A Novel Text Classification Approach based on Meta-path Similarities and Graph Neural Networks

Huan Wang, Jiang Li*, Qing Zhou*, Liang Ge
College of Computer Science, Chongqing University, Chongqing, China
{whuan, lijfrank, tzhou, geliang}@cqu.edu.cn
*Corresponding authors: Jiang Li (lijfrank@cqu.edu.cn) and Qing Zhou (tzhou@cqu.edu.cn)

***Abstract*—With the rise of neural networks, studies on text classification have transitioned from traditional methods to deep learning, especially to graph neural networks on text graphs constructed from corpora. In this paper, we model the complex instances and rich interactions in text classification as a heterogeneous graph. Nevertheless, due to the overlook of indirect relations between documents, graph neural networks have not been fully exploited for the heterogeneous text graph with different types of nodes and links. Consequently, we propose a Meta-Path-based Text Graph Neural Network (MPTGNN) for text classification. Specifically, we first construct a heterogeneous text graph from corpora; we then transform the text graph into several homogeneous weighted graphs via some pre-defined meta-paths; we also propose a Two-stage Multi-graph Information Fusion method (TMIF) for document representation. Empirical results on multiple benchmark datasets have proved that our proposed method outperforms state-of-the-art graph-based methods like Text GCN.**

***Keywords*—heterogeneous graph, text classification, natural language processing, graph neural networks, meta-path.**

I. Introduction

Natural Language Processing (NLP) is a significant research direction in the field of computer science and artificial intelligence, in which text classification is one of crucial and classical tasks. The purpose of text classification is to annotate a given text sequence with one (or multiple) class label(s) describing its textual content [1]. Traditional text classification methods rely heavily on feature engineering and have stringent requirements on the input text data. Recently, neural network models have been exploited for text classification such as Convolutional Neural Networks (CNNs) [2] and Recurrent Neural Networks (RNNs) [3]. In order to increase the representation flexibility of such models, the attention mechanism has been introduced as a component of text classification model. Although these methods are effective, they cannot directly process graph-structured data, which leads to the loss of link information in a corpus.

Graph Neural Networks (GNNs), as deep learning techniques for graph-structured data, have shown superior performance and have attracted widespread attention [4]. For text classification based on GNNs, researchers need to first construct a graph from the text corpus. Zhang et al. [5] improved Defferrard et al.'s [6] work by applying word co-occurrence and document-word relations. However, this method ignores the document-document relationships and fails to capture semantic information in the heterogeneous network. The constructed text graph is usually a heterogeneous graph containing different types of vertices and links. Such text heterogeneous graphs integrate complex objects and rich semantic information, and are not fully considered in general GNNs, e.g., Graph Convolutional Network (GCN) [7] and Simplifying Graph Convolutional Network (SGC) [8] are only suitable for homogeneous graphs.

Taking into account the limitations of existing solutions, we hold the opinion that it is of critical importance to propose a method that can be used to heterogeneous text graph classification. In this work, we propose a novel text classification method based on meta-path similarities and graph neural networks, which is equipped with the following steps to effectively tackle the challenge of heterogeneous text graph classification: 1) we construct a heterogeneous text graph, which integrates rich semantic relations and structural information from the text corpus; 2) we transform the text graph into several homogeneous weighted-graphs based on some pre-defined meta-paths, where the edge weights depend on the document similarities of each meta-path; and 3) we propose a Two-stage Multi-graph Information Fusion method (TMIF) for document representation, which contains node-level and semantic-level aggregation. In the node-level aggregation, the graph convolution network is employed to integrate the neighboring document representations by weight; in the semantic level aggregation, the attention mechanism is adopted to fuse the document representations from different homogeneous graphs by weight.

Our proposed model named MPTGNN can flexibly utilize the rich interactive and semantic information in heterogeneous graph due to the consideration of meta-path. The overall model can be optimized via backpropagation in an end-to-end fashion. Our main contributions in this paper are as follows:

- We propose a Meta-Path-based Text Graph Neural Network (MPTGNN) for text classification, the complex heterogeneous text graph is converted into multiple homogeneous weighted graphs. In addition, the homogeneous graphs based on meta-path contain rich structural information and semantic relations.
- A Two-stage Multi-graph Information Fusion method (TMIF) is proposed for document representation, in which multiple weighted homogeneous graphs are used

DOI reference number: 10.18293/SEKE2021-006

as inputs of GCN to obtain multiple document representations, and an attention mechanism is employed to fuse multiple document representations by weight in semantic-level aggregation.
- Results on several benchmark datasets have demonstrated that our proposed method is of effectiveness and outperforms state-of-the-art graph-based approach for text classification. It promotes the development of text classification method based on graph model.

