Graph Based Text Classification Using a Word-Reduced Heterogeneous Graph

Hiromu Nakajima
Major in Computer and Information Sciences
Graduate School of Science and Engineering,
Ibaraki University
Hitachi, Ibaraki, Japan
e-mail: 22nm738g@vc.ibaraki.ac.jp

Minoru Sasaki
Department of Computer and Information Sciences
Faculty of Engineering, Ibaraki University
Hitachi, Ibaraki, Japan
e-mail: minoru.sasaki.01@vc.ibaraki.ac.jp

Abstract— **Text classification, which determines the label 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. Traditional text classification methods utilizing graph structure data represent the connections between words, words and documents, and between documents themselves through edge weights between nodes. These are subsequently trained by feeding them into a graph neural network. However, such methods require a very large amount of memory, which can lead to operational issues or an inability to process large datasets in certain environments. In this study, we introduce a more compact graph structure by eliminating words that appear in only one document, deemed unnecessary for text classification. This approach not only conserves memory but also enables the use of larger trained models by utilizing the saved memory. The findings demonstrate that this method successfully reduces memory usage while maintaining the accuracy of conventional approaches. By utilizing the saved memory, the proposed method succeeded in using larger trained models, and the classification accuracy of the proposed method was dramatically improved compared to the conventional method.**

Keywords-text classification; graph convolutional neural network; Word-Reduced Heterogeneous Graph; semi-supervised learning.

I. INTRODUCTION

This article presents an extended version of the international conference paper titled "Text Classification Using a Word-Reduced Graph", which was presented during DATA ANALYTICS 2023 [1].

Text classification is the task of estimating the appropriate label for a given document from a predefined set of labels. This text classification technique has been applied in the real world to automate the task of classifying documents by humans. Many researchers are interested in developing applications that take advantage of text classification techniques, such as spam classification [2], topic labeling [3], and sentiment analysis [4].

Conventional text classification studies based on machine learning can be categorized into two phases: vector representation of text using feature extraction and machine learning-based classification algorithms [5]. In vector representation of text, the vector space model is commonly used to represent a text as a numerical feature vector in the Euclidean feature space. Classification algorithms using machine learning analyze annotated text corpora by automatically inferring which features of the text are relevant for classification. Since about a decade ago, with advances in deep learning, it has become popular to use deep learning to perform text classification. In this approach, transformer-based vector representations, which are effective text embedding techniques, have been studied widely to capture the contextual meaning of textual documents. In addition, to utilize global features in text representation, researchers have been studying graph neural networks (GNNs) [6], which learn embeddings of nodes by aggregating information from their neighbors through edges.

Among various types of GNNs, Graph Convolutional Neural Networks (GCNs) [7], which can take advantage of data in graph structures, are particularly popular for solving text classification tasks. TextGCN [8], VGCN-BERT [9], and BertGCN [10] are examples of text classification methods that utilize data in graph structures. In TextGCN [8], word and document nodes are represented on the same graph (heterogeneous graph), which is input into GCNs for learning. VGCN-BERT [9] constructs a graph based on the word embedding and word co-occurrence information in Bidirectional Encoder Representations from Transformers (BERT) and learns by inputting the graph into Vocabulary Graph Convolutional Network (VGCN). BertGCN [10] is a text classification method that combines the advantages of transductive learning of GCNs with the knowledge obtained from large-scale prior learning of BERT. Although the graphs used in these graph-based text classification methods represent relationships between words and between words and documents, they do not use relationships between documents, which creates the potential for topic drift. Therefore, the work in [11] proposed a graph structure that combines these relations with additional document-to-document relations to solve this problem. This method achieved the best performance among existing text classification methods on the three datasets (20NG, R8, and Ohsumed). However, a new problem arises from the addition of relationships between documents to the graph, which increases the size of the graph and requires a lot of memory space. Consequently, we hypothesized that by compacting the graph structure, we could mitigate memory constraints and facilitate the utilization of

larger data sets and the construction of more sophisticated models.

In this paper, we propose a text classification method that uses a graph structure in which words that are considered unnecessary are removed to solve the problem of memory shortage problem that occurs when a large graph structure is used. In this study, two objectives exist. The first is to successfully save memory by constructing a graph structure that removes words considered unnecessary in text classification to solve the problem of insufficient memory. The second is to improve classification accuracy over conventional methods by utilizing the reduced memory and using larger trained models. Specifically, words that appear in only one document are removed from the graph, reducing both the weights of edges between word nodes and the weights of edges between word nodes and document nodes, thereby saving memory. We believe that this will result in a graph that is more compact than the graphs created by conventional methods, saving memory and improving the accuracy of text classification by using a larger trained model.

