ME-GCN: Multi-dimensional Edge-Embedded Graph Convolutional Networks for Semi-supervised Text Classification

Anonymous ACL submission

Abstract

Compared to sequential learning models, graph-based neural networks exhibit excellent ability in capturing global information and have been used for semi-supervised learning tasks, including citation network analysis or text classification. However, most GCNs are designed with the single-dimensional edge feature and neglected to utilise the rich edge information about graphs. In this paper, we introduce the ME-GCN (Multi-dimensional Edge-enhanced Graph Convolutional Networks) for semi-supervised text classification. A text graph for an entire corpus is firstly constructed to describe the undirected and multi-dimensional relationship of word-to-word, document-document, and word-to-document. The graph is initialised with corpus-trained multi-dimensional word and document node representation, and the relations are represented according to the distance of those words/documents nodes. Then, the generated graph is trained with ME-GCN, which considers the edge features as multi-stream signals, and each stream performs a separate graph convolutional operation. Our ME-GCN can integrate a rich source of graph edge information of the entire text corpus. The results have demonstrated that our proposed model has significantly outperformed the state-of-the-art methods across eight benchmark datasets.

1 Introduction

Deep Learning models, such as Recurrent Neural Networks (RNN) or Transformer, have performed well and have been widely used for text classification. However, the performance is not always satisfactory when utilising small labelled datasets. In many practical scenarios, the labelled dataset is very scarce as human labelling is time-consuming and may require domain knowledge. There is a pressing need for studying semi-supervised text classification with a relatively small number of labelled training data in deep learning paradigm. For the successful semi-supervised text classification, it is crucial to maximize effective utilization of structural and feature information of unlabelled data.

Graph Neural Networks (GNN) have recently received lots of attention as it can analyse rich relational structure, prioritize global features exploitation, and preserve global structure of a graph in graph embeddings. Due to these benefit, there have been some successful attempts to revisit semi-supervised learning with Graph Convolutional Networks (GCN) (Kipf and Welling, 2017). TextGCN (Yao et al., 2019) initialises the whole text corpus as a document-word graph and applies GCN for text classification. It shows potential of GCN-based semi-supervised text classification. Linmei et al. (2019) worked on semi-supervised short text classification using GCN with topic-entity, and Liu et al. (2020) proposed tensorGCN with semantic, syntactic, and sequential information.

One major problem in those existing GCN-based text classification models is that edge features are restricted to be one-dimensional, which are the indication about whether there is edge or not (e.g. binary connectedness) or often one-dimensional real-value representing similarities (e.g. pmi, tf-idf). Instead of being a binary indicator variable or a single-dimensional value, edge features can possess rich information and fully incorporated by using multi-dimensional vectors. Addressing this problem is likely to benefit several graph-based classification problems but particularly important for the text classification task. This is because the relationship between words and documents can be better represented in a multi-dimensional vector space rather than a single value. For example, word-based vector space models embed the words in a vector space where similarly defined words are mapped near to each other. Rather than using the lexical-based syntactic parsers or additional resources, words that share semantic or syntactic relationships will be represented by vectors of sim-

ilar magnitude and be mapped in close proximity to each other in the word embedding. Using this multi-dimensional word embedding as node and edge features, it would be more efficient to analyse rich relational information and explore global structure of a graph. Then, what would be the best way to exploit edge features in a text graph convolutional network? According to the recently reported articles (Gong and Cheng, 2019; Khan and Blumenstock, 2019; Huang et al., 2020; Liu et al., 2020), more rich information should be considered in the relations in the graph neural networks.

In this paper, we propose a new multi-dimensional edge enhanced text graph convolutional networks (ME-GCN), which is most suitable for the semi-supervised text classification. Note that the focus of our semi-supervised text classification task is on small proportion of labelled text documents with no other resource, i.e. no pre-trained word embedding or language model, syntactic tagger or parser.

We construct a single textual large graph from an entire corpus, which contains words and documents as nodes. The graph describes the undirected and multi-dimensional relationship of word-to-word, document-document, and word-to-document. Each word and document are initialised with corpus-trained multi-dimensional word and document embedding, and the relations are represented based on the semantic distance of those representations. Then, the generated graph is trained with ME-GCN, which considers edge features as multi-stream signals, and each stream performs a separate graph convolutional operation. We conduct experiments on several semi-supervised text classification benchmark datasets. The proposed model can achieve strong text classification performance with a small proportion of labelled documents with no additional resources. The main contributions are as follows:

1) To the best of our knowledge, this is the first attempt to apply multi-dimensional edge features on GNN for text classification.

2) ME-GCN[1] is proposed to use corpus-trained multi-dimensional word and document-based edge features for the semi-supervised text classification.

3) Experiments are conducted on several benchmark datasets to illustrate the effectiveness of ME-GCN for semi-supervised text classification.

[1] An overview architecture is presented in the Appendix.

2 Related Works

2.1 Semi-supervised text classification

Due to the high cost of human labelling and the scarcity of fully-labelled data, deep learning based semi-supervised models have received lots of attention in text classification. Latent variable models (Chen et al., 2015) apply topic models by user-oriented seed information and infer the documents' labels based on category-topic assignment. The embedding-based model (Tang et al., 2015; Meng et al., 2018) utilise seed information to derive text (word or document) embeddings for documents and labels for text classification. Yang et al. (2017) leveraged sequence-to-sequence Variational AutoEncoders (VAEs) model on text classification and sequential labelling. Miyato et al. (2017) utilized adversarial and virtual adversarial training to the text domain by applying perturbations to the word embeddings. Recently, graph convolutional networks (GCN) have been popular in semi-supervised learning as it shows superior global structure understanding ability.(Kipf and Welling, 2017).

