Text Classification Using a Graph Based on Relationships Between Documents

Hiromu Nakajima
Graduate School of Sci. and Eng., Ibaraki Univ./ 4-12-1 Nakanarisawacho, Hitachi City, Ibaraki Prefecture, 316-8511, Japan
22nm738g@vc.ibaraki.ac.jp

Minoru Sasaki
Graduate School of Sci. and Eng., Ibaraki Univ./ 4-12-1 Nakanarisawacho, Hitachi City, Ibaraki Prefecture, 316-8511, Japan
minoru.sasaki.01@vc.ibaraki.ac.jp

Abstract

Text classification, which determines the genre of a document based on cues such as the co-occurrence of words and their frequency of occurrence, has been studied in various approaches to date. Conventional text classification methods using graph-structured data express relationships between words and between words and documents in the form of weights of edges between each node. Then, the graph is input to a graph neural network for learning. However, conventional methods do not represent the relationship between documents on the graph, and thus cannot directly consider the relationship between documents. Therefore, we propose a text classification method using the graph considers the relationships among documents. This method directly expresses the relationship between documents by adding the similarity of documents as weights of edges between document nodes to the graph of the conventional method. The constructed graph is then input to a graph convolutional neural network for learning. We conducted experiments using five English corpus (20NG, R52, R8, Ohsumed, and MR) to evaluate proposed method. The results show that the proposed method improves accuracy compared to the conventional method and that the use of relationships among document nodes is effective. Experimental results also show that the proposed method is particularly effective on datasets with relatively long documents.

1 Introduction

Text classification is the task of estimating an appropriate label for a given document from a set of predefined labels. This task is one of the fundamental problems in natural language processing. This technique has been applied in the real world to automate the task of document classification by humans. Many researchers are interested in developing applications that leverage text classification methods such as Junk mail classification, topic labeling and sentiment analysis.

In the past few years, convolutional neural networks that can take advantage of graph structural information have been used in solving text classification problems. TextGCN (Yao et al., 2019) is one of the examples of graph-based text classification methods. In TextGCN, word nodes and document nodes are represented on the same graph, which is input to GCN for learning. VGCN-BERT (Lu et al., 2020) trained by constructing a graph based on word co-occurrence information and word embedding representation of BERT and inputting the graph to GCN. RoBERTaGCN (Yuxiao et al., 2021) is a text classification method that combines the benefits of GCN's transductive learning with the knowledge gained from BERT's large-scale prior learning using large amounts of unlabeled data. This method boasts the best performance among existing methods for text classification with four datasets: 20NG, R8, Ohsumed, and MR. The graphs in these text classification methods use word-to-word and word-to-document relationships. However, conventional graph-based text classification methods do not use the relationship between documents. Therefore, we thought that accuracy could be improved by using the relationship between documents.

In this study, we aimed to solve the problem of Conventional graph-based text classification methods described above paragraph by adding relations between documents to the edges between document nodes, and to improve the classification

performance of RoBERTaGCN. Specifically, we input each document into the BERT model and obtain the vector of '[CLS]' token in final hidden layer. Then, we calculated the cosine similarity of these '[CLS]' token vectors of each document and added the cosine similarity that exceeded a predetermined threshold as a weight between document nodes. Then, we can create an effective graph structure that considers the relation between document nodes to improve the accuracy in each dataset of RoBERTaGCN. In addition, we consider that topic drift is less likely to occur because document information can be propagated without going through word nodes.