This paper is organized as follows. In Section II, we first describe existing research on text classification using graphs and graph neural networks used for text classification. In Section III, we describe the proposed text classification method using a reduced word graph. In Section IV, we describe the experiments we conducted to evaluate the proposed method and show the experimental results. We discuss the experimental results presented in Section V and conclude in Section VI.

II. Related Works

In this section, we provide an overview of three types of relevant research: Conventional Text Classification Using Machine Learning, Text Classification Using Deep Learning Models and Text Classification Using Graph Neural Networks.

A. Conventional Text Classification Using Machine Learning

Text classification is one of the core tasks in understanding language with computers. Conventional text classification studies based on machine learning can be categorized into two main phases: vector representation of text using feature engineering and classification algorithms using machine learning [5]. Feature engineering involves leveraging domain knowledge of the data set to develop meaningful attributes or characteristics that make machine learning algorithms work. To make text processable computationally, it is represented as a vector of numbers while preserving as much original information as possible. For feature engineering, commonly used features are BOW (Bag-Of-Words) [12], N-gram [13], TF-IDF (Term Frequency-Inverse Document Frequency) [14], co-occurrence relations between words [15], etc. A variety of classification algorithms have been developed to categorize textual data based on the extracted features. Among traditional methods for text classification, general classification models such as Naive Bayes [16], Logistic Regression [17], K-Nearest Neighbor [18], Support Vector Machine [19] and Random Forest [20] have been proposed.

B. Text Classification Using Deep Learning Models

Text classification based on neural networks has been actively researched since about a decade ago. In early studies, deep learning architectures were used to learn word embeddings from large text corpora, which were then employed for text classification [21]. A typical word embedding methods are Word2vec [22], Glove [23], FastText [24], Long Short-Term Memory (LSTM) [25], ELMo [26], BERT [27] and RoBERTa [28]. Liu et al. proposed a multi-task deep neural network (DNN) model for learning representations across multiple tasks [29]. This multi-task DNN approach addresses both query classification and ranking tasks within the context of web search. Wang et al. introduced Label-Embedding Attentive Models (LEAM) as a method to represent both text and labels within the same space for text classification [30]. By incorporating label descriptions, LEAM improves text classification performance. Shen et al. introduced a new method to text classification called Simple Word-Embedding-based Models (SWEMs) [31]. SWEMs employ word embeddings and parameter-free pooling operations to encode text sequences. Their research demonstrated the effectiveness of deep learning methods for this task. Some recent studies have employed neural networks such as the Multi-Layer Perceptron (MLP) [32], the Convolutional Neural Network (CNN) [33], Recurrent Neural Network (RNN) [43] and Long Short Term Memory (LSTM) [34] as classification models. A Deep Average Network (DAN) computes a sentence embedding by averaging pre-trained word embeddings and then processes this embedding through two fully-connected layers and a softmax output layer [32].

C. Text Classification Using Graph Neural Networks

Recent text classification research has explored graph-based approaches where the connections between words and documents are quantified by edge weights. Graph Neural Network (GNN) [6] is a neural network that learns relationships between graph nodes via the edges that connect them. There are several types of GNNs depending on their form. Employing GNNs for large-scale text processing comes at a significant cost in terms of computational resources. To remove unnecessary complexity and redundant computations in the model, Wu et al. proposed the Simple Graph Convolution model (Simplified GCN) by repeatedly removing the non-linearities and merging weight matrices between consecutive layers into a single linear transformation [35]. Graph Convolutional Neural networks (GCNs) [7][41][42] is a neural network that takes a graph as input and learns the relationship between nodes of interest and their neighbors through convolutional computation using weights assigned to the edges between the nodes. Graph Autoencoder (GAE) [36] is an extension of autoencoder, which extracts important features by dimensionality reduction of input data, to handle graph data as well. Graph Attention Network (GAT) [37] is a neural network that updates and learns node features by multiplying the weights of edges between nodes by

TABLE I. MODEL NAME DEFINITION BASED ON PRE-TRAINING MODELS AND GNN TYPES.

