Combining Topic and GNN Models for Text Classification

Lawrence Y. H. Low **Yen-Tsang Wu** **Jenq-Haur Wang**
Department of Computer Science and Information Engineering
National Taipei University of Technology
Web Information Retrieval Lab
t113999402@ntut.org.tw buddyswu@gmail.com jhwang@ntut.edu.tw

Abstract

Deep Learning (DL) models in Natural Language Processing (NLP) are divided into two categories: Sequential-based and Graph-based. More recently, Sequential models use Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), and Bidirectional Encoder Representations from Transformers (BERT). Researchers have also applied graph-based model to NLP, building graphs to learn the semantic features of text. In this study, we propose an efficient graph-based model called TopicGraph by combining Topic modeling and Graph Neural Network (GNN) which allows the reusing of the trained model to easily build graphs for inference since words from a topic has been learned during the training phrase. We use four different English datasets for our experimentation. Comparing with other sequential and graph-based models, the accuracy of our proposed model TopicGraph on MR, R8, R52, and Ohsumed datasets achieve 79%, 97%, 94%, and 72%, respectively. By adding a Subgraph-wise sampling technique, GraphSAINT for feature extraction, together with BERT as a multi-class classifier, we further improve the accuracy of our TopicGraph model of MR by 12%, R8 by 0.08%, R52 by 1.71% respectively. This clearly demonstrated the effectiveness of our proposed approach.

Keywords: Topic model, Graph neural network, Subgraph-wise sampling, Text classification

1 Introduction

With the advancement of Large Language Models (LLMs) and Generative Artificial Intelligence (AI), more works have been researched using these techniques in the field of Deep Learning (DL) recently, especially in the area of Natural Language Processing (NLP). However, it is also equally important to research into the primary task of text classification to understand the semantic meaning of text in order to accurately and efficiently classifying them.

The traditional text classification method usually is using bag-of-words to extract the semantic and linguistic meaning of text by using term frequency (TF). Topic Modeling is a traditional Machine Learning (ML) method which try to classify text document based on topics that are extracted from a text. The above two methods do not take into consideration of word position within a document, but focus instead on the word occurrence.

More advance Neural Network (NN) techniques was employed to look into this limitation by applying CNN (Kim, 2014), RNN (Liu, Qiu, & Huang, 2016) and BERT (Devlin et al., 2019) framework with the advancement of DL technology. RNN and BERT both considered when and where a text appears in a document in order to extract more vital features from a text, this can further improve the text classification accuracy.

CNN was initially used for image classification by using the convolutional layer to extract the image features. In the field of NLP, CNN can be applied to extract the local information of N-gram features in a text document. More recently, Graph Convolutional Neural Network (GCN) (Kipf and Welling, 2017) is a GNN technique combining both CNN and NN in order to capture the global information or Co-occurrence features in a text document using graph-based method.

In this study, we propose a graph-based model called TopicGraph by combining both Topic model and Graph Neural Network (GNN) model to classify text.

2 Related Work

2.1 Dependency parser graph-based model

Chou et al. (2020) introduced a dependency parser graph-based model for text classification. The researchers used RNN to calculate the text features in a text sequence. All texts are treated as a graph node by using the node modified relationship between the texts to build a graph. Their Dep-GAT-root model, combined with Graph Attention Network (GAT) and a dependency parser for graph construction, extracted text features to perform English text classification with good results.

2.2 Gated Recurrent Unit (GRU)

The Gated recurrent unit (GRU) is a form of RNN (Cho et al., 2014). This hidden unit consists of 2 gates, a reset gate and an update gate. These 2 gates control the information flowing within the hidden cell, either to update the hidden state using the update gate or forgetting the previous state using the reset gate. The update gate will determine if a hidden state requires an update with a new hidden state, whereas the reset gate determines the exemption of a previous hidden state. This GRU unit is slightly different from Long Short-Term Memory (LSTM) which has 4 gates and a memory cell (Graves, 2012). Since GRU has only 2 gates, it has fewer parameters than LSTM, hence it is more efficient and requires less computational powers.

2.3 Latent Dirichlet Allocation (LDA) for Topic Modeling

The Latent Dirichlet Allocation (LDA) model (Blei et al., 2003) is a generative probabilistic topic model that can discover hidden latent topics from a corpus that is categorized by their word distribution. In the LDA model, the nodes represent random variables whereas the edges represent the dependent relationship in between the nodes and the edges. In general, the generative process of LDA is defined as follows:

1) For each topic k, get a distribution over the vocabulary V.
2) For every document d, get a distribution over topics (per-document topic proportion).
3) With every document d, for a word w within document d, get a topic assignment (per-word topic assignment).
4) With every document d, for a word w within document d, get a word from the vocabulary V.