2 Related Work

Graph neural networks (Scarselli et al., 2008) are neural networks that learn relationships between graph nodes via the edges that connect them. There are several types of GNNs. The graph convolutional networks (Kipf and Welling, 2016a) takes a graph as input and learns the relationship between the nodes of interest and their neighbors through convolutional computation using weights assigned to the edges between the nodes. The graph autoencoder (Kipf and Welling., 2016b) is the autoencoder that extracts important features by dimensionally collapsing the input data. In Graph Attention Network (Velickovi et al., 2017), the weights of edges between nodes and the coefficients representing the importance of neighboring nodes are used to extract important features. GNNs have been used in a wide range of tasks in the field of machine learning, such as relation extraction, text generation, machine translation and question answering. In the field of machine learning, GNNs have been used in a wide range of tasks and have demonstrated high performance. The success of GNNs in these wide range of tasks has motivated us to study text classification methods using GNNs. In TextGCN (Yao et al., 2019), document nodes and word nodes are represented on the same graph (heterogeneous graph), which is input to the GCN for training. Recently, there has been a lot of research on text classification methods that combine large scale pre-training models such as BERT with GNNs. VGCN-BERT (Lu et al., 2020) trained by constructing a graph based on word co-occurrence information and word embedding representation of BERT and inputting the graph to GCN. In BertGCN (Yuxiao et al., 2021), the heterogeneous graph of words and documents is constructed based on word co-occurrence information and BERT's document embedding representation, and the graph is input to GCN for learning. A detailed description of BertGCN is given in the next chapter.

3 RoBERTaGCN

BertGCN is a text classification method that combines the transductive learning of GCN with the knowledge gained from large-scale pre-training using large amounts of unlabeled data in BERT. BertGCN is trained by inputting each document into BERT, extracting document vectors from its output, and inputting them into GCN as initial representations of document nodes along with heterogeneous graphs of documents and words. BertGCN has now achieved state-of-the-art in the text classification task.

In BertGCN, the weights between nodes on a heterogeneous graph of words and documents are defined as shown in Equation (1) below. PPMI is used as the weights between word nodes, and TF-IDF is used as the weights between word and document nodes. As shown in equation (1), BertGCN does not express the relations between document nodes as the form of edge weights between nodes.

$$A_{i,j} = \begin{cases} PPMI(i,j), & i,j \; are \; words \; and \; i \neq j \\ TF-IDF(i,j), & i \; is \; document, j \; is \; word \\ 1, & i = j \\ 0, & otherwise \end{cases} \quad (1)$$

Yuxiao Lin et al. distinguish the names of the training models depending on the type of GNN and the pre-trained models of BERT. The names of the models are listed in Table 1. In this study, we targeted RoBERTaGCN for improvement.

Pre-Trained Model	GNN	Name of Model
bert-base	GCN	BertGCN
roberta-base	GCN	RoBERTaGCN
bert-base	GAT	BertGAT
roberta-base	GAT	RoBERTaGAT

Table 1. Names of the Models

4 Method

First, we construct a heterogeneous graph of words and documents using each document. Next, we

input the graph information (weight matrix and initial node feature matrix) to BERT and GCN and obtain each prediction. Finally, we calculate the linear interpolation of each prediction and adopt the result as the final prediction.

4.1 Build Heterogeneous Graph

First, we build a heterogeneous graph containing word nodes and document nodes. The weights of the edges between nodes i and j are defined as in Equation (2).

$$A_{i,j} = \begin{cases} COS\_SIM(i,j), & i,j \text{ are documents and } i \neq j \\ PPMI(i,j), & i,j \text{ are words and } i \neq j \\ TF-IDF(i,j), & i \text{ is document}, j \text{ is word} \\ 1, & i = j \\ 0, & otherwise \end{cases} \quad (2)$$

In RoBERTaGCN, as shown in Equation (1), the relation between words and the relation between words and documents were considered as the form of edge weights between nodes, but the relation between documents was not considered. Therefore, we improved RoBERTaGCN to consider the relation between documents by expressing the relation between documents as the form of edge weights between document nodes. $COS\_SIM(i,j)$ in equation (2) is the weight of the edge between document nodes and represents the cosine similarity. Specifically, we added the weights of the edges between document nodes by following the steps I to III below.

I. tokenize each document

Each document is tokenized by the BertTokenizer and converted into a sequence of tokens that can be input to BERT. If the number of words in a document exceeds the BERT input limit of 512, including special tokens, 510 words were extracted from the front of the document and used.

II. obtain the CLS vector

Each tokenized document is entered into BERT to obtain the CLS vector at its final hidden layer, which is a vector reflecting the features of the entire document.

III. calculate and add cosine similarity

Calculate the cosine similarity between the CLS vectors of each acquired document. If the obtained cosine similarity exceeds a predetermined threshold, the cosine similarity is added as the weight of the edge between the corresponding document nodes.