II. Related Work

A. Text Classification

Traditional text classification studies mainly focus on feature engineering and classification algorithm [9]. For feature engineering, the most commonly used methods are one-hot encoding, TF-IDF and word2vec. Some recent studies [10], [11] convert texts into graphics and extract path-based features for classification. For classification algorithm, the frequently used methods are K-Nearest Neighbor, Naive Bayes, Support Vector Machines and so on. Although these traditional techniques have succeeded in carefully edited and formal texts, they perform worse for general texts.

The research of text classification based on deep learning revolves around word embedding model and deep neural network. Several recent studies [1], [12] have shown that the success of text classification based on deep learning depends largely on the effectiveness of word embedding. Some authors aggregate unsupervised word embeddings as document embeddings [13], while others jointly learn word, document and label embeddings [14]. For deep neural networks, the most representative models are Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). Kim et al. [15] uses a single-layer CNN for sentence-level classification tasks to achieve promising results. Conneau et al. [16] verify the possibility of character-level CNN to complete text classification tasks. Zhang et al. [17] use LSTM, a specific type of RNN, to learn text representation. Nevertheless, these methods rely heavily on the amount of training data and are insufficient to capture complex semantic information due to the overlook of the relations among documents or words.

Yao at al. [5] take inspiration from the recent developments of GNNs to propose a method termed Text GCN for text classification. They turn text classification problem into a node classification problem, which captures high order neighborhoods information. The work of Text GCN mainly includes two parts: 1) they regard words and documents as nodes and construct a large graph from an entire corpus; and 2) they put this graph as input into GCN to train a model. According to Text GCN, its adjacency matrix A is defined as follows:

$$A_{ij} = \begin{cases} PMI(i,j) & i,\ j \text{ are words, PMI}(i,\ j) > 0 \\ TF-IDF_{ij} & i \text{ is document, } j \text{ is word} \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where A_{ij} represents the weight of edge between node i and node j. They employed point-wise mutual information (PMI) to calculate the weights between two word nodes, and treated term frequency-inverse document frequency (TF-IDF) as the weight of the edge between a document node and a word node. They actually constructed a heterogeneous graph, but simply fed it into the GCN as a homogeneous graph. Therefore, Text GCN may cause inaccurate classification accuracy due to missing rich semantic and structural information.

B. Graph Neural Networks

Recently, Graph Neural Networks (GNNs) have achieved success in processing graph-structured data, which has certificated its virtue on modeling behaviors in networks [18]. Li et al. [19] presented a propagation model that incorporates gated recurrent units to propagate information across all nodes. Kipf et al. [7] proposed a spectral approach, called Graph Convolutional Network (GCN), which designs a graph neural network model via a localized first-order approximation of spectral graph convolutions. GCN is a multi-layer neural network, which directly operates on a homogeneous graph and obtains the node's embedding vector by learning its neighborhood information. More formally, consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are sets of nodes and edges, respectively. Let $X \in \mathbb{R}^{n \times m}$ be a matrix containing node features $x_v \in \mathbb{R}^m$. The layer-wise propagation rule for GCN is as follows:

$$H^{(l+1)} = \sigma(\hat{A} H^{(l)} W^{(l)}) \tag{2}$$

where $\hat{A} = \widetilde{D}^{-\frac{1}{2}} \widetilde{A} \widetilde{D}^{-\frac{1}{2}}$ is the symmetric normalized adjacency matrix, $\widetilde{A} = A + I$, $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$, and $W^{(l)}$ is a layer-specific trainable weight matrix, and $\sigma(\cdot)$ denotes an activation function. $H^{(l)} \in \mathbb{R}^{n \times m}$ is the hidden representation matrix for nodes in the l-th layer. Initially, $H^{(0)} = X$.