Model Name	Pre-Trained Model	GNN type
BertGCN	bert-base (bert-large)	GCN
BertGAT	bert-base (bert-large)	GAT
RoBERTaGCN	roberta-base (roberta-large)	GCN
RoBERTaGAT	roberta-base (roberta-large)	GAT

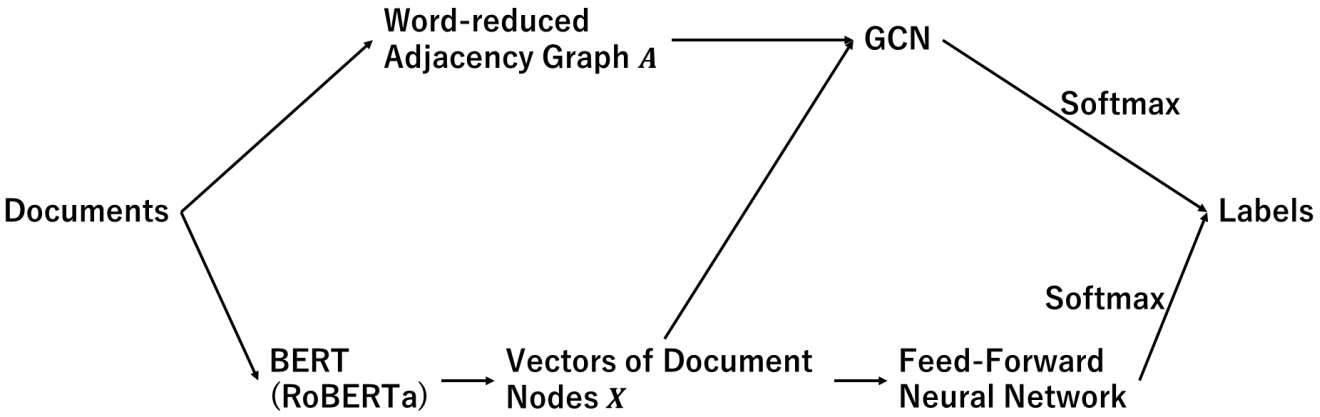

Figure 1. Schematic Diagram of the Proposed Method.

Attention, a coefficient representing the importance of neighboring nodes. GNNs are used in a wide range of tasks in the field of machine learning, such as relation extraction, text generation, machine translation, and question answering, and have demonstrated high performance. The impressive performance of GNNs across these wide-ranging tasks has inspired researchers to explore GNN-based approaches for text classification, with a particular focus on GCN models. In TextGCN [8], document and word nodes are represented on the same graph (heterogeneous graph), which is input into GCNs for training. In recent years, text classification methods that combine large-scale pre-trained models such as BERT with GCNs have also been studied extensively. VGCN-BERT [9] constructs a graph based on word co-occurrence information and BERT's word embedding and inputs the graph into GCNs for learning. In BertGCN, a heterogeneous graph of words and documents is constructed based on word co-occurrence information and BERT's document embedding, and the graph is input into GCNs for learning [10]. In [11], we propose a graph structure that exploits relationships between documents. TensorGCN, a model proposed by Liu et al., addresses text classification by combining intra-graph and inter-graph information propagation [29]. This enables the model to learn effective representations for both individual text elements and the overall document. The detailed description of the proposed text classification model is given in Section III.

III. TEXT CLASSIFICATION METHOD USING A WORD-REDUCED GRAPH

In this section, we describe the text classification method using a word-reduced graph.

A. Definition of Classification Models Based on Pre-training Models and GNN Types

BertGCN is a text classification method that combines BERT model obtained by large-scale pre-training language model utilizing large unlabeled data with the GCN models for transductive learning [9]. In the BertGCN model, documents are encoded by BERT to yield document vectors, which serve as initial node representations in a GCN. The GCN is trained on a heterogeneous graph composed of documents and words.

Lin et al. distinguish the model names according to the pre-trained BERT model and the type of GNN used [10]. Table I shows the definitions of the model's name, corresponding pre-trained models and GCN types. This study focuses on enhancing the performance of RoBERTaGCN, a model that integrates roberta-base and GCN.

B. Text Classification Based on GCN Using Word-reduced Graph

This subsection describes the details of the proposed classification method. Figure 1 shows a schematic diagram of the proposed method. First, a reduced heterogeneous graph of words and documents is constructed from documents. Next, the graph information (weight matrix and initial node feature matrix) is input into the GCN, and the document vector is input into the feed-forward neural network. Finally,

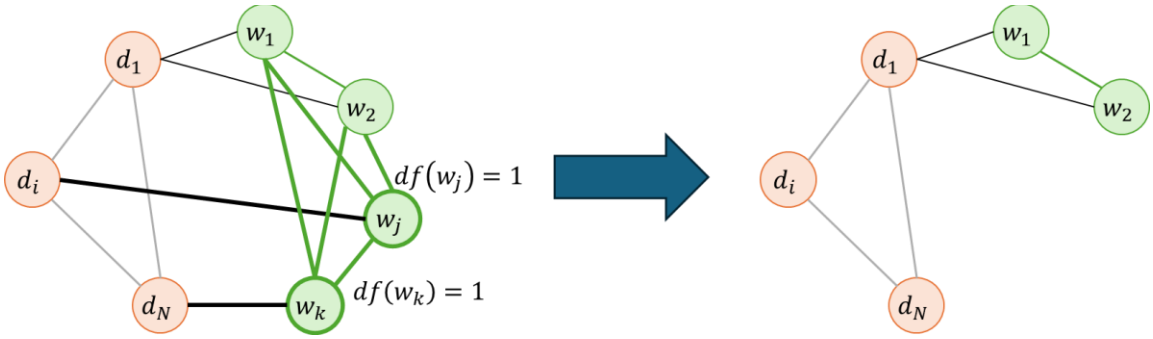

Figure 2. Methods for Removing Word Nodes in the Proposed Method.

a linear interpolation of the two predictions is computed and the result is used as the final prediction.