We used positive mutual information (PPMI) for weight of the edges between word nodes. We used TF-IDF for weight of the edges between word nodes and document nodes. The process from the second section onward is in accordance with RoBERTaGCN.

4.2 Creating the Initial Node Feature Matrix

The next step is to create the initial node feature matrix to be input to the GCN. We use BERT to obtain document embedding representations and treat them as input representations of document nodes. The embedding representation X_{doc} of a document node is represented by $X_{doc} \in \mathbb{R}^{n_{doc} \times d}$ using the number of documents n_{doc} and the number of embedding dimensions d. In general, the initial node feature matrix is given by the following equation (3).

$$X = \begin{pmatrix} X_{doc} \\ 0 \end{pmatrix}_{(n_{doc}+n_{word}) \times d} \quad (3)$$

4.3 Input to GCN and Learning by GCN

The weights of the edges between nodes and the initial node feature matrix shown in equations (2) and (3) are input to the GCN for training. The output feature matrix $L^{(i)}$ of the i-th layer is calculated by Equation (4).

$$L^{(i)} = \rho\left(\tilde{A} L^{(i-1)} W^{(i)}\right) \quad (4)$$

ρ is the activation function, $\tilde{A}$ is the normalized adjacency matrix. $W^i \in \mathbb{R}^{d_{i-1} \times d_i}$ is the weight matrix at layer i, $L^{(0)}$ is X, which is the input feature matrix of the model. The output of the GCN is treated as the final representation of the document nodes, and its output is input to the softmax function for classification. The prediction by the output of GCN is given by equation (5). g represents the GCN model.

$$Z_{GCN} = softmax\left(g(X,A)\right) \quad (5)$$

4.4 Interpolation of Predictions with BERT and GCN

We optimize the GCN with an auxiliary classifier that directly handles the BERT embedded

Dataset	Number of Documents	Average of Words	Training Data	Test Data
20NG	18846	206.4	11314	7532
R8	7674	65.7	5485	2189
R52	9100	69.8	6532	2568
Ohsumed	7400	129.1	3357	4043
MR	10662	20.3	7108	3554

Table2. Information of Each Data Set

representation for faster convergence and better performance. Specifically, we create an auxiliary classifier with BERT by feeding the document embedding representation X and the weight matrix W directly into the softmax function. The prediction by the auxiliary classifier is given by the following equation (6).

$$Z_{BERT} = softmax(WX) \quad (6)$$

Then, a linear interpolation is performed using Z_{GCN} which prediction from RoBERTaGCN and Z_{BERT} which prediction from BERT, and the result of the linear interpolation is adopted as the final prediction. The result of linear interpolation is given by equation (7).

$$Z = \lambda Z_{GCN} + (1 - \lambda) Z_{BERT} \quad (7)$$

λ controls the trade-off between the two predictions, meaning that if $\lambda = 1$, we use the full RoBERTaGCN model, and if $\lambda = 0$, we use only the BERT module. $\lambda \in (0, 1)$, we can balance the predictions from both models and RoBERTaGCN model can be more optimized. $\lambda = 0.7$ is the optimal value of λ, as shown by the experiments of Yuxiao.

5 Experiments

We evaluated the classification performance of the proposed method by conducting experiments with the cosine similarity threshold set between 0.5 and 0.95 to 0.995 in increments of 0.005 and investigated the optimal cosine similarity threshold for each data set.

5.1 Dataset

We evaluated the performance of the proposed method by conducting experiments using the five data sets shown in Table 2. We used the same data used in RoBERTaGCN. Each dataset was already divided into training and test data, which we used as is.[1] The number of data for training and test data is shown in Table 2.

・20-Newsgroups(20NG)

20NG is a dataset in which each document is categorized into 20 news categories, and the total number of documents is 18846. In our experiments, we used 11314 documents as training data and 7532 documents as test data.

・R8, R52

Both R8 and R52 are subsets of the dataset provided by Reuters (total number is 21578). R8 has 8 categories and R52 has 52 categories. The total number of documents in R8 is 7674, and we used 5485 documents as training data and 2189 documents as test data. The total number of documents in R52 is 9100, and we used 6532 documents as training data and 2568 documents as test data.