Wu et al. [8] presented a simple linear model, named Simplifying Graph Convolutional Network (SGC), which repeatedly eliminates the nonlinearity between GCN layers and folds the resulting function into a linear transformation. These graph neural networks can only be applied to homogeneous graphs, and cannot fully deal with heterogeneous graphs containing various types of nodes and links.

C. Meta-path

A meta-path $\mathcal{P}$ is defined as a path in the form of $O_1 \xrightarrow{R_1} O_2 \xrightarrow{R_2} \dots \xrightarrow{R_{l-1}} O_l$ (abbreviated as $O_1 O_2 \dots O_l$), which describes a composite relation $R = R_1 \circ R_2 \circ \dots \circ R_{l-1}$ between the pair of types O_1 and O_l. The commuting matrix is defined by Sun et al. [20] to compute the frequencies of all the paths related to a meta-path. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a commuting matrix $M_{\mathcal{P}}$ for a metapath $\mathcal{P} = (O_1 O_2 \dots O_l)$ is defined as $M_{\mathcal{P}} = \mathcal{A}_{O_1 O_2} \mathcal{A}_{O_2 O_3} \dots \mathcal{A}_{O_{l-1} O_l}$, where $\mathcal{A}_{O_i O_j}$ is the adjacency matrix between types O_i and O_j. $M_{\mathcal{P}}(i,j)$ represents the number of path instances between objects x_i and y_j.

Given a user-specified meta-path $\mathcal{P} = (O_1 O_2 \dots O_l)$, we should calculate the similarity of a pair of objects $x \in O_1$ and

$y \in O_l$. There are several straightforward similarity measures: path count, random walk or pair-wise random walk. These measures, however, are biased towards highly visible objects or highly concentrated objects, so they cannot capture equivalent semantics [20]. According to Sun et al., PathSim between two objects of the same type x and y is suitable and defined as follows:

$$s(x,y) = \frac{2 \times |\{p_{x \rightsquigarrow y} : p_{x \rightsquigarrow y} \in \mathcal{P}\}|}{|\{p_{x \rightsquigarrow x} : p_{x \rightsquigarrow x} \in \mathcal{P}\}| + |\{p_{y \rightsquigarrow y} : p_{y \rightsquigarrow y} \in \mathcal{P}\}|} \tag{3}$$

where $p_{x \rightsquigarrow y}$ is a path instance between x and y.

III. Methodology

A. Overall Framework

In order to make full use of the rich interactive information between objects in the text graph, we propose a Meta-Path-based Text Graph Neural Network (MPTGNN) to learn the structural feature representation of documents. The method proposed in this paper converts a heterogeneous text graph into multiple homogeneous graphs through meta-path, which can flexibly capture the rich structural and semantic information in Heterogeneous Information Network (HIN). The entire framework of our proposed method is shown in Figure 1. The method mainly includes four steps: 1) we construct a heterogeneous text graph from corpora, which contains various types of nodes and links; 2) based on several pre-defined meta-paths, we transform the text graph into several homogeneous weighted-graphs, where the edge weights depend on the document similarities from each meta-path; 3) we propose a Two-stage Multi-graph Information Fusion method (TMIF) for document representation; and 4) we apply a multi-layer perceptron for text classification.

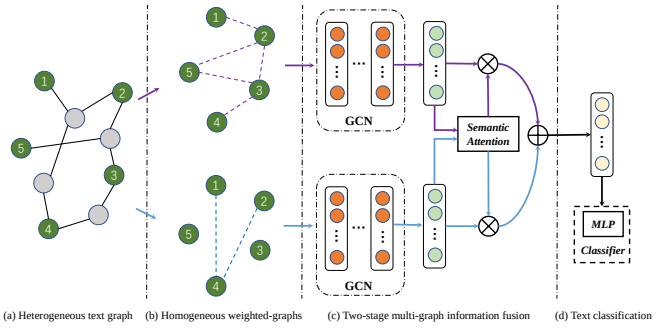

Fig. 1. The framework of this method. (a) Constructing a heterogeneous text graph containing document-nodes (green circular) and word-nodes (gray circular). (b) Transforming heterogeneous text graph into various homogeneous weighted-graphs. (c) Two-stage Multi-graph Information Fusion (TMIF) considers both of node-level and semantic-level aggregation. (d) A multi-layer perceptron for text classification.