2.2 GNN for Text Classification

Graph Neural Networks have received lots of attention and successfully used in various NLP tasks (Bastings et al., 2017; Tu et al., 2019; Cao et al., 2019; Xu et al.). Yao et al. (2019) proposed the Text Graph Convolutional Networks by applying a basic GCN (Kipf and Welling, 2017) to the text classification task. In their work, a text graph for the whole corpus is constructed; word and document nodes are initialised with one-hot representation and edge features are represented as one-dimensional real values, such as PMI, TF-IDF. Several studies have attempted multiple different graph alignments using knowledge graph or semantic/syntactic graph. Vashishth et al. (2019) applied GCN to incorporate syntactic/semantic information for word embedding training. Cao et al. (2019) proposed an alignment-oriented knowledge graph embedding for entity alignment. TensorGCN (Liu et al., 2020) proposed semantic, syntactic, and sequential contextual information. In their framework, multiple aspect graphs are constructed from external resources, and those graph are jointly trained. However, our model ME-GCN constructs and trains multi-dimensional node and edge features alone based on the given text corpus.

3 ME-GCN

We propose the Multi-dimensional Edge-enhanced Graph Convolutional Networks (ME-GCN) for semi-supervised text classification. Note that all graph components are only based on the given text corpus without using any external resources. We utilize the GCN as a base component, due to its simplicity and effectiveness. In this section, we first give a brief overview of GCN and introduce details of how to construct our corpus-based textual graph from a given text corpus. Finally, we present ME-GCN learning model.

GCN Graph A GCN (Kipf and Welling, 2017) is a generalised version of the convolutional neural networks for semi-supervised learning that operates directly on the graph-structured data and induces embedding vectors of nodes based on properties of their neighbourhoods. Consider a graph $G = (V, E, A)$, where $V(|V| = N)$ is the set of graph nodes, E is the set of graph edges, and $A \in R^{N \times N}$ is the graph adjacency matrix.

3.1 Textual Graph Construction

We first describe how to construct a textual graph that contains word/document node representation and multi-dimensional edge features for a whole text corpus. We apply a straightforward textual construction approach that treats words and documents as nodes in the graph. Unlike Yao et al. (2019), we have three types of edges, namely word-document edge, word-word edge, and document-document edge with the aim to investigate all possible relations between nodes. Formally, we define a ME-GCN graph $G_{ME} = (V, E^{(t)}, ME^{(t)})$, where t denotes the t^{th} dimensional edge, $V(|V| = N)$ is the set of graph nodes of word/document, $E^{(t)}$ are the set of graph edges, which can be one of the three types, and $ME^{(t)}$ is the set of adjacency matrix at the t^{th} dimension. The details of node and edge features construction are presented as follows.

3.1.1 Textual Node Construction

From an entire textual corpus, we construct word and document nodes in a graph so that the global word and document distance can be explicitly modeled and graph convolution can be easily adapted. ME-GCN considers the word and document nodes as components for preserving rich information and representing the global structure of a whole corpus, which can fully support for the successful semi-supervised text classification. With this in mind, ME-GCN trains word/node feature by using a Word2Vec (Mikolov et al., 2013) for word nodes, and a Doc2Vec (Le and Mikolov, 2014) for document nodes. For instance, Word2Vec takes as its input a whole corpus of words, and the trained word vectors are positioned in a vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space. This is well-aligned with the role of graph neural networks, representing the global structure of the corpus, and preserving rich semantic information of the corpus. Most importantly, those word/document embeddings are distributed representations of text in an T-dimensional space so the distance between words and documents can be represented as a multi-dimensional vector. Formally, the word/document node features in ME-GCN are initialised as follows. Note that the negative sampling is applied to reduce the training time.

Word Node Construction We train the Word2Vec CBOW (Mikolov et al., 2013) using context words to predict the centre word. Assume we have a given text corpus consisting of K documents and U unique words. The input is a set of context words X_{ik} in document $k \in K$ encoded as one-hot vector of size U. Then the hidden layer H and output layer $Output$ are formulated in equation (1) and (2), in which $W_{U \times T}$ and $W'_{T \times U}$ are two projection matrix. After training, we extract the U vectors of dimension T from the updated matrix $W_{U \times T}$ representing the corresponding U unique words in the whole corpus.

$$H = \sum_{i=1}^{C} X_{ik} W_{U \times T} \quad (1)$$

$$Output = H W'_{T \times U} \quad (2)$$

Document Node Construction Doc2Vec CBOW (Le and Mikolov, 2014) is essentially the same as Word2Vec. In Doc2Vec, we feed the context words X_{ik} together with the current document k to the model, which is also encoded as one-hot vector based on the document id, and the vector size becomes $\hat{U} = U + K$. We have the projection matrix $W_{\hat{U} \times T}$ containing $U + K$ vectors. After training, those K vectors in the updated $W_{T \times \hat{U}}$ are used for representing the corresponding K document.

