wang@curtin.edu.au)

\begin{abstract}
Graph Convolutional Networks (GCNs) have received considerable attention in the field of artificial machine intelligence (AMI) and natural language processing research because they can build more sophisticated accompanying graph structures than traditional neural networks for feature engineering. Graph is used as feature in neural network because it is easy to find relations among nodes. In text classification applications, a GCN can create complex and rich relation-based adjacent matrix graphs as features to be trained. The existing methods, on the other hand, only generated adjacent matrix graphs in GCN at the word-document and word-word levels as features. In this paper, we propose a document-relational GCN to achieve superior accuracy in text classification by adding cumulative term frequency-inverse document frequency (TF-IDF) document-document relations as features. The performance of the proposed method is evaluated using five popular benchmark databases. In addition, different hidden nodes and proportions of document-document features are tested to achieve an advantageous outcome.
\end{abstract}

\begin{IEEEkeywords}
NLP, document-document relation, graph convolutional network, text classification.
\end{IEEEkeywords}

\section{Introduction}

Text classification is one of the most important tasks in deep learning problems. It recognises and classifies documents for different applications, such as sentiment classification [1], semantic segmentation [2], rule text based recommendation [3] and news classification [4]. Many researchers have attempted to solve text classification problems in different text ranges, including at document [5] and sentence levels [6].

To achieve better classification performance, significant efforts have been made to accelerate the process of model development from machine learning methods [7], [8] to deep learning methods [2], [9], [10]. And the development of deep learning computing efficiency [11], [12] techniques have been assisting running time of deep learning much faster. Compared to other methods for text classification, deep learning methods have achieved better effective and efficient performance owing to the rapid development of computer hardware and deep learning research.

Topic classification is an important text classification application that uses machine learning [13], [14], [15], and deep learning [16], [17], [18] methods. For example, in road construction documents, a text classification program needs to be developed to categorise the regulatory compliance language. Document inspection for road construction regulatory compliance begins with text classification, as described in [19] and [20]. As different forms of code compliance documents have varied roles in terms of topics, topic classification is an appropriate solution for code compliance text classification.

News classification is one of the most popular and widely used applications in the field of topic classifications. Every minute, there are so many news releases that individuals cannot keep up with them. It is possible to classify news in a manner that allows humans to comprehend it more quickly and correctly. On Facebook and Twitter, for example, a plethora of news has been disseminated. The news should be classified into their own categories [21]. As a result, people and social media firms should be responsible to structure news data. To comprehend data using the Internet structural approach, Trieu et al. [22] developed a word vector technique for news...
classification based on Twitter data. Furthermore, false news accounts for a significant proportion of online news releases. Recently, fake news classification has improved, which can help consumers detect false news [23], [24], [25]. Furthermore, news classification can assist healthcare companies in extracting relevant data from social media news [26]. Therefore, news classification is worth investigating to improve its performance effectively and efficiently.

In recent years, graph networks have received considerable interest in deep learning research [27], and the use of graph networks for text classification has shown excellent performance [28], [29], [30]. Because many applications may build huge amount of data in a structured fashion, graphs have the benefit of storing rich relations between nodes [31]. Then, this type of structural data can be used as features in a deep learning model to obtain good results for a realistic relation. The development of deep learning networks has been assisted the development of GCN, such as Convolutional Neural Network (CNN) [32], [33]. To obtain better citation classification performance in an effective and efficient manner, Kipf and Welling [34] developed a GCN model that can build relations of objects throughout the entire dataset. The GCN model was then used to classify documents by Yao et al. [35], who achieved improved performance in five distinct benchmark datasets, including a news dataset. However, in their GCN model, they only generated the feature graph using document-word relations. Tang et al. [36] constructed a text classification integration model based on a GCN and achieved competitive results. A document-document link was added into feature graph to seek a better result for text classification.

Till now, there have been a few techniques that can be used to enhance graph features. Meta-path is a type of text classification characteristic that can be used to create document relations. Wang et al. [37] utilised a meta-path approach to build linkages between documents, and a machine learning classifier produced superior classification results in two topic-based datasets. The meta-path approach was used by Ding et al. [38] and was applied to enrich the knowledge graph for the entire dataset. In summary, it can be concluded that the link relation between the generated documents in the dataset has better performance on topic sensitive documents. Therefore, text classification effect can be improved from the perspective of feature engineering to enhance the knowledge network in the entire dataset. In conclusion, the link relations between the documents produced in the dataset performs better on the topic sensitive documents to enhance the text classification.