B. PathSim-based Homogeneous Weighted-graphs

As proposed method by Yao et al. [5], we build a single heterogeneous text graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{X})$ for the corpus based on word co-occurrence and document word relations, where $\mathcal{V}, \mathcal{E}, \mathcal{A}$ and $\mathcal{X}$ denote node-set, edge-set, adjacency matrix and feature-set, respectively. The set of nodes $\mathcal{V}$ contains all documents $\mathcal{D}$ and unique words $\mathcal{W}$, i.e., $\mathcal{V} = \mathcal{D} \cup \mathcal{W}$. The set of edges $\mathcal{E}$ includes two major types of relations, which are word-word edges and document-word edges. $\mathcal{A}_{ij}$ represents the weight of node i and node j in the heterogeneous text graph.

In our work, we introduce meta-path similarities to transform a heterogeneous text graph into several homogeneous weighted-graphs. In a heterogeneous information graph, two objects can be connected through different paths. For example, in the text graph we established, two documents can be connected through the meta-path of $document \rightarrow word \rightarrow document (\mathcal{DWD})$, or through the meta-path of $document \rightarrow word \rightarrow word \rightarrow document (\mathcal{DWWD})$. Therefore, we can capture the relationship between documents, which is helpful to fully capture the rich semantic information in the original heterogeneous graph. We use meta-path similarities to represent the weights of document-document connections to construct a new adjacency matrix. Since the meta-paths in the text graph we established are all symmetrical, we adopt PathSim to capture the subtlety of peer-to-peer similarities. Given a symmetric meta-path $\mathcal{P}$, PathSim between two objects x_i and x_j from the same type can be calculated as:

$$S_{\mathcal{P}}(i,j) = \frac{2M_{\mathcal{P}}(i,j)}{M_{\mathcal{P}}(i,i) + M_{\mathcal{P}}(j,j)} \tag{4}$$

where $M_{\mathcal{P}}$ is the commuting matrix for the meta-path $\mathcal{P}$, $M_{\mathcal{P}}(i,i)$ and $M_{\mathcal{P}}(j,j)$ are the visibility for x_i and x_j in the network given the meta-path.

The adjacency matrix $\mathcal{A}$ can be divided into $\mathcal{A}_{\mathcal{DW}}$, $\mathcal{A}_{\mathcal{WD}}$ and $\mathcal{A}_{\mathcal{WW}}$, where $\mathcal{A}_{\mathcal{DW}}$ is the adjacency matrix between type $\mathcal{D}$ (documents) and type $\mathcal{W}$ (words). We define a commuting matrix $M_{\mathcal{P}}$ for each meta-path in the text graph:

$$M_{\mathcal{P}=(\mathcal{DWD})} = \mathcal{A}_{\mathcal{DW}}\mathcal{A}_{\mathcal{WD}} \tag{5a}$$

$$M_{\mathcal{P}=(\mathcal{DWWD})} = \mathcal{A}_{\mathcal{DW}}\mathcal{A}_{\mathcal{WW}}\mathcal{A}_{\mathcal{WD}} \tag{5b}$$

where commuting matrix $M_{\mathcal{P}=(\mathcal{DWD})}$ is a weight matrix, each element denotes the sum of the weights of pair-documents with the co-owned words.

For the purpose of the reduction in time and space resources, we utilize top-k similarity search for an object $x_i \in O_1$ is to find sorted k objects in the same type, such that $S_{\mathcal{P}}(x_i, x_j) \geq S_{\mathcal{P}}(x_i, \bar{x}_j)$, for any $\bar{x}_j$ not in the returning list and x_j in the returning list. The top-k similarity search is shown as follows:

$$S_{\mathcal{P}} = rank_i(S_{\mathcal{P}}(x_i, \cdot), k) \tag{6}$$

where $rank(S_{\mathcal{P}}(x_i, \cdot), k)$ is a ranking operation, which keeps k-largest values for the object x_i in $S_{\mathcal{P}}(x_i, \cdot)$ and assigns 0 to the rest. We can convert the heterogeneous text graph to several homogeneous weighted-graphs via meta-path similarity, obtaining $S_{\mathcal{P}=(\mathcal{DWD})}$ and $S_{\mathcal{P}=(\mathcal{DWWD})}$ for all the document nodes.