$$H = D_k W_{\hat{U}_\times T} + \sum_{i=1}^{C} X_{ik} W_{\hat{U} \times T} \quad (3)$$

$$Output = H W'_{T \times \hat{U}} \quad (4)$$

3.1.2 Multi-dimensional Edge Construction

In this section, we describe how to construct a multi-dimensional edge feature in a graph. A traditional textual graph edge (Yao et al., 2019) was based on word occurrence in documents (document-word edges), and word co-occurrence in the whole corpus (word-word edges), however, the occurrence information is not enough to extract how close two pieces of text are in both surface proximity and meaning. According to Mikolov et al. (2013); Kusner et al. (2015), the distance between word/document embeddings learn semantically meaningful representations for words from local co-occurrences in sentences. Inspired by this, we utilise the distance between word/document embeddings to preserve the rich semantic information captured edges, which are also presented as multi-dimensional vectors. To represent all possible edge types, we propose three types of edges: word-word edges, document-document edges, and word-document edges. Our goal is to incorporate the semantic similarity between individual node pairs (each unique word and document) into multi-dimensional edge features. One such measure for word/document node similarity is provided by their Euclidean distance in the Word2Vec or Doc2Vec embedding space. We separately use each dimension space in the node feature (Word2Vec/Dec2Vec) for representing each of the dimension in the multi-dimensional node edge. Thus, we will have T dimensional edges between nodes of T dimensional features and each $t \in \{1, 2, ..., T\}$ is represented by one dimensional Euclidean distance calculation in the t^{th} dimensional space. This edge calculation method is applied to word-word and doc-doc edge features.

Word-Word Edge Feature We draw on the learned semantics in each feature dimension of the word embedding of size T to calculate the edge weight for each dimension. Concretely, the T-dimensional word-word edge $E^{(t)}_{w_i,w_j}, t \in \{1, 2, ..., T\}$ between word i and word j is formulated as in equation (5), in which $W^{(t)}_i$ and $W^{(t)}_j$ represents the feature value at the dimension t of the word embedding W_i for word i and W_j for word j respectively. The denominator calculates the distance of the two words regarding dimension t and $tanh(^{-1})$ is used for normalization.

$$E^{(t)}_{w_i,w_j} = tanh\frac{1}{|W^{(t)}_i - W^{(t)}_j|} \quad (5)$$

Doc-Doc Edge Feature The document-document edge is constructed in a way similar to the word-word edge. As is shown in equation (6), the T-dimensional document-document edge $E^{(t)}_{d_i,d_j}$ is calculated based on the normalized Euclidean distance between the values $D^{(t)}_i$ and $D^{(t)}_j$ at each dimension t of the features for document i and j.

$$E^{(t)}_{d_i,d_j} = tanh\frac{1}{|D^{(t)}_i - D^{(t)}_j|} \quad (6)$$

Word-Doc Edge Feature We use the same calculation method for a single-dimension word-document edge as in TextGCN while repeating it for each dimension t. Thus, the T-dimensional word-document edge $E^{(t)}_{w_i,d_j}$ is simply represented as the TF-IDF value of word i and document j. This is repeated for each dimension t, as is formulated in equation (7). We also found using TF-IDF weight is better than using term frequency only.

$$E^{(t)}_{w_i,d_j} = \text{TF-IDF}_{w_i,d_j} \quad (7)$$

Formally, the multi-dimensional edge weights between node i and j is defined as follows.

$$ME^{(t)}_{ij} = \begin{cases} E^{(t)}_{w_i,w_j} & w_i, w_j \text{ are words} \\ E^{(t)}_{d_i,d_j} & d_i, d_j \text{ are docs, } W_{d_i \cap d_j} \geq u \\ E^{(t)}_{w_i,d_j} & w_i \text{ is word, } d_j \text{ is doc} \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

We noted that the threshold u for the doc-doc edges is not compulsory but efficient for the better computation. The detailed threshold is in Section 4.3.

3.2 ME-GCN Learning

After constructing the multi-dimensional edge enhanced text graph, we focus on applying effective learning framework to perform GCN on the textual graph with multi-dimensional edge features.

The traditional GCN learning takes into the initial input matrix $H^{(0)} \in R^{N \times d_0}$ containing N node features of size d_0. Then the propagation through layers is made based on the rule in equation (9), which takes into consideration both node features and the graph structure in terms of connected edges.

$$H^{(l+1)} = f(H^{(l)}, A) = \sigma(\hat{A}H^{(l)}W^{(l)}) \quad (9)$$

The l and $(l+1)$ represents the two subsequent layers, $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ is the normalized symmetric adjacency matrix $\tilde{A} = A + I$ (I is an identity matrix for including self-connection), $\tilde{D}$ is the diagonal node degree matrix with $\tilde{D}(i,i) = \Sigma_j \tilde{A}(i,j)$, and $W^{(l)} \in R^{d_l \times d_{l+1}}$ is a layer-specific trainable weight matrix for lth layer. d_l and d_{l+1} in-

dicates the node feature dimension for lth layer and $(l+1)$th respectively. σ denotes a non-linear activation function for each layer such as Leaky ReLu/ReLU except for the output layer where softmax is normally used for the classification.