Working on feature engineering is an effective way to enhance the text classification performance. Data are the basis of feature engineering to achieve good performance, and almost all models require data to drive [39], [40]. There are two methods of feature engineering which are searching relations inside a dataset, such as graph networks [27], [28], [29], [30] and increasing size of one dataset, such as data augmentation [41], [42]. Zhang et al. [43] found an effective way to perform data augmentation which simply replaced words or phrases in the dataset by their synonym ones.

We propose to add document-document link weights to the data graph to improve text classification performance, inspired by the notion of adding visual information to features through meta-paths in terms of feature engineering. Also, we compare our method to text graph networks and synonym data augmentation. The contributions of this study are as follows:

- We generate a cumulative term frequency-inverse document frequency (TF-IDF) value for document-document relations to achieve higher text classification accuracy.
- We tuned an appropriate hyperparameter value in the document-relational GCN model to obtain the best text classification performance.

The remaining parts of this paper are organised as follows. We present previous GCN models in Section 2. The proposed cumulative TF-IDF document-relational GCN model is established in Section 3. Section 4 presents the experimental results and their analysis. Finally, the conclusions of this paper and the prospects for future work are presented in Section 5.

II. RELATED WORK
In this section, we briefly describe previous works on document classification for both graph generation and GCN model construction.

A. GRAPH GENERATION
Yao et al. [35] constructed a large heterogeneous graph for an entire dataset. The total number of nodes in a corpus was equal to the number of documents and unique words. Thus, the dimension of the graph adjacent matrix is \((\text{documents} + \text{uniquewords}) \times (\text{documents} + \text{uniquewords})\). Furthermore, in the graph, there are two types of edge relations which are document-word and word-word. TF-IDF determines the values of document-word edges:

\[
TF_{i,j} = \frac{N_{i,j}}{\sum_{k \neq j} N_{k,j}} \tag{1}
\]

\[
IDF_j = \log \left\lceil \frac{|D|}{|\{ j : t_i \in d_j \}| + 1} \right\rceil
\]

\[
TFIDF_{i,j} = TF_{i,j} \times IDF_j
\]

where \(TF_{i,j}\) is frequency of the term (word) \(i\) in the document \(j\), and \(IDF_j\) is the logarithmic number of the total documents divided by the number of the documents that contain the word \(i\).

Word-word relations are represented by another sort of edges. To deal with these relations, traditional approaches are to count word occurrences in a confined context region, which is known as word co-occurrence statistics [44], [45], [46]. The word-word link weight value in the constructed graph is calculated using the point-wise mutual information (PMI) method, which is based on the word co-occurrence
B. GCN MODEL

The document graph in the previous subsection was used as a feature in the GCN model for further training and testing. Kipf and Welling [34] developed a GCN model expressed by a multilayer neural network that directly operates on graphs as features. Let $G = (N, E)$ denote the document graph, where $N$ and $E$ are the nodes and edges, respectively. The number of nodes is the sum of the number of documents and unique vocabulary terms. Adjacent matrix $A$ is used to represent the graph. Then, the adjacent matrix $A$ must be normalised as $\tilde{A} = D^{-0.5}A D^{-0.5}$, where $D$ is the degree matrix of $A$ and $D_{ii} = \sum_j A_{ij}$. There is a feature matrix $X$ followed by a normalised adjacent matrix $\tilde{A}$. In the GCN model for text classification, the feature matrix is usually set to be an identity matrix as $X = I$, so $X$ is not shown in the future equations. The normalised adjacent matrix $\tilde{A}$ is a feature of the model. The forward propagation equations for the text classification version are as follows:

$$L_1 = r(\tilde{A}W_0)$$
$$L_{j+1} = r(\tilde{A}L_j W_j)$$
$$\tilde{D} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$$
$$D_{ii} = \sum_j \tilde{A}_{ij}$$
$$r(x) = \max(0, x)$$

(3)

where $L$ is the hidden or result layer matrix, $j$ is the layer number, $\tilde{A}$ is the normalised adjacent matrix of the graph, and $r(x)$ is an activation function of the neural network, called ReLU.