C. Two-stage Multi-graph Information Fusion

We propose a Two-stage Multi-graph Information Fusion method (TMIF) for document representation, including node-level aggregation and semantic-level aggregation. The node-level aggregation integrates the influence from neighboring

document representations via graph convolutional network, while the semantic-level aggregation integrates the influence from different homogeneous graphs through the attention mechanism.

1) Node-level Aggregation: For node-level aggregation, we perform the weighted integration of neighboring document representations based on weighted graph convolutional network, which can fully extract the interactive information of objects in the text graph. Considering the node-level aggregation in our homogeneous weighted-graphs, a single-layer GCN based on different meta-paths can be described as follows:

$$H_{\mathcal{P}} = \hat{S_{\mathcal{P}}} X W_{\mathcal{P}} \tag{7}$$

where $\hat{S_{\mathcal{P}}} = D^{-\frac{1}{2}} S_{\mathcal{P}} D^{-\frac{1}{2}}$ is the symmetric normalized similarity matrix based on meta path $\mathcal{P}$, and $W_{\mathcal{P}}$ is the node-level trainable weight matrix based on the meta-path $\mathcal{P}$. X is the feature matrix for all the document nodes.

2) Semantic-level Aggregation: Through node-level aggregation, we can get the node embedding of each type of meta-path, which can only denote the document representation of specific semantics. In order to learn node embedding which has more rich semantic information, we use an attention mechanism to automatically learn the importance of different meta-paths and fuse the document representations by weight in the semantic-level aggregation. With the learned weights as coefficients, we can fuse all the semantic-specific node embeddings to obtain the final embedding Z as follows:

$$Z = \sum_{\mathcal{P} \in \Psi} \boldsymbol{\alpha}_{\mathcal{P}} H_{\mathcal{P}} \tag{8}$$

where $\boldsymbol{\alpha}_{\mathcal{P}}$ represents the learned weight vector under the meta-path $\mathcal{P}$ for all the node embeddings, and Ψ is the set of meta-path types. Z is the final representation matrix for all the document nodes.

To learn the importance of node representation in each meta-path, we perform nonlinear transformation and employ a semantic-level attention vector μ. The importance of node representation in each meta-path is calculated as follows:

$$e_{\mathcal{P}} = \mu^T \sigma(W h_{\mathcal{P}} + b) \tag{9}$$

where W is the weight matrix, b is the bias vector, and μ is the semantic-level attention vector. $h_{\mathcal{P}}$, a column from $H_{\mathcal{P}}^T$, is a learned node embedding from the node-level aggregation. For the meaningful comparison, all the above parameters are shared for the semantic-specific node embeddings in all meta-paths.

After obtaining the importance of node representation in each meta-path, we normalize them through a softmax function. The weight of node representation in the meta-path $\mathcal{P}$, denoted as $\alpha_{\mathcal{P}}$, is calculated as follows:

$$\alpha_{\mathcal{P}} = \frac{exp(e_{\mathcal{P}})}{\sum_{\bar{\mathcal{P}} \in \Psi} exp(e_{\bar{\mathcal{P}}})} \tag{10}$$

where Ψ is the set of meta-path types, and $\alpha_{\mathcal{P}}$ can be interpreted as the contribution of the meta-path $\mathcal{P}$ for the learned node embedding.

For document classification tasks, we first feed the final representation matrix to a $log\_softmax$ layer; then we exploit the negative log likelihood loss over training data with the L2-norm. The details are defined as follows:

$$V = log\_softmax(Z^{(l)}) \tag{11a}$$

$$L = -\sum_{i \in D_{train}} V_{im} + \eta \|\Theta\|_2 \tag{11b}$$

where $Z^{(l)}$ denotes the l-th layer document embedding. D_{train} is the set of document node indices for training, V_{im} is the predicted value, Θ denotes other learnable parameters in the model, and η is regularization factor.