2.4 Graph Convolutional Network (GCN)

A graph is a data structure type that is made up of 2 components, a node and an edge. The most basic graph can be represented as $G = (V, E)$ where V is the set of nodes, and E is the edges or links between them. Edges between the nodes are identified as directed if there are any directional dependencies in between the nodes. Edges are identified as undirected when there are no directional dependencies.

Graph Neural Network (GNN) is divided into four main categories (Wu et al., 2019). They are Recurrent Graph Neural Networks (GNN), Convolutional Graph Neural Networks (CGNN), Graph Autoencoders, and Spatial-Temporal Graph Neural Networks. In this paper, we focus on works in the category of Graph Convolutional Network (GCN), Kipf and Welling (2017) introduced a scalable graph architecture that is used for semi-supervised learning on graph-structured data. This architecture is based on the CNNs which operates directly on graphs. The GCN model can scale linearly based on the number of graph edges. More importantly, the hidden layer representations that used to encode both the local graph structure and features of nodes are learnable.

For GCN to be scalable and computationally efficient, GCN uses layer-wise propagation rule which is based on the first-order approximation of spectral convolutions on graphs. GCN is a transductive model that is trained on a specific graph and makes predictions for the nodes within that graph. The model has access to the entire graph structure during both training and inference. The key characteristic is that the model can leverage information from the entire graph, including all nodes and edges, to learn embeddings and make predictions. However, this characteristic has its limitation; this GCN model cannot be applied to new data that was not seen during training.

2.5 Graph Attention Networks (GAT)

To overcome the transductive nature of the GCN model, adding an attention layer to the GCN model proved to be effective. Veličković et al. (2018) presented the Graph Attention Networks (GAT) model which is a combination of a GNN with an attention layer architecture.

GAT leveraged on the masked self-attentional layers in the architecture to address the drawbacks

based on graph convolutions. The implementation of the attention mechanism enables different weights to be assigned to the different nodes in a neighborhood. GAT model is parametrized by a weight vector with LeakyReLU activation and the GAT multiheaded attention have nodes with different weights assignment on its neighborhood. The key difference between GAT and GCN is how the information from one-hop neighborhood is combined.

2.6 Word2Vec

Mikolov et al. (2013) proposed the 2 new model architectures known collectively as Word2Vec to create dense representation of word embeddings. Word2Vec is a continuous vector representation of words for any set of corpora. However, it does not consider the positions or relationships between the words and 1 word is usually a representation in Word2Vec.

Continuous Bag-of-Words (CBOW) is an architecture in Word2Vec. CBOW uses both words in the past and words in the future for word embeddings. Skip-gram is the other architecture in Word2Vec. Instead of using past and future words for word embeddings in the case of CBOW, Skip-gram focuses on a centralized word to predict the surrounding words. In the CBOW model, the surrounding context words representations are used to predict a target word. For example, given a series of context words "the brown bear up the tree", CBOW model will use these context words to predict the target word such as "climbs", "hops" … based on the model parameters.

Different from CBOW, Skip-gram model used the target context word to predict its surrounding context words. For example, given the context word "climbs", Skip-gram model will use the target context word to predict its surrounding words such as "the", "brown", "bear", "up", "the", "tree" … based on the model parameters.

2.7 Subgraph-wise Sampling: GraphSAINT

Graph sampling based inductive learning method (GraphSAINT) introduced by Zeng et al. (2020) is a Subgraph-wise sampling technique. The purpose of GraphSAINT is to provide a scalable algorithm for large scale GNN models training by utilizing small subgraphs sampling.

First, GraphSAINT builds minibatches for training by sampling the training graph instead of using the nodes or edges across the full GCN layers. A complete GCN is constructed from the sampled subgraph for each iteration. In this way, GraphSAINT ensures that a fixed number of well-connected nodes are connected throughout all the layers.

The training process can be defined in 3 steps as follows:

1) Select subgraphs randomly.
2) Construct and run GNN only on the subgraph level.
3) Repeat recursively.

In this paper, we generated the subgraphs using 3 random sampling techniques in GraphSAINT (Zeng et al., 2020), which are Node sampler, Edge sampler and Random Walk sampler (Wang et al., 2019). The random walk sampler begins at a randomly selected starting node, which is selected from the set of root nodes.

The process of random walking steps can be illustrated as follows:

1) Step 1: Randomly select a starting node as the starting point, move to a neighboring node.
2) Step 2: After reaching the neighboring node in Step 1, move to another neighboring node.

We select a shorter random walk as it will capture nearby local neighborhood information around the starting nodes. This can be useful for capturing fine-grained, local structures in the graph. GraphSAINT increases the overall performance of a graph model both in terms of accuracy and training time.