1) Build Heterogeneous Graph

First, a heterogeneous graph containing word and document nodes is constructed from given documents. The proposed method uses a heterogeneous graph as shown in the existing study [11]. Figure 3 shows the weighting methods for three types of nodes. As shown in Equation (1), the proposed method represents relationships among documents, among words, and between words and documents as weights on the edges of the graph. In the existing study [11], a node is created for every word that appears in the dataset and the weights of the edges are calculated. However, in this study, to reduce the number of nodes, the word nodes with a document frequency of 1 for a word ($df(w) = 1$) are removed from the heterogeneous graph and the PPMI and TF-IDF are not calculated, as shown in Figure 2. By removing the unimportant word nodes in the graph, we expect to make efficient use of the memory space that is required for the representation of the graph.

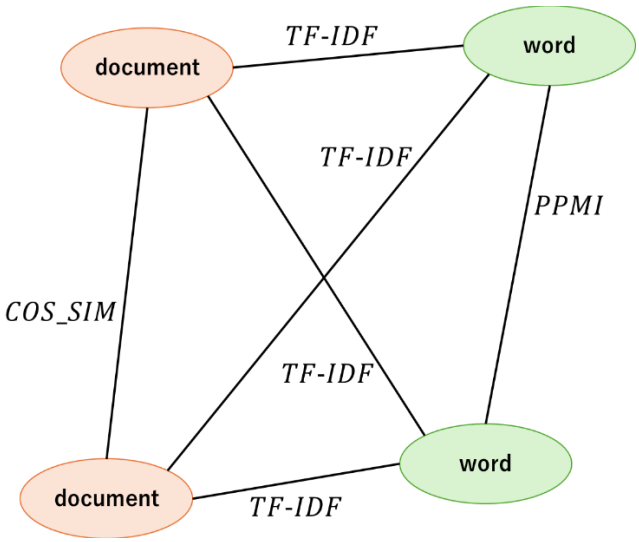

Figure 3. Weighting methods for three types of nodes.

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

The weights of the edges between document nodes in the equation (1) represent the cosine similarity $COS\_SIM(d_i, d_j)$ between the two document nodes d_i and d_j, which is a measure of how similar the two documents are. This $COS\_SIM(d_i, d_j)$ is defined as follows:

$$COS\_SIM(d_i, d_j) = \frac{d_i \cdot d_j}{\|d_i\|\|d_j\|}, \quad (2)$$

where d_i and d_j are document embeddings of the document i and j.

Each document is converted into a sequence of tokens that can be entered into BERT. A special classification token ([CLS]) is added to the beginning of the document and a special separator token ([SEP]) is added to the end of the document. These are special tokens. The [CLS] token indicates the beginning of the sentence, and the [SEP] token indicates the end of the sentence. In this study, a single document was considered to be a single sentence. For long documents (more than 512 words), we extract the first 510 words and add special tokens to make it 512 words long. For short documents (less than 510 words), we fill them with 0s to reach the 512-word limit for BERT.

Each tokenized document is fed into BERT to obtain a [CLS] vector of the last hidden layer in BERT. The [CLS] vector is a representation of the entire document that captures the context of the document. We compute the cosine distance between the [CLS] encodings of each document and add edges between corresponding document nodes if the cosine similarity is greater than a predefined threshold, where the weight of each edge is the cosine similarity of the [CLS] vectors.

The function $PPMI(w_i, w_j)$ represents positive point-wise mutual information (PPMI) that is used to weight edges between word nodes [38]. This $PPMI(w_i, w_j)$ gives more importance to measure the semantic similarity between the word w_i and the word w_j in a document. For any word pair (w_i, w_j), point-wise mutual information (PMI) is defined as the log ratio between their joint probability and product of their marginal probabilities as follows [39]:

$$PMI(w_i, w_j) = \log_2 \frac{p(w_i, w_j)}{p(w_i)p(w_j)}, \quad (3)$$

Therefore, as shown in Equation (4), the PPMI converts the maximum of the calculated PMI and 0 as the weight of edges between word nodes w_i, w_j.