IV. Data and Experiment

A. Datasets and Baselines

We conduct extensive experiments on 5 benchmark text datasets including MR, Ohsumed, R8, R52 and AGNews. The statistics for preprocessed datasets are summarized in Table I.

TABLE I
STATISTICS OF TEXT BENCHMARK DATASETS.

Dataset	#Documents	#Train	#Test	#Words	#Classes	Average Length
MR	10,662	7,108	3,554	18,764	2	20.39
Ohsumed	7,400	3,357	4,043	14,157	23	135.82
R8	7,674	5,485	2,189	7,688	8	65.72
R52	9,100	6,532	2,568	8,892	52	69.82
AGNews	6,000	4,000	2,000	9,402	4	7.62

- **MR**: A movie review dataset for binary sentiment classification, in which each movie review contains only one sentence [21]. The corpus has 5,331 positive and 5,331 negative reviews.
- **Ohsumed**: It is a bibliographic database of important medical literature maintained by the National Library of Medicine, which is from the MEDLINE database. We divide training set and test set according to text GCN.
- **R8 and R52**: They are two subsets of the Reuters 21578 dataset, which is a collection of documents that appeared on Reuters newswire in 1987.
- **AGNews**: We randomly selected 6,000 pieces of news from AGNews, evenly distributed into 4 classes. The ratio of training set and test set is 2:1.

In order to evaluate our method comprehensively, we compare it with the following methods. Text GCN and SGC are graph-based techniques.

- **CNN**: CNN [2] with and without pre-trained word embeddings, named CNN-rand and CNN-pretrain, respectively.
- **LSTM**: LSTM [3] with and without pre-trained word embeddings, named LSTM-rand and LSTM-pretrain, respectively.
- **fastText**: FastText [22] is a simple and efficient approach for representation learning, which treats the average of

http://www.cs.cornell.edu/people/pabo/movie-review-data/
http://disi.unitn.it/moschitti/corpora.htm
https://www.cs.umb.edu/∼smimarog/textmining/datasets/
http://www.di.unipi.it/∼gulli/AG_corpus_of_news_articles.html

TABLE II
TEST ACCURACIES (%) OF DIFFERENT METHODS ON BENCHMARK DATASETS. WE RUN EACH MODEL FOR 10 TIMES AND RECORD ITS MEAN ± STANDARD DEVIATION. THE BEST RESULTS ARE IN BOLD, AND THE SECOND-BEST RESULTS ARE UNDERLINED.

Model	MR	Ohsumed	R8	R52	AGNews
CNN-rand	72.11 ± 0.70	51.85 ± 1.72	96.27 ± 0.24	92.00 ± 0.51	57.66 ± 0.98
CNN-pretrain	75.96 ± 0.32	68.60 ± 0.39	97.10 ± 0.09	93.57 ± 0.20	76.72 ± 0.65
LSTM-rand	70.05 ± 0.77	31.68 ± 1.27	93.77 ± 0.51	88.20 ± 0.60	60.71 ± 0.90
LSTM-pretrain	75.00 ± 0.50	45.55 ± 2.01	95.90 ± 0.85	88.62 ± 1.81	72.00 ± 1.19
fastText-pretrain	76.00 ± 1.36	58.16 ± 0.78	96.23 ± 0.31	90.50 ± 0.31	76.85 ± 0.68
fastText-pretrain -bigrams	73.22 ± 0.56	46.60 ± 0.88	94.98 ± 0.28	88.70 ± 0.79	62.39 ± 0.95
Text GCN	76.62 ± 0.13	68.26 ± 0.30	**97.18 ± 0.09**	93.68 ± 0.09	76.52 ± 0.14
SGC	76.91 ± 0.02	69.50 ± 0.02	96.39 ± 0.04	**94.17 ± 0.05**	70.05 ± 0.00
MPTGNN	**77.23 ± 0.33**	**69.63 ± 0.45**	97.13 ± 0.11	93.81 ± 0.29	**77.60 ± 0.30**

word/n-grams embeddings as document representations, then feeds document representations into a linear classifier.

- **Text GCN**: Text GCN [5] builds a graph for the corpus based on word co-occurrence and document word relations, then applies GCN for text classification.
- **SGC**: SGC [8] is a simple linear model by simplifying the graph convolutional network. We apply SGC with the text graph established by Text GCN for text classification.