2.8 Bidirectional Encoder Representations from Transformers (BERT)

BERT is a Language Model (LM) created by Google researchers, (Devin et al., 2018) to improve the fine-tuning approaches used in pre-trained language representations, such as OpenAI Generative Pre-trained Transformer (GPT). OpenAI GPT uses a unidirectional architecture, where only the previous tokens are the main focus in the self-attention layers of the Transformer (Vaswani et al., 2017).

To mitigate this unidirectionality problem, BERT introduces a "Masked Language Model" (MLM) pre-training method that masked the input tokens randomly. The goal is to predict the original Vocabulary ID of the masked word at its context

level only, which is different from standard LM architecture using a left-to-right unidirectional LM pre-training approach.

BERT uses same architecture in both pre-training and fine-tuning approaches in pre-training the LMs. All the different NLP down-stream tasks utilize the same pre-trained model parameters to initialize the model.

During fine-tuning process, all fine-tuning parameters are being fine-tuned. A special token [CLS] is added in front of each text input, and another special separator token [SEP] is used for separating 2 texts, such as in the questions-answers pairs.

To enable BERT to handle various NLP tasks, a BERT input or output representation can represent both a sentence and a pair of 2 sentences in 1 single token sequence clearly. A sentence in a BERT model refers to an arbitrary span of contiguous text, instead of an actual linguistic sentence. A typical BERT input representation is the summarization of its token embeddings, segmentation embeddings, and position embeddings.

BERT MLM method is used to pre-train a deep bidirectional Transformer, which itself is an Encoder only Transformer which is different from the standard Transformer that consists of a Decoder. BERT also introduces a "Next Sentence Prediction" (NSP) task that can jointly pre-trains text-pair representations. BERT improved in 11 state of the art (SOTA) NLP tasks, which include advancing the GLUE score to 80.5%, MultiNLI accuracy to 86.7%, SQuAD v1.1 question answering Test F1 to 93.2 and SQuAD v2.0 Test F1 to 83.1 (Devin et al., 2018).

3 Methodology

We divided our proposed methodology into 5 major processes as shown in Figure 1 below (Chou, 2020). The 5 processes are Topic modeling, Graph Construction, Feature Extraction, Feature Fusion, and Text Classification.

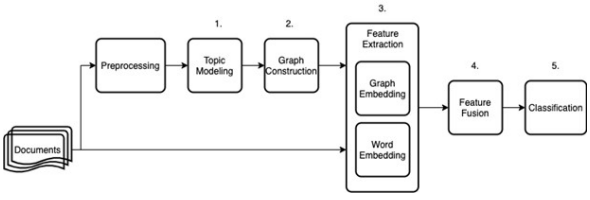

Figure 1. Methodology of TopicGraph model.

The data go through a round of data preprocessing before the 5 processes. We filtered out the lesser important words from the dataset by removing the punctuations and using stopwords from spaCy. After data preprocessing, we train our topic model using a LDA model to extract the topics distribution and construct a word-topic graph. For feature extraction, we utilize the GCN-GAT framework to extract the graph embedding whereas for word embedding, we use Word2Vec and RNN. By concatenating both the word-topic graph and word embedding during feature fusion, we extract the final embedding to perform supervised text classification.

3.1 Topic Modeling

For Topic Modeling, we train a LDA model to extract the most relevant topics from a text document. For example, the word "Genetics" may be the most significant word in a text document containing the various text consisting of "human", "genome", and "DNA" and so forth.

The LDA model is able to find the topic and its related relevant words. These features are a kind of global information of a text document which we can treat it as a form of word co-occurrence. For example, when we have words like "human", "genome", and "DNA" appearing in a text document at the same time, we may assume that the text document is related to the topic "Genetics". In this study, we try to introduce these features for constructing a graph.

3.2 Graph Construction

To construct our graph, first we input our text sequence into the LDA model to find the most probable topic for each word, *p(topic|word)* in the text sequence as shown in the Figure 2 below (Chou, 2020).

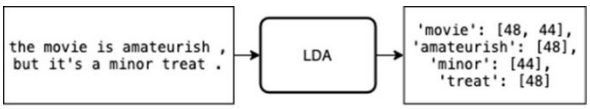

Figure 2. LDA topic modeling example.

For our example in the above Figure 2 which we set a maximum K topic of 50, the input text sequence "the movie is amateurish, but it's a minor treat." may generate for the word "movie" with 2 topics [48, 44], "amateurish" with 1 topic [48] and so forth.

Based on the above results, we construct a graph using the topics and words as nodes, and the

connected relationships between the topics and words as edges as shown in Figure 3 below (Chou, 2020).