III. PROPOSED METHOD

In this section, we develop cumulative TF-IDF edges to connect document-document nodes, which differs from the graph that simply uses word-document relations. We generate document relations by adding the TF-IDF values multiplication of the same words.

$$A_{ij}' = \log(A(row_i) \cdot A(column_j))$$

(4)

where $i, j$ are document indices, $A(row_i)$ is the row vector for document $i$, and $A(column_j)$ is the column vector for document $j$. The multiplication of document vectors $i$ and $j$ shows that the TF-IDF values of two documents with the same vocabulary are multiplied and added together. Then we only save the edges that are greater than or equal to 3 because only a large value of document-document relation has a positive effect on text classification performance. Other edge relations of the graph, such as word-word and document-word relations, are identical to those in [35]. Thus, the total adjacent matrix can be defined as follows:

$$A_{ij}' = \begin{cases} 
\log(A(row_i) \cdot A(column_j)) & \text{if } i, j \text{ are documents} \\
PMI(i, j) & \text{if } i, j \text{ are words} \\
TF - IDF_{i,j} & \text{if } i = j \\
0 & \text{otherwise}
\end{cases}$$

Document-document, word-document, and word-word relational values comprise adjacency matrix $A_{ij}'$. The document-document relation is obtained from the equation (4). The word-document relation is the TF-IDF value from the equation (1). The word-word relation is PMI value from the equation (2). The adjacent matrix $A_{ij}'$ has a square shape. The document and vocabulary together constitute the size of the adjacency matrix. The details of how to construct the neighbouring matrix of the graph are shown in Pseudo Code 1.

The adjacent matrix of graph $A_{ij}'$ is then sent to the GCN model in the next step. A previous study ([34], [35], [47]) shows that a GCN model with two layers performs better than that with one layer, whereas a GCN model with more than two layers does not perform better than that with two layers. Thus, we use a two-layer GCN model to train and test our text classification model in this study. Let

$$Z = \softmax(\tilde{A} \cdot r(\tilde{A}W_0)W_1)$$
$$\softmax(x_i) = \frac{\exp(x_i)}{\sum_i \exp(x_i)}$$
$$\tilde{A} = D^{-1/2}A D^{-1/2}$$
$$D_{ii} = \sum_j \tilde{A}_{ij}$$

(5)

where $\tilde{A}$ is the same as that in (3), $W_0$ and $W_1$ are two weight matrices to learn, and $\softmax(\cdot)$ is a probability normalisation function that can convert the result vector elements into a probability distribution. For each vector, the number of elements is the number of document classes that must be classified. Thus, the total of all elements in a vector after the $\softmax$ computation is 1, and the highest value position represents the text classification class. Then, the cross-entropy loss function is applied to calculate the error distance between
The loss function has two weights that must be learned. The GCN model does forward propagation and calculates the loss function each time. The derivatives of two weights ($W_0$ and $W_1$) are then determined. To reduce loss value, the $W_0$ and $W_1$ are updated in the opposite direction as the derivatives. The optimal weights ($W_0$ and $W_1$) are discovered, and it can now approach the best document classification prediction using the present model and inputs.

IV. EXPERIMENT

In this section, for text classification accuracy, we compare our document-document updated GCN to the original text GCN and augmented data with text GCN. Synonym data augmentation [43] is used to enlarge the five datasets which is double the size of each dataset. Furthermore, we verify the accuracy of our model by filtering alternative maximum values of document-document graph weights in adjacent matrix of the graph. In the updated document GCN model, we evaluate the performance for different numbers of hidden layer dimensions.

A. DATASETS

In the text classification field, we consider five benchmark datasets.

- **20ng**. Dataset of 20 newsgroup records and 20 different types of news. There are a total of 18846 documents and 42757 vocabulary items. 11314 documents are used for training and 7532 for testing.
- **R8** and **R52**. Two subsets of Reuters data. There are 8 and 20 classes in each. R8 is equipped with 5485 train documents, 2189 test documents, and 7688 vocabulary items. R52 has 6532 train documents, 2568 test documents, and 8992 vocabulary items in its database.
- **Ohsumed** [48]. It comes from the “MEDLINE10” medical database, which contains titles and abstracts from 270 medical journal articles published between 1987 and 1992. Diseases are divided into 23 categories.
- **MR**. Dataset of movie reviews with only two sentiment classes. There are 10662 documents in total, with half of them being positive reviews and the other half being negative reviews.