B. Experimental Settings

In the construction of heterogeneous text graph part, we set the window size as 20. In the meta-path similarities part, we set $k = rate \times \#Documents$ for the top-k similarity search, where the $rate$ is 0.4 for MR and R52, the $rate$ is 0.45 for Ohsumed, and the $rate$ is 0.55 for R8 and AGNews. In the node-level aggregation part, we set the hidden dimension as 64 and the dropout rate as 0.5. In the semantic-level aggregation part, we set the dimension of the semantic-level attention vector μ to 64, and the dropout rate as 0.1. Furthermore, we set the layer number l of MPTGNN as 1, set the regularization factor $\eta = 1e-5$, use $Adam$ algorithm with learning rate 0.03 to train MPTGNN up to 500 epochs, and stop training if the validation loss does not decrease for 10 consecutive epochs. For baseline methods using pre-trained word embeddings, we use 300-dimensional Google's word2vec word embeddings.

C. Experimental Results and Analysis

The node classification accuracies of different methods on benchmark datasets are shown in Table II. We can see that our method termed MPTGNN outperforms all the baselines on MR, Ohsumed and AGNews. Our method achieves the second-best results in the other two datasets i.e., R8 and R52, and its performance is very close to the best model.

Whether using randomly initialized word embeddings or pre-trained word embeddings, CNN performs much better than LSTM on MR, Ohsumed, R8, and R52. FastText-pretrain without bigrams performs well, outperforming CNN and LSTM on MR and AGNews, and even achieves the second-best result for AGNews. Text GCN based on graph neural network is a little better than CNN-pretrain on MR, R8, and R52, getting the best result for R8. The simplified neural network model SGC performs slightly better than Text GCN on MR, Ohsumed, and R52, achieving the best result for R52.

Our method outperforms all the methods on a lot of datasets, which showcases the effectiveness of our proposed method. The reasons why MPTGNN works well include that 1) we propose a method based on meta-path similarity to effectively capture the indirect relationships between documents; 2) the homogeneous weighted-graphs we constructed contains rich interactive and semantic information; 3) the top-k similarity search can select the top-k document nodes with the largest edge weight with the current node, which can reduce the noise; and 4) the attention mechanism can be helpful for weighted fusion of multiple document representations based on different meta-paths.

D. Ablation Study

1) Effect of the meta-path similarities: To test the effectiveness of the meta-path similarities, we compare MPTGNN with its simplified version, i.e, the meta-path similarities are removed and only GNN is applied. As shown in Table III, MPTGNN performs better than GNN, demonstrating the importance of meta-path similarities for our method.

TABLE III
TEST ACCURACIES (%) OF MPTGNN AND ITS SIMPLIFIED VERSION, GNN. WE RUN EACH MODEL FOR 10 TIMES AND REPORT ITS MEAN ± STANDARD DEVIATION.

Model	MR	Ohsumed	R8	R52	AGNews
GNN	76.95 ± 0.61	69.30 ± 0.61	97.01 ± 0.30	93.13 ± 0.73	77.25 ± 0.29
MPTGNN	**77.23 ± 0.33**	**69.63 ± 0.45**	**97.13 ± 0.11**	**93.81 ± 0.29**	**77.60 ± 0.30**

2) Effect of the attention mechanism: In order to test the validity of attention mechanism in the semantic-level aggregation part, we compare our model with some variants. As shown in Table IV, we compare MPTGNN with two variant models. In the semantic-level aggregation part, the concatenation and average instead of attention mechanism in MPTGNN_con and MPTGNN_ave are considered respectively. We can notice that MPTGNN performs better than MPTGNN_con and MPTGNN_ave on most datasets, demonstrating the effective-