Our goal is to represent the node representation by aggregating neighbour information with each edge features in a multi-stream manner. Hence, we generalize the traditional GCN learning approach to perform multi-stream(MS) learning for the multi-dimensional edge enhanced graph. The overall MS learning procedure is in equation (10), for each node feature in $H^{(l)} \in R^{N \times d_l}$, we will apply the multi-stream GCN learning f_{MS} that formulates t streams of traditional GCN learning through the t dimensions of the connected edge, resulting in the multi-stream hidden feature $H_t^{(l+1)} \in R^{N \times d_{ms}^{(l+1)}}$ at $(l+1)$th layer. Here $t \in \{1, 2, ..., T\}$ and $d_{ms}^{(l+1)}$ is the multi-stream feature size for each edge dimension at this layer. Then a multi-stream aggregation function ϕ_{MS} is applied over the t streams, producing the feature matrix $H^{(l+1)} \in R^{N \times d_{(l+1)}}$ that contains the aggregated feature for each node in N. Here we use $concatenation$ function as ϕ_{MS} for the hidden layer in the multi-stream aggregation, leading us to have $d_{l+1} = t * d_{ms}^{(l+1)}$. Specifically, for the output layer, $pooling$ method is used instead and the details are provided in later paragraph. Accordingly, the updated propagation rule is provided in equation (11). Unlike the original GCN propagation in equation (9), we have T streams of GCN learning in each layer, sharing the same input $H^{(l)}$ and propagating based on the T adjacency matrices $ME^{(t)}$, which involves a set of layer and stream specific trainable weight matrices denoted as $W^{(l)(t)}$. We also tried the shared-stream learning that shares the trainable weight matrices across each stream but found that separate stream-specific trainable weight matrices have better performance. The comparison of the two learning mechanisms is provided in Section 5.2.

$$H^{(l)} \xrightarrow{f_{MS}} H_t^{(l+1)} \xrightarrow{\phi_{MS}} H^{(l+1)} \tag{10}$$

$$\begin{aligned} H^{(l+1)} &= \phi_{MS}(f_{MS}(H^{(l)}, ME^{(t)})) \\ &= \phi_{MS}(\sigma(\hat{ME}^{(t)} H^{(l)} W^{(l)(t)})) \end{aligned} \tag{11}$$

3.2.1 Pooling

Unlike the hidden layers where we use $concatenation$ to aggregate the node features over each stream to continue propagation to next layer, we instead apply the $pooling$ method at the output layer to further synthesize the multi-stream features of each node in order to do the final classification. Equation (12) formulizes $max\ pooling$, in which $H_t^{(l_O)} \in R^{N \times d_{ms}^{l_O}}, t \in \{1, 2, ..., T\}$ denotes the T streams of node features for N nodes at the output layer l_O, and here $d_{ms}^{l_O}$ is the node feature dimension that equals to the classification label number C. Through $max\ pooling$, we select the best valued features over the T streams for each node in N before the final classification. We also tried other $pooling$ methods and provide the comparison in Section 5.2.

$$pooling_{max} = \max_{1 \le t \le T}(H_t^{(l_O)}) \tag{12}$$

4 Evaluation Setup

We evaluate the performance of our ME-GCN on semi-supervised text classification, and carefully examine the effectiveness of corpus-based multi-dimensional edge features.

4.1 Baselines[2]

We aim to compare ME-GCN with state-of-the-art semi-supervised text classification models, which do not use any external resources. Additionally, we also include four baseline models, which use pretrained embedding or language model: CNN-Pretrained, LSTM-Pretrained, BERT, and TMix.

1)TF-IDF+LR, 2)TF-IDF+SVM: Term frequency inverse document frequency for feature engineering with Logistic Regression or SVM with rbf kernel. **3)CNN-Rand, 4)-Pretrained**: TextCNN (Kim, 2014) is used as the classifier. Both CNN-Rand using random initialized word embedding and CNN-Pretrained using pretrained word embedding are evaluated. We used English Glove-pretrained and Chinese Word Vectors (Li et al., 2018) for Chinese dataset-zh. **5)LSTM-Rand, 6)-Pretrained**: We apply the same set-up as the CNN model, but with Long Short-Term Memory (LSTM). **7)TextGCN**: We follow the same hyper-parameters of the TextGCN (Yao et al., 2019). **8)BERT**: BERT (Devlin et al., 2018) is a pre-trained model which has achieved good performance in text classification. We use the $\text{BERT}_{\text{BASE}}$ in our experiments ('bert-base-chinese' model from huggingface is used for Chinese). **9)TMix**: TMix(Chen et al., 2020) generates new training text data by interpolating over labeled text encoded

[2]All baseline related links are provided in Appendix D.

Datasets	# Doc	# Words	# Node	# Class	Avg. length
20NG	3,000	6,095	9,095	20	249.4
R8	3,000	4,353	7,353	8	84.2
R52	3,000	4,619	7,619	52	104.5
Ohsumed	3,000	8,659	11,659	23	132.6
MR	10,662	4,501	15,163	2	18.4
Agnews	6,000	5,360	11,360	4	35.2
Twit nltk	3,000	634	3,634	2	11.5
Waimai(zh)	11,987	10,979	22,966	2	15.5

Table 1: The summary statistics of datasets

using BERT hidden representation and train on the generated text data for text classification. We use the default setting provided in the official github.