・Ohsumed

This is a dataset of medical literature provided by the U.S. National Library of Medicine, and total number of documents is 13929. Every document has one or more than two related disease categories from among the 23 disease categories. In the experiment, we used documents that had only one relevant disease category, and the number of documents is 7400. We used 3357 documents as training data and 4043 documents as test data.

・Movie Review(MR)

This is a dataset of movie reviews and is used for sentiment classification (negative-positive classification). The total number of documents was 10662. We used 7108 documents as training data and 3554 documents as test data.

[1] https://github.com/ZeroRin/BertGCN/tree/main/data

GPU	Tesla V100（SXM2） ／A100（SXM2）
Memory	12.69GB（standard） ／51.01GB（CPU／GPU(high memory)） ／35.25GB（TPU(high memory)）
Disk	225.89GB（CPU／TPU） ／166.83GB（GPU）

Table3. Details of the Specifications of Google Colaboratory Pro+

	20NG	R8	R52	Ohsumed	MR
Text GCN	86.34	97.07	93.56	68.36	76.74
Simplified GCN	88.50	-	-	68.50	-
LEAM	81.91	93.31	91.84	58.58	76.95
SWEM	85.16	95.32	92.94	63.12	76.65
TF-IDF+LR	83.19	93.74	86.95	54.66	74.59
LSTM	65.71	93.68	85.54	41.13	75.06
fastText	79.38	96.13	92.81	57.70	75.14
RoBERTaGCN	89.15	98.58	94.08	72.94	88.66
0.5	×	49.47	×	64.73	×
0.95	89.29	98.26	92.83	73.73	88.21
0.955	89.42	98.63	94.08	72.74	88.21
0.96	89.74	98.49	**94.16**	73.49	88.52
0.965	89.54	98.45	93.15	**74.13**	88.15
0.97	89.43	98.45	93.77	73.41	88.66
0.975	**89.82**	98.63	93.57	73.49	**89.00**
0.98	89.60	98.54	93.96		88.29
0.985	89.64	98.54	92.95	73.46	88.58
0.99	89.76	98.36	93.42	73.71	88.55
0.995	89.51	**98.81**	93.26		88.31

Table4. Result of Experiment

5.2 Experimental Environment

The experiments were conducted using Google Colaboratory Pro+, an execution environment for Python and other programming languages provided by Google. The details of the specifications of Google Colaboratory Pro+ are shown in Table 3.

We experimented by setting the threshold of cosine similarity between 0.5 and 0.95 to 0.995 in increments of 0.005 when adding the cosine similarity of CLS vectors as the weight of edges between document nodes. The performance of the proposed method was evaluated by verifying the prediction results with test data and obtaining the percentage of correct answers.

5.3 Result of Experiment

The result of experiment for each threshold of cosine similarity are shown in Table 4, along with the correct response rate of the original RoBERTaGCN.

The items marked as × are experiments could not be completed due to lack of memory. Items marked with "-" are those for which the percentage of correct responses was not indicated in the original paper. In experiment with Ohsumed, the experiments with threshold of 0.99 and 0.995, and threshold of 0.975 and 0.98 had the same number of edges of the cosine similarity of CLS vectors, so they are denoted together. It was confirmed that the proposed method outperformed the original RoBERTaGCN on all datasets at certain thresholds, but for R8, R52, and MR, there were only one or two thresholds where the proposed method outperformed the original RoBERTaGCN. On the other hand, 20NG outperformed the original RoBERTaGCN at all thresholds from 0.95 to 0.995, and Ohsumed also outperformed the original RoBERTaGCN at most of the thresholds. Most notably, the experiment with 20NG of threshold 0.975 outperformed the original RoBERTaGCN by 0.67% and the experiment with Ohsumed of

Dataset	pmi Edge	tf-idf Edge	cos_sim Edge	Average of Cosine Similarity
20NG	22413246	2276720	×	0.838
R8	2841760	323670	29441186	0.846
R52	3574162	407084	41400215	0.840
Ohsumed	6867490	588958	27376155	0.837
MR	1504598	196826	56674250	0.823

Table5. Number of Various Edges Added and the Average of Cosine Similarity

Dataset	Total Number of Document Node Combinations	Number of cos_sim Edges Added	Percentage of Edges Added
20NG	177576435	753	0.0004240
R8	29441301	175	0.0005944
R52	41400450	28890	0.0697818
Ohsumed	27376300	15	0.0000547
MR	56833791	921	0.0016205

Table 6. Percentage of the Number of Edges Added

threshold 0.965 outperformed the original RoBERTaGCN by 1.19%.