$$PPMI(w_i, w_j) = \max\left(0, PMI(w_i, w_j)\right) \quad (4)$$

Term frequency-inverse document frequency (TF-IDF) [40] is used for the weights of edges between word nodes and document nodes. TF-IDF values are larger for words that occur more frequently in one document but less frequently in other documents, i.e., words that characterize that document. The TF-IDF value can be calculated by multiplying the TF value by the IDF value. The TF value is a value representing the frequency of occurrence of a word. The TF value is calculated by Equation (5).

$$TF(w_i, d_j) = \frac{f(w_i, d_j)}{\sum_{w_k \in d_j} f(w_k, d_j)} \quad (5)$$

d_j is a document. w_i is a word that appears in d_j. The function $f(w_i, d_j)$ is the frequency of the word w_i in the document d_j. The IDF value of a word w_i is calculated by taking the logarithm of the total number of documents n_{doc} in the data set divided by the number of documents containing the word w_i as shown in Equation (6).

$$IDF(w_i) = \log\left(\frac{n_{doc}}{df(w_i) + 1}\right) \quad (6)$$

N is the total number of documents. df is the number of documents in which w_i appears. TF-IDF value is calculated by Equation (7).

$$TF\text{-}IDF(w_i, d_j) = TF(w_i, d_j) \cdot IDF(w_i) \quad (7)$$

2) Creating the Initial Node Feature Matrix

Each document is converted into a sequence of tokens that Creating the Initial Node Feature Matrix

Next, we create the initial node feature matrix to be input into the GCNs. We use BERT to obtain document embeddings and treat them as the input representations of the document nodes. The embedded representation X_{doc} of a document node is represented by $X_{doc} \in \mathbb{R}^{n_{doc} \times d}$, where n_{doc} is the number of documents and d is the number of embedding dimensions. Overall, the initial node feature matrix is given by (8).

$$X = \begin{pmatrix} X_{doc} \\ 0 \end{pmatrix}_{(n_{doc}+n_{word}) \times d} \quad (8)$$

3) Input into GCN (GAT) and Learning by GCN (GAT)

The weights of the edges between nodes and the initial node feature matrix are input into GCNs for training. The output feature matrix $L^{(i)}$ of layer i is computed by (9).

$$L^{(i)} = \rho\left(\tilde{A}L^{(i-1)}W^{(i)}\right) \quad (9)$$

ρ is the activation function and $\tilde{A}$ 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, the input feature matrix of the model. The dimension of the final layer of W is (number of embedded dimensions) × (number of output classes). The output of the GCNs is treated as the final representation of the document node, and its output is input into the softmax function for classification. The prediction by the output of the GCNs is given by (10). The function g represents the GCNs model. The cross-entropy loss in labeled document nodes is used to cooperatively optimize the parameters of BERT and GCNs.

$$Z_{GCN} = softmax\left(g(X, A)\right) \quad (10)$$

When GAT is used, the feature update of node i is given by Equation (11).

$$\vec{h}_i' = \rho\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W \vec{h}_j\right) \quad (11)$$

$\vec{h}$ is a vector, of each node. $\mathcal{N}$ is some neighborhood of node i. α is the attention between node i and node j. Attention α is given by Equation (12).

$$\alpha_{ij} = \frac{exp\left(LeakyReLU\left(\vec{a}^T\left[W\vec{h}_i \| W\vec{h}_j\right]\right)\right)}{\sum_{k \in \mathcal{N}_i} exp\left(LeakyReLU\left(\vec{a}^T\left[W\vec{h}_i \| W\vec{h}_k\right]\right)\right)} \quad (12)$$

The attention mechanism a is a single-layer feedforward neural network and applying the $LeakyReLU$ nonlinearity. GAT's prediction is given by the following Equation (13).

$$Z_{GAT} = softmax\left(gat(X, A)\right) \quad (13)$$

C. Interpolation of Predictions with BERT and GCN

A linear interpolation is computed with Z_{GCN}, the prediction from RoBERTaGCN, and Z_{BERT}, the prediction from BERT, and the result of the linear interpolation is adopted as the final prediction. The result of the linear interpolation is given by the following Equation (14).

TABLE II. OPTIMAL VALUE FOR COSINE SIMILARITY THRESHOLD.

Dataset	Optimal Threshold Value
20NG	0.99
R8	0.975
R52	0.96
Ohsumed	0.965
MR	0.97

TABLE III. INFORMATION OF EACH DATA SET.

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

TABLE IV. DETAILS OF THE SPECIFICATIONS OF GOOGLE COLLABORATORY PRO+.