4.2 Dataset[3]

We evaluated our experiments on five widely used text classification benchmark datasets (Yao et al., 2019), 20NG, R8, R52, MR and Ohsumed, and three additional semi-supervised text classification datasets (Linmei et al., 2019), Agnews, Twitter nltk and Waimai. All the data is split based on the extreme low resource text classification enviornment-1% training and 99% test set. The summary statistics of the datasets can be found in Table 1. For the data sample selection, we randomly select them but the class distribution is followed by the original datasets. **1)20NG** is a 20-class news classification dataset and we select 3,000 samples from the original dataset. **2)R8**, **3)R52** are from Reuters which is a topic classification dataset with 8 classes and 52 classes. 3,000 samples from each dataset are selected. **4)MR**(Pang and Lee, 2005) is a binary classification dataset about movie comments and we use all samples from the dataset. **5)Ohsumed** is a medical dataset with 23 classes, and we select 3,000 samples from the original dataset. **6)Agnews**(Zhang et al., 2015) is a 4-class news classification dataset and 6,000 samples are selected. **7)Twitter nltk** is a binary classification sentiment analysis from Twitter, we sampled 1,500 positive samples and 1,500 test samples from the original dataset. **8)Waimai** is a binary sentiment analysis dataset about food delivery service comments from a Chinese online food ordering platform. The dataset is in Chinese and pre-tokenized. We use all samples from the original dataset.

4.3 Settings

Before training, words occurring no more than 5 times have been excluded. Both word2vec and Dec2vec are trained on the corpus we get using $gensim$ package with $window\_size = 5$ and $iter = 200$. The initial feature dimension for node and document is set to $d_0 = 25$, which is same to the multi-dimension number for edge features and multi-stream number T in ME-GCN learning. Different multi-stream numbers are tested and discussed in 5.3. The threshold $u = 5$ is used for document-document edge construction. We use two-layers of multi-stream GCN learning with $d_{ms}^{l_1} = 25$ (thus $d^{l_1} = 625$) for the first multi-stream GCN layer and $d_{ms}^{l_O} = C$(no. of label in the datasets) for the output layer. In the training process, following Liu et al. (2020), we use dropout rate as 0.5 and learning rate as 0.002 with Adam optimizer. The number of epochs is 2000 and 10% of the training set is used as the validation set for early stopping when there is no decreasing in validation set's loss for 100 consecutive epochs.

5 Results Analysis

5.1 Performance Evaluation

Table 2 presents a comprehensive performance experiment, conducted on the benchmark datasets. The most bottom row shows the accuracy from our best models using either max or average pooling.[4]

Overall, our proposed model significantly outperforms the baseline models on all eight datasets, demonstrating the effectiveness of our ME-GCN on semi-supervised text classification for various length of text. With in-depth analysis, CNN/LSTM-Rand is quite low in performance on several datasets but increases significantly when using pre-trained embeddings. While TextGCN achieves better accuracy than above baselines on most datasets, the performance is all lower than ME-GCN. This shows the efficiency of preserving rich information using multi-dimensional edge features. The merit of pre-training stands out with BERT and TMix, producing better accuracy than the baseline TextGCN on most datasets. Especially, BERT achieves the best and second best performance on MR and Waimai, which are short-text sentiment analysis datasets. This would be because of the two aspects of sentiment classification: (1) compared to topic-specific text classification, sentiment analysis task may benefit from the pretrained general semantics learned from a large external text; (2) word order matters for sentiment analysis, which could be missing in GNNs. Nevertheless, our ME-GCN, with no external resources, still outperforms those

[3]Source links for all datasets are provided in Appendix D.

[4]The detailed comparison of pooling method variants can be found in Table 3.

Methods	Pretrained	20NG	R8	R52	Ohsumed	MR	Agnews	Twit nltk	Waimai(zh)
TFIDF + SVM	✗	0.2529	0.7246	0.5932	0.1589	0.5884	0.4241	0.5737	0.7521
TFIDF + LR	✗	0.2633	0.7249	0.6332	0.1798	0.5871	0.5370	0.5791	0.7381
CNN - Rand	✗	0.0768	0.7219	0.6325	0.1889	0.5641	0.3825	0.5822	0.7784
CNN - Pretrained	✓	0.2380	0.7428	0.6896	0.2458	0.6005	0.6636	0.6088	0.7926
LSTM - Rand	✗	0.0545	0.6788	0.4253	0.1319	0.5442	0.3444	0.5458	0.6458
LSTM - Pretrained	✓	0.0593	0.6919	0.5285	0.0948	0.5933	0.5815	0.6098	0.6663
TextGCN	✗	0.1188	0.8628	0.4847	0.1612	0.6222	0.7420	0.7806	0.8065
BERT	✓	0.1347	0.5148	0.6291	0.1464	**0.7666**	0.7261	0.7024	0.8248
TMix	✓	0.2286	0.7322	0.6195	0.1721	0.6267	0.8025	0.6111	0.6376
Our ME-GCN	✗	**0.2861**	**0.8679**	**0.7828**	**0.2740**	0.6811	**0.8043**	**0.8232**	**0.8393**

Table 2: Test accuracy comparison with baselines on benchmark datasets. The bottom row shows the best test accuracy from our proposed model using either max pooling or average pooling. The comparison of our model performance for each dataset using the three pooling methods is provided in Table 3. The second best is underlined.

Pooling Method	20NG	R8	R52	Ohsumed	MR	Agnews	Twit nltk	Waimai(zh)
Max Pooling	0.2775	0.8473	**0.7828**	0.2475	**0.6811**	**0.8043**	**0.8232**	**0.8393**
Avg Pooling	**0.2861**	**0.8679**	0.7675	**0.2740**	0.6658	0.7911	0.8205	0.8303
Min Pooling	0.0424	0.2987	0.2550	0.0294	0.5000	0.2005	0.5000	0.6663

Table 3: Test accuracy of ME-GCN with three different pooling methods, max, average, and min pooling

Learning Methods	20NG	R8	R52	Ohsumed	MR	Agnews	Twit nltk	Waimai(zh)
Separated Learning	**0.2861**	**0.8679**	**0.7828**	**0.2740**	**0.6811**	**0.8043**	**0.8232**	**0.8393**
Shared Learning	0.1582	0.8016	0.6554	0.2635	0.6575	0.6993	0.7037	0.8137

Table 4: Test accuracy of ME-GCN with two multi-stream learning methods, shared and separated learners.

pertrained models in seven datasets, illustrating the potential superiority of self-exploration on the corpus via multi-dimensional edge graph in comparison of pretraining on large external resource.