6 Discussion

Table 5 shows the number of various edges added and the average of cosine similarity in each data set. The item marked as × is the experiment could not be completed adding weight due to lack of memory. In the experiment where the threshold was set to 0.5, the experiment could not be completed due to lack of memory in the datasets of 20NG, R52, and MR. Even for R8 and Ohsumed, which were able to complete the experiment, the classification performance was much lower than that of the original RoBERTaGCN. The reason for both is that the number of edge weights between document nodes to be added became too large. In all datasets, the number of edges of cosine similarity is more than twice as large as the number of PMI edges and TF-IDF edges. In addition, since the average of the cosine similarity of the CLS vector is between 0.8~0.85 in all datasets, it is thought that a huge number of weights of edges between document nodes that are not in the same genre are also added, and they have become noise.

Analyzing the average number of words for each dataset in Table 2 and the experimental results in Table 4, we can see that the proposed method tends to obtain higher classification performance for datasets with higher average number of words compared to the original RoBERTaGCN. We believe this is because the higher the average word count, the better the CLS vectors of the documents reflect the features of those documents and the more cosine similarity weights are added between document nodes of the same genre. On the other hand, the lower the average number of words, the less the difference in the CLS vectors of the documents, the higher the cosine similarity of the CLS vectors of the documents in different genres, and the more cosine similarity weights were added to the weights between the nodes of the documents in different genres. This is thought to be the reason why the classification performance did not improve as expected in experiments with dataset have lower the average number of words.

We calculated the percentage of the number of added cosine similarities at the threshold of the cosine similarity of the CLS vector that shows the highest classification performance in Table 4. The calculation results are shown in Table 6.

Since there is no relationship between the percentage of the number of added edges and the classification performance, we think it is necessary to conduct future experiments using criteria such as "upper XX% of the cosine similarity value", instead of using the threshold of the cosine similarity of the CLS vector to determine the weights to be added between document nodes, to clarify the relationship between the number of edges between document nodes and the classification performance.

7 Conclusion and Future Work

In this paper, we confirmed that RoBERTaGCN can be improved by adding the cosine similarity of CLS vectors of documents as weights of edges between document nodes, and that it outperforms the classification performance of the original

RoBERTaGCN. In particular, experiments show that the proposed method is effective for long documents.

In the future, we intend to study the compatibility of the proposed method with GAT and the optimal value of the parameter λ for linear interpolation.

References

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. IEEE Transactions on Neural Networks, 20(1):61–80.

Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima's an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1957-1967, Copenhagen, Denmark. Association for Computational Linguistics.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 7370-7377.

Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2019. Text level graph neural network for text classification. arXiv preprint arXiv:1910.02356.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. arXiv preprint arXiv:1710.10903.

Rushlene Kaur Bakshi, Navneet Kaur, Ravneet Kaur, and Gurpreet Kaur. 2016. Opinion mining and sentiment analysis. In 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), pages 452-455. IEEE.

Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, Jianfeng Gao. 2021. Deep Learning Based Text Classification: A Comprehensive Revie. arXiv:2004.03705v3

Thomas N Kipf and Max Welling. 2016b. Variational graph auto-encoders. arXiv preprint arXiv:1611.07308.

Thomas N Kipf and Max Welling. 2016a. Semisupervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. 2020. Tensor graph convolutional networks for text classification

Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li and Fei Wu. 2021. BertGCN: Transductive Text Classification by Combining GCN and BERT.

Zhibin Lu, Pan Du, and Jian-Yun Nie. 2020. Vgcn-bert: augmenting bert with graph embedding for text classification. In European Conference on Information Retrieval, pages 369-382. Springer.