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）

$$Z = \lambda Z_{GCN} + (1-\lambda) Z_{BERT} \quad (14)$$

λ controls the trade-off between the two predictions. $\lambda = 1$ means using the full RoBERTaGCN model, while $\lambda = 0$ means using only the BERT module. When $\lambda \in (0,1)$, the predictions from both models can be balanced, making the RoBERTaGCN model more optimal. Experiments by Lin et al. using the graph structure in (1) show that $\lambda = 0.7$ is the optimal value of λ [10].

IV. EXPERIMENTS

In this study, two experiments were conducted.

Experiment 1: Experiment to confirm the effectiveness of the graphs of the proposed method.

In Experiment 1, the classification performance of the proposed method using compact graphs was compared with other methods. The parameter λ, which controls the balance between predictions from BERT and predictions from GCNs, was fixed at 0.7. Preliminary experiments were conducted on the validation data, and the optimal values of the threshold of the cosine similarity for each dataset are shown in Table II. We used the values in Table II as our threshold values. The trained model used was roberta-base. Accuracy was used to evaluate the experiment. Positive is the label of the correct answer, negative is the label of the incorrect answer, and negative is all the remaining labels except the correct label.

Experiment 2: Experiment to check classification accuracy when changing to a larger trained model.

In Experiment 2, we take advantage of the memory savings and check the accuracy of the proposed method by applying a larger trained model. Specifically, the learned model is changed from roberta-base to roberta-large. λ and cosine similarity values are set to the same values as in Experiment 1.

A. Data Set

We evaluated the performance of the proposed method by conducting experiments using the five data sets shown in Table III. We used the same data used in RoBERTaGCN. Each dataset was already divided into training and test data, which we used as is. The ratio of training data to test data is about 6:4 for 20NG, about 7:3 for R8 and R52, about 4.5:5.5 for Ohsumed, and about 6.5:3.5 for MR. All five datasets were used in the experiments after the preprocessing described in the next subsection.

1) 20-Newsgroups (20NG)

20NG is a dataset in which each document is categorized into 20 news categories, and the total number of documents is

TABLE V. CLASSIFICATION PERFORMANCE OF THE PROPOSED METHOD.

	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
BERT	85.30	97.80	96.40	70.50	85.70
RoBERTa	83.80	97.80	96.20	70.70	89.40
RoBERTaGCN	89.15	98.58	94.08	72.94	88.66
Extended RoBERTaGCN [11]	89.82	**98.81**	94.16	74.13	89.00
Proposed method（base）	**90.02**	98.58	**96.88**	73.53	89.65
Proposed method（large）	89.95	98.58	96.81	**76.08**	**91.50**

18846. 11314 documents, corresponding to about 60% of all documents, are training data. 7532 documents, corresponding to about 40% of all documents, are test data.

2) 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.

3) 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.

4) 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.

B. Preprocessing

The following three preprocessing steps were applied to all data. These preprocessing steps are the same as those done in RoBERTaGCN [10].

Step1: Noise Removal.
All characters and symbols except alphanumeric characters and certain symbols (( ) , ! ? ' `) were removed as noise.

Step2: Word Normalization.
All alphanumeric characters were normalized to half-width alphanumeric characters. Then, normalized alphanumeric characters are unified into lowercase letters.

Step3: Stop Words Removal.
Stop words in text were removed using stop words list of Natural Language Toolkit (NLTK).

C. 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 IV.

D. Evaluation Metric

The accuracy was used as the evaluation index for the experiment. In previous studies, including RoBERTaGCN [9] and TextGCN [8], accuracy has been used as an evaluation index, and to make it easier to compare results, accuracy was also used in this study. The accuracy is calculated by Equation (14). Positive is the label of the correct answer, and negative is the label of the incorrect answer. Negatives are all the remaining labels except the correct answer label. TP(True-Positive) is the number of items that should be classified as positive that were correctly classified as positive. TN(True-Negative) represents the number of items that should be classified as negative that were correctly classified as negative. FP (False-Positive) indicates the number of cases where items that should have been classified as negative were incorrectly classified as positive. FN (False-Negative) indicates the

TABLE VI. NUMBER OF WORDS REMOVED.

Dataset	Number of Words	Number of Words Removed
20NG	42757	755
R8	7688	225
R52	8892	245
Ohsumed	14157	851
MR	18764	8687

TABLE VII. NUMBER OF PPMI EDGES REMOVED.

Dataset	Number of PPMI Edges	Number of Edges Removed
20NG	22413246	127662
R8	2841760	32954
R52	3574162	36138
Ohsumed	6867490	129938
MR	1504598	314950

TABLE VIII. NUMBER OF TF-IDF EDGES REMOVED.