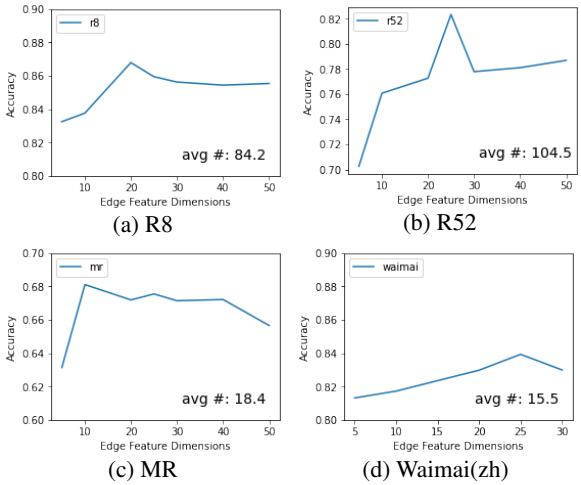
5.2 Learning and Pooling Variant Testing

We compare ME-GCN with three different pooling approaches (max, average, and min pooling) and the result is shown in Table 3. Most datasets produce better results when using max pooling, and the result with max and average pooling outperforms that with min pooling. This is very obvious because the min pooling captures the minimum value of each graph component.

We also compare two multi-stream graph learning methods, including separated and shared stream learning to examine the effectiveness of ME-GCN learning with multi-dimensional edge features. Table 4 presents that the separated stream learners significantly outperforms the shared learners. This shows it is much efficient to learn each dimensional stream with an individual learning unit and initially understand the local structure, instead of learning all global structures at once.

5.3 Impact of Edge Feature Dimension

To evaluate the effect of the dimension size of the edge features, we tested ME-GCN with different dimensions. Figure 1 shows the test accuracy of our ME-GCN model on the four dataset, including R8, R52, MR, Waimai(zh). The bottom right corner for each subgraph includes the average number of the words per document. We noted that the test accuracy is related to the average number of words per document in the corpus. For instance, for 'MR' (avg #: 18.4), test accuracy first increases with the increase of the size of edge feature dimensions, reaching the highest value at 10; it falls when its dimension is higher than 15. However, for R8 and R52 (avg : 84.2 and 104.5), got the highest value at 20 or 25. This is consistent with the intuition that

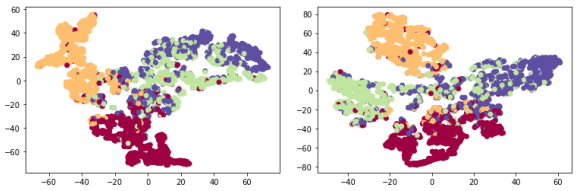


(a) R8 (b) R52 (c) MR (d) Waimai(zh)

Figure 1: Test accuracy by varying edge feature dimensions. The bottom right corner shows the average number of words per document in each corpus.

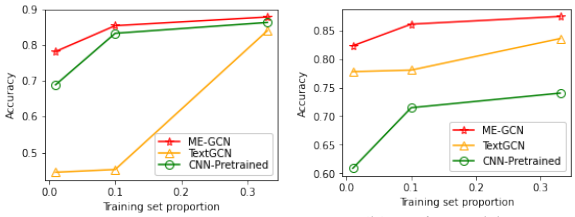
Word Embedding	20NG	R8	R52	Ohsumed	MR	Agnews	Twit nltk	Waimai(zh)
Word2Vec	**0.2861**	**0.8679**	0.7828	0.2740	0.6811	**0.8043**	0.8232	**0.8393**
fastText	0.2510	0.8394	0.7783	0.2550	0.6727	0.7812	0.8333	0.8191
GloVe	0.2526	0.8247	**0.7835**	**0.2832**	**0.6895**	0.7628	**0.8341**	0.8298

Table 5: Test accuracy comparison of our ME-GCN model with different word embedding techniques to train word node embeddings and word-word multi-dimensional edge features.



(a) Dim = 5, second layer (b) Dim = 25, second layer

Figure 2: t-SNE visualisation of test set document embeddings in AgNews (4 classes). The (a) and (b) show second layer document embeddings learned by 5 and 25 dimensional node and edge features respectively.



(a) R52 (b) Twitter nltk

Figure 3: Test accuracy comparison with different number of labelled documents.

the average number of words per document in the corpus should align with the dimension size of the edge features in ME-GCN. The trend is different in waimai dataset as it is Chinese, this is because different languages would have different nature of choosing the efficient edge feature dimension.

Moreover, in order to analyse the impact of the edge feature dimension, we present an illustrative visualisation of the document embeddings learned by ME-GCN. We use the t-SNE tool (Van der Maaten and Hinton, 2008) in order to visualise the learned document embeddings. Figure 2 shows the visualisation of test set document embeddings in AgNews learned by ME-GCN (second layer) 5 and 25 dimensional node and edge features. The AgNews has 4 classes and the average number of words per document is 35.2. Instead of dim=5, having dim=25 as edge features would better to separate them into four classes.