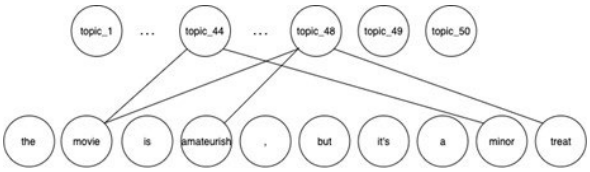

Figure 3. LDA topic modeling nodes example.

From the above example in Figure 3, we can see that the relationship between the words in a text document. At the same time, we can further observe that for the word node, its 1-hop neighborhood is its relationship with its topic feature whereas the 2-hop neighborhood is its relationship with other related words with the same topics in the text sequence. We hope that by using these 2 connected relationship features, the model can learn new features from the text.

We use our Topic model that computes topic classification and utilizes the topic node which has an edge with the word node. A text sequence is converted into word embeddings using a pre-trained Word2Vec file before inputting into the bi-directional RNN layer with GRU activation unit. Following that, the topics-word model is computed using LDA with to generate the topics-word embeddings.

At the same time, the graph embeddings will be generated by a 2-layer GCN together with a GAT layer. Once this is done, both the word and graph embeddings go through a fully connected linear layer before concatenating to generate the final output, which is a 100-dimensional word embedding file.

The text input sequence goes through several transformations in the TopicGraph model, which is listed as follows:

1) Extract forward and backward relationship of the text sequence using GRU.
2) Construct node embeddings using topic features from LDA and sequence features from GRU.
3) Reduce the GRU concatenated hidden features dimensionality using linear layers.
4) Extract the graph node embeddings from GRU concatenated hidden features using GAT layers and graph linear layer.
5) Concatenate the sequence and graph embeddings without the topic nodes.
6) Get the unnormalized output logits after the merged embeddings passed through a linear layer.
7) The output logits are aggregated to produce the final prediction output.
8) Compute the topic-specific output logits.
9) The model output is the final prediction output logits and the topic-specific logits.

3.3 Feature Extraction

Our feature extraction process consists of 3 methods as follows:

1) Using a pre-trained Word2Vec model directly.
2) Using RNN to find word embeddings that include contextual relationships.
3) Using GNN-GAT framework to find word embeddings.

For RNN feature extraction, we used a Bidirectional RNN (Schuster and Paliwal, 1997) to extract the contextual relationships of a word in text sequence as shown in Figure 4 below (Chou, 2020). In this study, each input text sequence is a vector converted by Word2Vec.

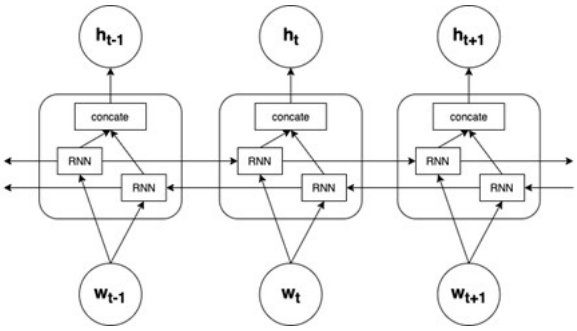

Figure 4. Schematic diagram of Bidirectional RNN.

GAT is a NN network that is applied in a graph architecture. It combines GCN with a self-attention mechanism (Bahdanau et al., 2015; Vaswani et al., 2017). When calculating the weights of the nodes in a graph, GAT focus on the importance of its neighboring nodes to update the weights of its own node, as shown in Eq. (1) and Eq. (2) below (Chou, 2020):

$$X' = GAT(X),\ x_i' = \alpha_{i,i}\Theta x_i + \sum_{j\in N(i)} \alpha_{i,j}\Theta x_j \quad (1)$$

$$\alpha_{i,j} = \frac{\exp\left(LeakyReLU(\vec{a}^T[\Theta x_i || \Theta x_j]\right)}{\sum_{k\in N_i}\exp\left(LeakyReLU(\vec{a}^T[\Theta x_i || \Theta x_k]\right)} \quad (2)$$

As shown in Figure 3 above, each text sequence is construct into a word-topic graph structure. The nodes are divided into two categories, topic nodes and word nodes. The features of the word nodes are used as the initial features by the vector calculated by Bidirectional RNN whereas the topic nodes are encoded with one-hot encoding. Since both topic nodes and word nodes are different in their features, different linear transformations applied here, as such Eq. (1) above is slightly modified into Eq. (3) as shown below (Chou, 2020):

$$x_i' = \alpha_{i,i}\Theta x_i + \sum_{j\in N(i)} \alpha_{i,j}\Theta x_j, \Theta = \begin{cases} \Theta_{word}, & x \text{ is word node} \\ \Theta_{topic}, & x \text{ is topic node} \end{cases} \quad (3)$$

Unlike RNN, which captures text features from the context, we expect GAT to capture different text features. From Figure 3 above, the text under the same topic will add topic features, such as "maturish" and "treat". As compared with the other words, "movie" belongs to two topics, so it will have a connection between the two topic features. Because the graph is undirected, the relationship between the topic feature and text features are also added. For example, topic_48 in Figure 3 above will add three text features. Although LDA can calculate the text distribution under each topic which is the importance of a word under a certain topic, we do not use the calculated values obtained by LDA, but instead we use GAT to learn new weights, which is more relevant in our target dataset.