Dataset	Number of TF-IDF Edges	Number of Edges Removed
20NG	2276720	755
R8	323670	225
R52	407084	245
Ohsumed	588958	851
MR	196826	8687

number of cases where items that should have been classified as positive were incorrectly classified as negative.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (14)$$

E. Result of Experiment

Table V compares the classification performance of the proposed method with the conventional methods. The previous study [6] shows the classification performance when using the graph structure in (2). The proposed method (base) is the result of Experiment 1, and the proposed method (large) is the result of Experiment 2.

Comparing the results of the Proposed method (base) with the other methods, the accuracy of 20NG, R52, and MR improved. The accuracy of the other datasets also maintains a high level. Even with a compact graph in which words that appear only in one document are removed, the classification performance remains high. Therefore, it can be said that the proposed method succeeds in saving memory.

Comparing the results of the Proposed method (large) with the other methods, the accuracy is significantly improved for Ohsumed and MR. The classification performance of Ohsumed was 76.08%, 1.95% higher than that of [5], and that of MR was 91.50%, 1.85% higher than that of the Proposed method (base).

V. DISCUSSION

Table VI shows the number of word types that appear in each dataset and the number of words that are removed in the graph structure of (3). Table VII shows the number of PPMI edges added in the original graph structure and the number of PPMI edges removed in the graph structure of (3). Table VIII shows the number of TF-IDF edges added in the original graph structure and the number of TF-IDF edges removed in the graph structure of (3). Since TF-IDF edges are added between word and document nodes, the number of edges removed is the same as the number of words removed. From these three tables, it can be seen that the graph of the proposed method reduces the number of edges by 1 to 20%. Experimental results show that the classification performance of the proposed method maintains performance of the method using the original graph structure. Therefore, it can be said that the proposed method succeeds in saving memory because it reduces the number of edges on the graph while maintaining accuracy.

We believe that the reason why the accuracy was maintained even with a compact graph is because the words to be removed were limited to words that appear only in a single document. Words that appear in only one document do

not propagate document topic information through the word node, and thus text classification performance is maintained even if those words are removed.

This study also confirmed the document classification performance when the trained model was changed to a larger one, taking advantage of the memory savings. When the learned model was changed from roberta-base to roberta-large, the accuracy improved significantly. It is thought that the change to roberta-large improved the accuracy because it was able to acquire embedded representations that better reflect the characteristics of the documents.

VI. Conclusion and Future Work

To solve the memory-consuming problem of conventional text classification methods based on graph structures, this paper proposes the text classification method using compact graphs in which words that appear only in one document are removed. Experimental results confirmed that the proposed method can maintain the accuracy of the conventional method while saving a lot of memory. The results also showed that the accuracy of text classification improves when the learned model is changed to a larger one, taking advantage of the memory saved. By utilizing the saved memory, the proposed method succeeded in using larger trained models, and the classification accuracy of the proposed method was dramatically improved compared to the conventional method.

Future work includes comparing accuracy with the proposed method when other features are used instead of cosine similarity and optimizing the parameter λ for each data.

References

[1] H. Nakajima and M. Sasaki, “Text Classification Using a Word-Reduced Graph”, Proceedings of The Twelfth International Conference on Data Analytics (DATA ANALYTICS 2023), pp. 25-30, 2023.

[2] A. Karim, S. Azam, B. Shanmugam, K. Kannoorpatti, and M. Alazab, "A Comprehensive Survey for Intelligent Spam Email Detection," in IEEE Access, vol. 7, pp. 168261-168295, 2019, doi: 10.1109/ACCESS.2019.2954791.

[3] J. H. Lau, K. Grieser, D. Newman, and T. Baldwin, ‘‘Automatic labelling of topic models,’’ in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 1536–1545, 2011.

[4] P. K. Jain, R. Pamula, and G. Srivastava, “A systematic literature review on machine learning applications for consumer sentiment analysis using online reviews”, Computer Science Review, Vol. 41, 2021.

[5] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. E. Barnes, and D. E. Brown, "Text Classification Algorithms: A Survey", Information Vol. 10, No. 4, 2019.

[6] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model." IEEE Transactions on Neural Networks, vol. 20, no. 1, pp. 61-80, 2008.

[7] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks.” In ICLR, 2017.

[8] L. Yao, C. Mao, and Y. Luo, “Graph convolutional networks for text classification.” In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 7370-7377, 2019.

[9] Z. Lu, P. Du, and J. Y. Nie, “Vgcn-bert: augmenting bert with graph embedding for text classification.” In European Conference on Information Retrieval, pp. 369-382, 2020.

[10] Y. Lin, Y. Meng, X. Sun, Q. Han, K. Kuang, J. Li, and F. Wu, “BertGCN: Transductive Text Classification by Combining GCN and BERT” In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pp. 1456–1462, 2021.