5.4 Impact of Ratio of Labelled Docs

We choose 3 representative methods with the best performance from Table 2: CNN-Pretrained, TextGCN and our ME-GCN, in order to study the impact of the number of labeled documents. Particularly, we vary the ratio of labelled documents and compare their performance on the two datasets, Twitter nltk and R52, that have the smallest number and largest number of classes. Figure 3 reports test accuracies with 1%, 10%, and 33% of the R52 and Twitter nltk training set. We note that our ME-GCN outperforms all other methods consistently. For instance, ME-GCN achieves a test accuracy of 0.8232 on Twitter nltk with only 1% training documents and a test accuracy of 0.8552 on R52 with only 10% training documents which are higher than other models with even the 33% training documents. It demonstrates that our method can more effectively take advantage of the limited labeled data for text classification

5.5 Comparison of Embedding Variants

ME-GCN apply a Word2Vec CBOW in order to train the word node embedding and the related multi-dimensional edge feature. We compare our model with three different word embedding techniques, Word2Vec, fastText, and Glove in Table 5. We noted that using Word2Vec and Glove, word-based models, is comparatively higher than applying the fastText, a character n-gram-based model. This would be affected because the node and edge of ME-GCN are based on words, not characters.

6 Conclusion

In this study, we introduced the ME-GCN (Multi-dimensional Edge-enhanced Graph Convolutional Networks) for semi-supervised text classification, which takes full advantage of both limited labeled and large unlabeled data by rich node and edge information propagation. We propose corpus-trained multi-dimensional edge features in order to efficiently handle the distance/closeness between words and documents as multi-dimensional edge features, and all graph components are based on the given text corpus only. ME-GCN demonstrates promising results by outperforming numerous state-of-the-art methods on eight semi-supervised text classification datasets consistently. In the future, it would be interesting to apply it to other natural language processing tasks.

References

Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'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.

Yixin Cao, Zhiyuan Liu, Chengjiang Li, Juanzi Li, and Tat-Seng Chua. 2019. Multi-channel graph neural network for entity alignment. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1452–1461.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2147–2157.

Xingyuan Chen, Yunqing Xia, Peng Jin, and John Carroll. 2015. Dataless text classification with descriptive lda. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Liyu Gong and Qiang Cheng. 2019. Exploiting edge features for graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9211–9219.

Zhichao Huang, Xutao Li, Yunming Ye, and Michael K. Ng. 2020. Mr-gcn: Multi-relational graph convolutional networks based on generalized tensor product. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 1258–1264. International Joint Conferences on Artificial Intelligence Organization. Main track.

Muhammad Raza Khan and Joshua E Blumenstock. 2019. Multi-gcn: Graph convolutional networks for multi-view networks, with applications to global poverty. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 606–613.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pages 1746–1751.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966. PMLR.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR.

Shen Li, Zhe Zhao, Renfen Hu, Wensi Li, Tao Liu, and Xiaoyong Du. 2018. Analogical reasoning on chinese morphological and semantic relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 138–143.

Hu Linmei, Tianchi Yang, Chuan Shi, Houye Ji, and Xiaoli Li. 2019. Heterogeneous graph attention networks for semi-supervised short 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)*, pages 4823–4832.

Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. 2020. Tensor graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8409–8416.

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2018. Weakly-supervised neural text classification. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 983–992.

Tomas Mikolov, Kai Chen, G. S. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*.

Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. *International Conference on Learning Representations*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1165–1174.

Ming Tu, Guangtao Wang, Jing Huang, Yun Tang, Xiaodong He, and Bowen Zhou. 2019. Multi-hop reading comprehension across multiple documents by reasoning over heterogeneous graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2704–2713.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Shikhar Vashishth, Manik Bhandari, Prateek Yadav, Piyush Rai, Chiranjib Bhattacharyya, and Partha Talukdar. 2019. Incorporating syntactic and semantic information in word embeddings using graph convolutional networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3308–3318.

Nuo Xu, Pinghui Wang, Long Chen, Jing Tao, and Junzhou Zhao. Mr-gnn: Multi-resolution and dual graph neural network for predicting structured entity interactions.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. In *International conference on machine learning*, pages 3881–3890. PMLR.

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.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 649–657.

A Model Architecture

The overall architecture of ME-GCN is presented in Figure 4. After the paper get accepted, the architecture figure will be moved to the main content.

B Hyperparemeter Search

For each dataset we use grid search to find the best set of hyperparameters and select the base model based on the average accuracy by running each model for 5 times. The number of stream: 5,10,20,25,30,40,50. The document edge threshold: 3,5,10,15. The pooling method: max pooling, min pooling, average pooling. The number of hyperparameter search trials is 72($= 6 * 4 * 3$) for each dataset. The best hyperparameters for each dataset and their average accuracy on test set shows in Table 6. And the trend of validation performance is very similar to the testing performance trend.

C Running Details

All the models are trained by using 16 Intel(R) Core(TM) i9-9900X CPU @ 3.50GHz and NVIDIA Titan RTX 24GB. The number of parameters for each part of the model is: Word2vec: $2UT$, Doc2vec: $2T(U+K)$, ME-GCN: $T^2 d_{ms}^{l_1}(1+C)$. And Table 7 shows the number of parameters and training time when using the default hyperparameters.