3.4 Feature Fusion

The overall architecture of TopicGraph is as shown in Figure 5 below (Chou, 2020).

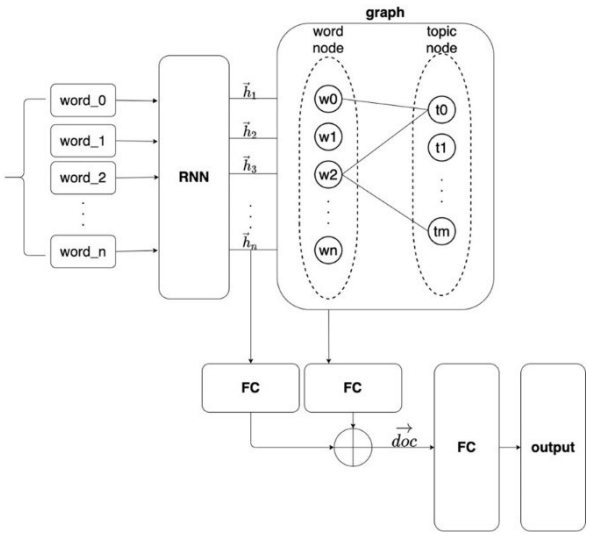

Figure 5. Overall Architecture of TopicGraph.

During feature fusion, we first extracted the vectors h of all words in a text sequence, n is the total number of words in the text sequence. We input the vectors in two directions into the model. First, the vectors are inputted into the graph that we constructed which allow each text to learn a new vector. Then we output the newly learned graph features into a Fully connected layer (FC). In this step, we define the node characteristics x in Eq. (4) and Eq. (5) as follows (Chou, 2020):

$$x_i = \vec{h}_i \quad \forall i \in \{1, 2, \dots, n\}, \; x_j = \overrightarrow{topic}_k \quad \forall j \in \{n+1, n+2, \dots, n+k\} \quad (4)$$

$$g = tanh\left(\Theta_{graph}\, GAT(x)\right), \quad g = \{\vec{g}_1, \vec{g}_2, \dots, \vec{g}_{n+k}\} \quad (5)$$

GAT(x) is a function for updating node features, and the algorithm is as shown in Eq. (3) above.

The other direction of the inputted vector, the learned text feature calculated from RNN is directly connected to FC, as shown in Eq. (6) as follows (Chou, 2020):

$$h' = tanh(\Theta_{\text{rnn}}\, h) \quad (6)$$

According to Eq. (5) and Eq. (6), we connect a layer of FC to re-extract the features learned by RNN and GAT, then we obtain two different vectors h' and g. In order to obtain the vectors of the entire text sequence, we concatenate h' and g together and get the average value. In this study, only the newly extracted n word embeddings are

used. Although using the average value cannot capture important features as well as the attention technique, it has good results in terms of performance and efficiency with no additional parameters required (Shen et al., 2018), as shown in Eq. (7) as follows:

$$\overrightarrow{doc} = \frac{1}{n}\sum_{i=1}^{n} concatenate\left(\overrightarrow{h'}_i, \vec{g}_i\right) \quad \forall i \in \{1,2,\dots,n\} \tag{7}$$

3.5 Text Classification

The training parameters for the entire model are calculated using the Gradient Descent method. In this study, we use cross-entropy as our loss function for text classification. We need to minimize two losses, one is the Softmax loss and the other is the loss that are used for Topic classification. We hope that the trained topic features can capture some important semantic features that is useful in text classification.

4 Experiments

4.1 Dataset

We use the 4 datasets used by TextGCN model (Yao et al., 2019) which are MR, R8, R52 and Ohsumed for model performance and comparison purposes. All the datasets consist of English paragraphs with different classification labels that are meant for different NLP subtasks.

4.2 Model Evaluation

In order to evaluate the effect of the topic model on the entire model, we use two different evaluation methods. Topic coherence score is used in the topic model to evaluate the quality of the topic whereas for text classification, we use accuracy for classification evaluation. At the same time, in order to determine the stability of the model, t-test was used for statistical testing.