The details of the datasets are summarised in Table 1. The number of training nodes is the sum of the number of true training documents and the number of valid training documents.

| Dataset  | 20ng  | R8    | R52   | Ohsumed | MR   |
|----------|-------|-------|-------|---------|------|
| # Actual train | 10183 | 4937  | 5879  | 3022    | 6398 |
| # Valid | 1131  | 548   | 653   | 335     | 710  |
| # Words | 42757 | 7688  | 8892  | 14157   | 18764|
| # Test  | 7532  | 2189  | 2568  | 4043    | 3554 |
| # Total | 61603 | 15362 | 17992 | 21557   | 29426|
| # Classes | 20    | 8     | 52    | 23      | 2    |

The model forward propagation result $Z$ and the target label:

$$Loss = - \sum_{d \in Yd} \sum_{c=1}^{C} Y_{dc} \log Z_{dc}$$  \hspace{1cm} (6)$$

where $Y_d$ is the label document index, and $C$ is the number of document class dimensions of the outcome feature. $Y_{dc}$ is a vector label and $Z_{dc}$ is a vector result. They represent the target label and predicted result of a document, respectively.

Pseudo Code 1 Build an Adjacent Matrix of the Graph

```python
windows ← empty list;
window_size ← 20;
for doc in documents do
    windows.add (every 20 words);
    for word in doc do
        TF[doc, word] ← number of the word in the doc;
        IDF[doc, word] ← log(document number/documents with the word);
        adj[doc, word] ← TF[doc, word] * IDF[doc, word];
    end for
end for
word_window_freq ← empty list;
for window in windows do
    for word in window do
        word_window_freq[word] ← word_window_freq[word] + 1;
    end for
end for
word_pair_count ← empty list;
for window in windows do
    for wordPair in window do
        word_pair_count[wordPair] ← word_pair_count[wordPair] + 1;
    end for
end for
num_window ← number of windows;
for key in word_pair_count do
    i ← first word in key;
    j ← second word in key;
    count ← word_pair_count[key];
    word_freq_i ← word_window_freq[i];
    word_freq_j ← word_window_freq[j];
    pmi ← log(count / num_window) / (word_freq_i * word_freq_j / (num_i * num_j));
    adj[i, j] ← pmi;
end for
for each word-doc relation in adj do
    adj[doc, doc] ← adj[doc-words row vector],
    adj[words-doc column vector];
end for
```
documents. The training documents are divided into two categories: actual training and validation. 90% of training documents are for actual training, while 10% are for validation.

B. EXPERIMENT PARAMETERS
The PyTorch library is one of the most widely used deep learning frameworks. Both GCN models (document-relational GCN and original text GCN model) are run on the PyTorch deep learning framework. In the training model, the weights for both models are updated using the Adam optimiser [49]. As previously stated, the drop out rate is 0.5, and the GCN model has two layers.

C. HIDDEN LAYERS
Yao et al. [35] chose 200 hidden layer dimensions of the GCN model in their experiment. According to our research, it is difficult to obtain good test performance with as few as 200 hidden layer dimensions, according to our research. Consequently, a number of hidden layer dimensions are investigated, as shown in Fig. 1. When we used 1200 and 1500 hidden layers in our document-relational GCN model, we found that we could achieve better test accuracy.

D. EXPERIMENT RESULTS
The test accuracy comparison for our document-relational GCN and the original model without document-document relation edges is presented in Table 2. Our new model is even more accurate than the original GCN model, which performed admirably in the previous experiment [35]. In four out of five datasets, our document-relational GCN model improves 0.2%-1% accuracy to text GCN. For the 20ng dataset, the model exhibits a noticeable performance improvement (1%) in terms of test accuracy compared to other datasets. Because of the new document-document relation in the graph, it performs better for the 20ng dataset than the others. The synonym augmented GCN performs better in the R52 and Ohsumed datasets. TF-IDF stores important words that can represent document topics, and contains information about document-document relations. It also increases the weight of topic words in documents on the same subject. Because the same news category belongs to the same topic, 20ng is both a news and topic classification dataset. In a hyperparameter experiment with hidden layers, overall, there is an increasing tendency of accuracy with an increasing number of hidden layers in our document-relational GCN model. The most accurate results are obtained with the number of hidden layers ranging from 1200 to 1500 in the four datasets. We can hardly obtain good results in both test accuracy and expected hidden layers for the Ohsumed dataset because using words link to stand for document is not sensitive to

| Table 2. Comparison test accuracy among document-relational GCN, text GCN and synonym augmented text GCN. |
|----------------------------------------------------------------------------------------------------------|
| Dataset | Doc-Relational GCN | Text GCN [35] | Synonym aug GCN |
|---------|--------------------|---------------|-----------------|
| 20ng    | 86.8%              | 85.81%        | 85.7%           |
| R8      | 96.96%             | 96.8%         | 96.3%           |
| R52     | 93.19%             | 92.47%        | 93.2%           |
| Ohsumed | 65.9%              | 66.99%        | 68.0%           |
| MR      | 76.57%             | 76.18%        | 73.8%           |