D Links Related to Datasets and Baseline Models

The links for Datasets:

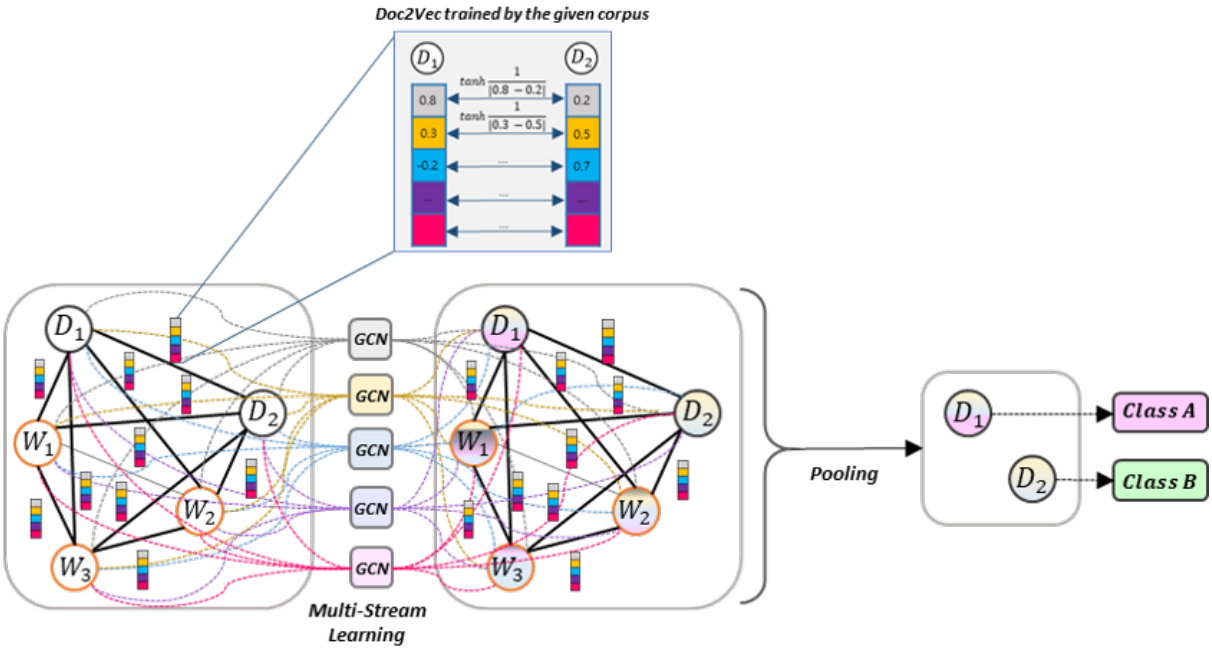


Figure 4: Model Architecture

	20NG	R8	R52	Ohsumed	MR	Agnews	Twit nltk	Waimai(zh)
# Stream	30	20	25	30	10	20	25	30
Document Threshold	15	10	15	5	5	5	3	3
Pooling Method	avg	avg	max	avg	max	avg	max	max
Accuracy	0.2861	0.8679	0.7828	0.2740	0.6811	0.8043	0.8232	0.8393

Table 6: Best hyperparameters for each dataset

		20NG	R8	R52	Ohsumed	MR	Agnews	Twit nltk	Waimai(zh)
Word2vec	# Parameters	304,750	217,650	230,950	432,950	225,050	268,000	31,700	548,950
	Running Time(s)	118	25	76	140	71	83	20	74
Doc2vec	# Parameters	454,750	367,650	380,950	582,950	758,150	568,000	181,700	1,148,300
	Running Time(s)	104	35	118	272	270	140	29	312
ME-GCN	# Parameters	328,125	140,625	828,125	375,000	46,875	78,125	46,875	46,875
	Running Time(s)	198	16	164	286	120	612	14	610
Total	# Parameters	1,087,625	725,925	1,440,025	1,390,900	1,030,075	914,125	260,275	1,744,125
	Running Time(s)	420	76	358	698	461	835	63	996

Table 7: Number of Parameters and Running time for each dataset

- **20NG**: http://qwone.com/~jason/20Newsgroups/
- **R8**, **R52**: https://www.cs.umb.edu/~smimarog/textmining/datasets/
- **MR**: http://www.cs.cornell.edu/people/pabo/movie-review-data/
- **Ohsumed**: http://disi.unitn.it/moschitti/corpora.htm
- **Agnews**: http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles
- **Twitter nltk**: http://nltk.org/howto/twitter.html
- **Waimai**: https://github.com/SophonPlus/ChineseNlpCorpus/

The links for Baseline Models:

- **TextCNN**: https://github.com/DongjunLee/text-cnn-tensorflow
- **TextGCN**: https://github.com/yao8839836/text_gcn
- **BERT BASE**: https://huggingface.co/bert-base-uncased
- **Tmix**: https://github.com/GT-SALT/MixText
- **Chinese BERT**: https://huggingface.co/bert-base-chinese
- **GloVe-pretrained**: https://nlp.stanford.edu/projects/glove/
- **Chinese Word Vectors**: https://github.com/Embedding/Chinese-Word-Vectors

The tokenizer used:

- **English Tokenizer - NLTK**: https://www.nltk.org/api/nltk.tokenize.html
- **Chinese Tokenizer - Jieba**: https://github.com/fxsjy/jieba