4.3 Experimental Settings

For topics modeling, we set the maximum number of topics to 100, with an interval of 10 plus the number of classes in the dataset. This will allow us to observe the effect of different number of topics on the dataset. The maximum epoch is set to 200 with early stopping level set at 10 epochs. The Learning Rate is set to 0.001 which is the result of preliminary observation during the experiment where we observed that validation loss decreases steadily. The number of graph convolution layer is set to 2 which we referenced from other studies (Hamilton et al., 2017; Kipf and Welling, 2017; Veličković et al., 2018).

4.4 Experimental Results

We show the model comparison of text classification results between our proposed model in this paper with other models (Chou et al., 2020). The comparison result is as shown in Table 1 below.

Model	R8	R52	Ohsumed	MR
TF-IDF+LR	0.9347	0.8695	0.5466	0.7459
LDA+LR	0.8149	0.7118	0.2537	0.5371
CNN-non-static	0.9571 ± 0.0052	0.8759 ± 0.0048	0.5844 ± 0.0106	0.7775 ± 0.0072
Bi-LSTM	0.9631 ± 0.0033	0.9054 ± 0.0091	0.4927 ± 0.0107	0.7768 ± 0.0086
PV-DBOW	0.8587 ± 0.0010	0.7829 ± 0.0011	0.4665 ± 0.0019	0.6109 ± 0.0010
PV-DM	0.5207 ± 0.0004	0.4492 ± 0.0005	0.2950 ± 0.0007	0.5947 ± 0.0038
PTE	0.9669 ± 0.0013	0.9071 ± 0.0014	0.5358 ± 0.0029	0.7023 ± 0.0036
fastText	0.9613 ± 0.0021	0.9281 ± 0.0009	0.5770 ± 0.0049	0.7514 ± 0.0020
SWEM	0.9532 ± 0.0026	0.9294 ± 0.0024	0.6312 ± 0.0055	0.7665 ± 0.0063
Bi-GRU-avg	0.9702 ± 0.0036	0.9235 ± 0.0072	0.6135 ± 0.018	0.7844 ± 0.0057
LEAM	0.9331 ± 0.0024	0.9184 ± 0.0023	0.5858 ± 0.0079	0.7695 ± 0.0045
Text GCN	0.9707 ± 0.0010	0.9356 ± 0.0018	0.6836 ± 0.0056	0.7674 ± 0.0020
Dep-GAT-root	0.9654 ± 0.0025	0.9263 ± 0.0062	0.6194 ± 0.0118	0.7942 ± 0.0059
Dep-GAT-avg	0.9611 ± 0.0075	0.9229 ± 0.0066	0.5630 ± 0.0146	0.7839 ± 0.0029
Ours				
TopicGraph	**0.9754 ± 0.0020**	**0.9427 ± 0.0048**	0.6362 ± 0.0104	**0.7933 ± 0.0041**
TopicGraph (without GRU)	0.9146 ± 0.0023	0.8022 ± 0.0096	**0.7271 ± 0.0121**	0.7573 ± 0.0091

Table 1: Classification model comparison.

The results show that our models can achieve good results on R8, R52 and MR datasets with an accuracy score of 0.9754, 0.9427 and 0.7933 respectively. RNN can get good results on shorter texts (Cho et al., 2014) and the features learned by the graph model help to improve text classification as well.

However, our TopicGraph model did not perform better than the TextGCN model in the Ohsumed dataset. The reason may be that our proposed model TopicGraph is not effective in classifying longer texts. In order to test whether RNN has indeed lower our model text classification performance, we did an ablation test

to remove the GRU module from the overall model as shown in Figure 6 below (Chou, 2020).

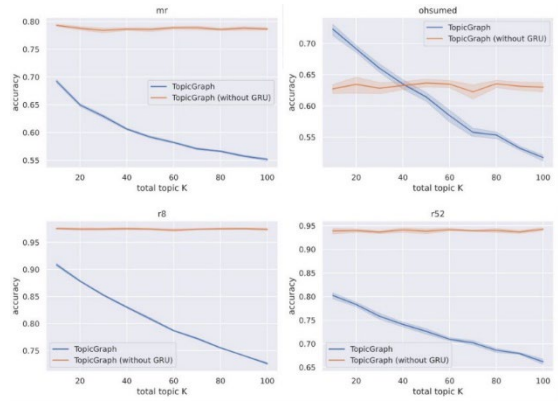

Figure 6. Ablation test by removing GRU.

From the above Figure 6, the TopicGraph (without GRU) model is added with sequential embedding of the RNN model. As compared with the TopicGraph model, we observed there is an increase in overall accuracy of the entire model and at the same time, the entire model accuracy is not affected by the numbers of topics in the Topic model.

Whereas for the Ohsumed dataset, we observed there is a decrease in the accuracy as compared to the TopicGraph model. However, if we removed the RNN embeddings, we observe that there is a significant improved results from the previous accuracy score of 0.6362 to 0.7271.