FIGURE 1. Relations Between the Number of Hidden Layers and Test Accuracy.
medical abstract documents. Overall, the cumulative TF-IDF document-document relation in our document-relational GCN can achieve much higher test accuracy in most datasets related to topic document classification, and a specific figure of the hidden layer hyperparameter can be used for model parameter tuning.

V. CONCLUSION

We have introduced a new document-relational GCN in this study, which provides a cumulative TF-IDF document-document relation for text classification. By testing its performance on five benchmark datasets, we discovered that with document-document relation edges in the adjacent matrix, the document-relational GCN model can improve the overall accuracy compared to the original text GCN and the synonym augmented GCN. Synonym augmented GCN has higher accuracy in the medical dataset. In addition, we experimented with different numbers of hidden layers to determine the optimal parameters of the model. We can see that excessive hidden layers do not help the original text GCN and data-augmented GCN have higher accuracy. However, in the document-relational GCN model, a large number of hidden layers have a better ability to store text and graph features, which helps text classification. Our future research plan is to determine the best percentage of document-document edges in the adjacent matrix, to use when filtering useful or important document relation features. We will also use a more sophisticated algorithm to calculate document-document relations, as well as filtering document-word or word-word relations to simplify the graph.

REFERENCES

[1] M. Kanakaraj and R. M. R. Guddeti, “NLP based sentiment analysis on Twitter data using ensemble classifiers,” in Proc. 3rd Int. Conf. Signal Process., Commun. Netw. (ICSCN), Mar. 2015, pp. 1–5.
[2] W. Zhou, J. Liu, J. Lei, L. Yu, and J.-H. Hwang, “GMNet: Graded-feature multilabel-learning network for RGB-thermal urban scene semantic segmentation,” IEEE Trans. Image Process., vol. 30, pp. 7790–7802, 2021.
[3] Z. Wu, C. Li, J. Cao, and Y. Ge, “On scalability of association-rule-based recommendation: A unified distributed-computing framework,” ACM Trans. Web, vol. 14, no. 3, pp. 1–21, 2020.
[4] C. Li, G. Zhan, and Z. Li, “News text classification based on improved Bi-LSTM-CNN,” in Proc. 9th Int. Conf. Inf. Technol. Med. Educ. (ITMEd), Oct. 2018, pp. 890–893.
[5] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, “Hierarchical attention networks for document classification,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol., 2016, pp. 1480–1489.
[6] T. Chen, R. Xu, Y. He, and X. Wang, “Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN,” Exp. Syst. Appl., vol. 72, pp. 221–230, Apr. 2017.
[7] F. Colas and P. Brazdil, “Comparison of SVM and some other classification algorithms in text classification tasks,” in Proc. IFIP Int. Conf. Artif. Intell. Theory Pract. Springer, 2006, pp. 169–178.
[8] G. Sun, Y. Cong, Q. Wang, B. Zhong, and Y. Fu, “Representative task self-selection for flexible clustered lifelong learning,” IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 4, pp. 1467–1481, Apr. 2022.
[9] Y. Kim, “Convolutional neural networks for sentence classification,” 2014, arXiv:1408.5882.
[10] H. Liu, J. Liu, S. Hou, T. Tao, and J. Han, “Perception consistency ultrasound image super-resolution via self-supervised CycleGAN,” Neural Comput. Appl., pp. 1–11, Jan. 2021.
[11] Z. Wu, A. Song, J. Cao, J. Luo, and L. Zhang, “Efficiently translating complex SQL query to MapReduce jobflow on cloud,” IEEE Trans. Cloud Comput., vol. 8, no. 2, pp. 508–517, Apr. 2020.
I. Introduction

II. Related Work

III. Methodology

IV. Experimental Setup

V. Results

VI. Conclusion

References

Appendices