[11] H. Nakajima and M. Sasaki, “Text Classification Using a Graph Based on Relationships Between Documents.” In Proceedings of the 36th Pacific Asia Conference on Language, Information and Computation, pp. 119–125, Manila, Philippines. De La Salle University. 2022.

[12] Y. Zhang, R. Jin, and Z. Zhou, "Understanding bag-of-words model: A statistical framework." International Journal of Machine Learning and Cybernetics 1, 1–4, pp. 43–52, 2010.

[13] W. B. Cavnar and J. M. Trenkle, "N-gram-based text categorization." In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, Vol. 161175, 1994.

[14] R. Baeza-Yates and B. Ribeiro-Neto, "Modern information retrieval. " ACM press, Vol. 463, 1999.

[15] F. Figueiredo, L. Rocha, T. Couto, T. Salles, M. A. Goncalves, and W. Meira Jr., "Word co-occurrence features for text classification. " Information Systems, 36(5), pp. 843-858, 2011.

[16] E. Maron, “Automatic indexing: An experimental inquiry,” Journal of the ACM, vol. 8, no. 3, pp. 404–417, 1961.

[17] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes." In Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, pp. 841-848, 2001.

[18] T. Cover and P. Hart, "Nearest neighbor pattern classification," in IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21-27, 1967.

[19] T. Joachims, “Text categorization with support vector machines: Learning with many relevant features,” In Proceedings of the 10th European Conference on Machine Learning, pp. 137-142, 1998.

[20] L. Breiman, “Bagging predictors,” Machine Learning, vol. 24, no. 2, pp. 123-140, 1996.

[21] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, “Deep Learning Based Text Classification: A Comprehensive Review.” ACM Computing Surveys, vol. 54, Issue 3, no. 62, pp. 1-40, 2021.

[22] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space.” In Proceedings of ICLR, 2013.

[23] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global vectors for word representation.” In Proceedings of the EMNLP, pp. 1532–1543, 2014.

[24] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification." arXiv preprint arXiv:1607.01759 2016.

[25] S. Hochreiter and J. Schmidhuber, “Long short-term memory.” Neural Computation, 9(8), pp. 1735–1780, 1997.

[26] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep Contextualized Word Representations.” In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 2227–2237, 2018.

[27] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 4171–4186, 2019.

[28] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “RoBERTa: A Robustly Optimized BERT Pretraining Approach.” ArXiv abs/1907.11692, 2019.

[29] X. Liu, X. You, X. Zhang, J. Wu and P. Lv, “Tensor graph convolutional networks for text classification” arXiv:2001.05313v1. pp. 8409-8416, 2020.

[30] G. Wang, C. Li, W. Wang, Y. Zhang, D. Shen, X. Zhang, R. Henao, and L. Carin, "Joint Embedding of Words and Labels for Text Classification." In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, pp. 2321–2331, 2018.

[31] D. Shen, G. Wang, W. Wang, M. R. Min, Q. Su, Y. Zhang, C. Li, R. Henao, and L. Carin, "Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms." In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, pp. 440–450, 2018.

[32] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III, "Deep Unordered Composition Rivals Syntactic Methods for Text Classification." In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pp. 1681-1691, 2015.

[33] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A Convolutional Neural Network for Modelling Sentences." In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pp. 655-665, 2014.

[34] K. S. Tai, R. Socher, and C. D. Manning, "Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks." In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pp. 1556-1566, 2015.

[35] F. Wu, T. Zhang, A. H. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying Graph Convolutional Networks." International Conference on Machine Learning, pp. 6861-6871, 2019.

[36] T. N Kipf and M. Welling, "Variational graph auto-encoders." arXiv preprint arXiv:1611.07308. 2016b.

[37] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks." arXiv preprint arXiv:1710.10903. 2017.

[38] D. Jurafsky and J. H. Martin, "Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition", Upper Saddle River, N. J.: Pearson Prentice Hall, 2009.

[39] K. W. Church and P. Hanks, "Word Association Norms, Mutual Information, and Lexicography", Computational Linguistics, 16(1): pp. 22–29, 1990.

[40] I. H. Witten, A. Moffat, and T. C. Bell, "Managing Gigabytes: Compressing and Indexing Documents and Images." Morgan Kaufmann, 1999.

[41] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. S. An, "Graph Convolutional Encoders for Syntax-aware Neural Machine Translation." In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 1957-1967, Copenhagen, Denmark. Association for Computational Linguistics. 2017.

[42] L. Huang, D. Ma, S. Li, X. Zhang, and H. Wang, "Text level graph neural network for text classification." In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3444–3450, 2019.

[43] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning." In IJCAI, pp. 2873–2879, AAAI Press. 2016.