From these observations, we realized that our proposed model performed better on shorter text dataset R8, R52 and MR but not on longer text dataset Ohsumed. This also shown that RNN models can RNN perform better on shorter texts (Cho et al., 2014).

Although using our TopicGraph (without GRU) can improve in performance on Ohsumed dataset, but it could not get better classification results when comparing with TextGCN model which uses word co-occurrence and word-document relation to build graphs.

This may be due to the following:

1) Our TopicGraph (without GRU) model does not contain sequential relationship in between texts.
2) Our TopicGraph (without GRU) model does not obtain good topic words on shorter text document.

4.5 Further Improvement on TopicGraph Model

To improve our model performance, we use the subgraph-wise sampling technique, GraphSAINT (GS) to extract additional contextual features. We then use the newly extracted features to input into a fine-tuned BERT base model (uncased) with 2 NN layers for text classification with improved results. We show our modified TopicGraph-GS-BERT model results as shown in Table 2 below.

From the results in the Table 2 below, we can see there are improvement in performance for shorter text dataset R8, R52 and MR by 0.08%, 1.71% and 12% respectively. Although the modified TopicGraph-GS-BERT model increased in performance against TopicGraph model by 5.06%, but it performed lower than the TopicGraph (without GRU) by 1.01%. We think this is due to our TopicGraph model low ability to classify longer text. The (-) in Table 2, Table 3 below denote that the corresponding GS sampler was not used as the extracted features are not the optimal out of the 3 GS samplers, and (*) denote no results.

Model	R8	R52	Ohsumed	MR
TextGCN	0.9707 ± 0.0010	0.9356 ± 0.0018	0.6836 ± 0.0056	0.7674 ± 0.0020
Dep-GAT-root	0.9654 ± 0.0025	0.9263 ± 0.0062	0.6194 ± 0.0118	0.7942 ± 0.0059
Dep-GAT-avg	0.9611 ± 0.0075	0.9229 ± 0.0066	0.5630 ± 0.0146	0.7839 ± 0.0029
PMI-GAT-avg	0.9667 ± 0.0041	0.9256 ± 0.0096	0.5512 ± 0.0578	0.7606 ± 0.0059
Ours				
TopicGraph (without GRU)	0.9146 ± 0.0023	0.8022 ± 0.0096	**0.7271** ± 0.0121	0.7573 ± 0.0091
TopicGraph	0.9754 ± 0.0020	0.9427 ± 0.0048	0.6362 ± 0.0104	0.7933 ± 0.0041
TopicGraph-GS$_{rw}$-BERT	0.9690	0.9578	0.6664	**0.9133**
TopicGraph-GS$_{node}$-BERT	**0.9762**	**0.9661**	-	-
TopicGraph-GS$_{edge}$-BERT	-	-	0.7170	-

Table 2: Classification model comparison 2.

We conduct an ablation test to check the if using GraphSAINT to extract additional contextual features and using a fine-tuned BERT model for text classification did indeed impact the overall model performance. We show our ablation test results as shown in Table 3 below.

Model	R8	R52	Ohsumed	MR
Ours				
TopicGraph (without GRU)	91.46	80.22	**72.71**	75.73
TopicGraph	97.54	94.27	63.62	79.33
TopicGraph-GS$_{node}$	95.98	90.07	51.30	75.32
TopicGraph-GS$_{edge}$	*	*	51.62	*
TopicGraph-GS$_{rw}$	86.48	85.05	45.96	77.04
TopicGraph-BERT	96.52	94.50	65.76	86.04
TopicGraph-GS$_{rw}$-BERT	96.90	95.78	66.64	**91.33**
TopicGraph-GS$_{node}$-BERT	**97.62**	**96.44**	-	-
TopicGraph-GS$_{edge}$-BERT	-	-	71.70	-

Table 3: TopicGraph-GS-BERT Ablation test.

We made the following observations from the ablation test above:

1) The TopicGraph-GS (without TopicGraph and BERT) model did not perform better than the TopicGraph model. At the same time, the accuracy scores of TopicGraph-GS-BERT also dropped by 1.64% to 20.08% for the 4 datasets. This show that TopicGraph and BERT contain relevant and important semantic information.
2) The TopicGraph-BERT (without GS) model perform better in 3 of the datasets except R8 dataset. At the same time, the accuracy scores of TopicGraph-GS-BERT also dropped by 1.1% to 5.94% for the 4 datasets. This show that GS is an important component of the modified TopicGraph-BERT model.
3) The TopicGraph-GS-BERT model perform better in 3 datasets except Ohsumed dataset. At the same time, the accuracy scores of TopicGraph-GS-BERT increased by 0.08% to 12% for the 3 datasets.

From the ablation test results, TopicGraph-GS-BERT model did indeed improve the overall performance of the TopicGraph on shorter text dataset R8, R52 and MR, but not in the longer text dataset Ohsumed.

5 Conclusion

In this study, we combine topic model and GNN to classify text. We first use the topic model to capture the co-occurrence features of text. Then we represent both text and topics as nodes in a graph, and combining the features of text and topics to learn new word embeddings. Because our TopicGraph model considered the text as a whole unit when constructing a graph, we can easily add a sequential model to capture additional contextual features and rebuilding of the graph is not required when predicting new data.

We also observe the impact of topics on text classification performance. As the number of topics increases, the overall performance of text classification and topic coherence will also decrease. For GNN, increasing the number of convolution layers will degrade the model performance and using a 2-hop neighborhood feature usually has a better text classification result. Adding contextual features will achieve a better accuracy for shorter text datasets, such as MR, R8 and R52. However, for longer text dataset such as Ohsumed dataset, we can focus on the co-occurrence features and graph features alone.

In order to improve our TopicGraph model performance, we add the subgraph-wise sampling technique, GraphSAINT to extract new features. Using the new features with a fine-tuned BERT-NN model, we are able to further improve our model performance.

One particular weakness in the proposed model is the lack of model comparison with other SOTA models. We intend to look into various established transductive and inductive graph benchmarks for model comparison as one of our main focuses for our future works.

Another shortcoming is the lack of visual proofs and clear visibility of the proposed model strengths in our proposed model. We also noticed our proposed model weakness in classifying longer text document. The other focuses in our future works will be rectification to these shortcomings to further enhance the model capability.

Since our proposed model work on four datasets, testing our proposed model framework on different datasets will be another challenge which we will focus in the future.

Acknowledgments

The authors would like to thank the supports from the National Science and Technology Council, Taiwan under the grant numbers: NSTC113-2221-E-027-096, and NSTC113-2634-F-027-001-MBK. The authors acknowledge and thank Kuan-Hsun Chou for his contribution and research works on the design framework and technical analysis of the initial TopicGraph model.

References

Bahdanau, D., Kyunghyun, C., & Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. *International Conference on Learning Representations.* San Diego, California, US.

Blei, D., Ng, A., & Jordan, M. (2003). Latent Dirichlet Allocation. *The Journal of Machine Learning Research, Volume 3*, 993 - 1022.

Cho, K., Merriënboer, B. v., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724–1734). Doha, Qatar: Association for Computational Linguistics.

Chou, K.-H. (2020). A Method Combining Topic and GNN Models for Text Classification.

Chou, K.-H., Wu, Y.-T., & Wang, J.-H. (2020). Combining Dependency Parser and GNN Models for Text Classification. *Proceedings of the 32nd Conference on Computational Linguistics and Speech Processing (ROCLING 2020)* (pp. 50–58). Taipei, Taiwan: The Association for Computational Linguistics and Chinese Language Processing (ACLCLP).

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4171–4186). Minneapolis, Minnesota: Association for Computational Linguistics.

Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks.* Springer.

Hamilton, W., Ying, R., & Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. *Advances in Neural Information Processing Systems 30 (NIPS 2017).* Long Beach, California, US.

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1746–1751). Doha, Qatar: Association for Computational Linguistics.

Kipf, T., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. *International Conference on Learning Representations.* Toulon, France.

Liu, P., Qiu, X., & Huang, X. (2016). Recurrent Neural Network for Text Classification with Multi-Task Learning. *25th International Joint Conference on Artificial Intelligence IJCAI-16.* New York.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *International Conference on Learning Representations.* Scottsdale, AZ, USA.

Schuster, M., & Paliwal, K. (1997). Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing (Volume: 45, Issue: 11, November 1997)*, 2673-2681.

Shen, D., Wang, G., Wang, W., Min, M., Su, Q., Zhang, Y., . . . Carin, L. (2018). Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 440–450). Melbourne, Australia: Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., . . . Polosukhin, I. (2017). Attention Is All You Need. *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, 6000-6010.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph Attention Networks. *International Conference on Learning Representations.* Vancouver, BC, Canada.

Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., . . . Zhang, Z. (2019). Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *ArXiv*.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. (2021). A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions*

on Neural Networks and Learning Systems (Volume: 32, Issue: 1, January 2021), 4-24.

Yao, L., Mao, C., & Luo, Y. (2019). Graph Convolutional Networks for Text Classification. *33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*, (pp. 7370-7377). Honolulu, Hawaii, US.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., & Prasanna, V. (2020). GraphSAINT: Graph Sampling Based Inductive Learning Method. *International Conference on Learning Representations.* Online.