https://code.google.com/archive/p/word2vec/

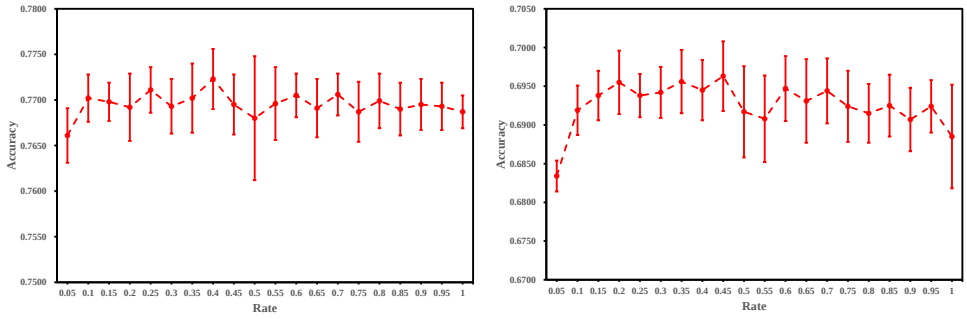

Fig. 2. Test accuracies with different rates. The left subgraph is the result of MR, and the right subgraph is the result of Ohsumed.

ness of attention mechanism in the semantic-level aggregation part.

TABLE IV
TEST ACCURACIES (%) OF MPTGNN AND ITS VARIANTS. WE RUN EACH MODEL FOR 10 TIMES AND REPORT ITS MEAN ± STANDARD DEVIATION.

Model	MR	Ohsumed	R8	R52	AGNews
MPTGNN_con	76.91 ± 0.27	69.28 ± 0.52	96.87 ± 0.16	93.58 ±0.24	77.89 ± 0.30
MPTGNN_ave	76.99 ± 0.28	68.94 ± 0.52	96.78 ± 0.17	93.51 ± 0.15	**78.03 ± 0.33**
MPTGNN	**77.23 ± 0.33**	**69.63 ± 0.45**	**97.13 ± 0.11**	**93.81 ± 0.29**	77.60 ± 0.30

3) Effect of the top-k similarity search: Test accuracies with different rates ($k = rate \times \#Documents$) for the top-$k$ similarity search on MR and Ohsumed are shown in Figure 2. The test accuracy affected by the $rate$ is fluctuating, and the $rate$ that is too close to 0 or 1 does not result in the best test accuracy. The test accuracy of MR reaches the best result when the $rate$ is 0.4. For Ohsumed, the test accuracy reaches the best result when the $rate$ is 0.45.

V. CONCLUSION

In this paper, we propose a Meta-Path-based Text Graph Neural Network (MPTGNN) for text classification. The proposed method can effectively capture the structural and semantic information in heterogeneous text network. The meta-path similarities are regarded as the weights of document-document connections. We also apply a Two-stage Multi-graph Information Fusion method (TMIF) to learn the embedding of each document. The attention mechanism can be productive for weighted fusion of multiple document representations based on different meta-paths. Experiments on public datasets demonstrate that our proposed method can improve the performance of text classifiers. In our future work, we will consider integrating external knowledge into heterogeneous text graph, such as text topics and knowledge graphs.

ACKNOWLEDGMENT

This work was supported in part by National Natural Science Foundation Project of CQ CSTC (No. cstc2020jcyj-msxmX0554).

REFERENCES

[1] 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," *arXiv: Computation and Language*, 2018.

[2] K. Shimura, J. Li, and F. Fukumoto, "Hft-cnn: Learning hierarchical category structure for multi-label short text categorization," pp. 811–816, 2018.

[3] D. Wang, J. Gong, and Y. Song, "W-rnn: News text classification based on a weighted rnn." *arXiv: Information Retrieval*, 2019.

[4] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchezgonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv: Learning*, 2018.

[5] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," vol. 33, no. 01, pp. 7370–7377, 2019.

[6] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," pp. 3844–3852, 2016.

[7] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv: Learning*, 2016.

[8] F. Wu, T. Zhang, A. Souza, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," *arXiv: Learning*, 2019.

[9] F. Sebastiani, "Machine learning in automated text categorisation: a survey," *ACM Computing Surveys*, 1999.

[10] F. Rousseau, E. Kiagias, and M. Vazirgiannis, "Text categorization as a graph classification problem," vol. 1, pp. 1702–1712, 2015.

[11] K. Skianis, F. Rousseau, and M. Vazirgiannis, "Regularizing text categorization with clusters of words," pp. 1827–1837, 2016.

[12] J. Bian, B. Gao, and T. Liu, "Knowledge-powered deep learning for word embedding," pp. 132–148, 2014.

[13] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," *arXiv: Computation and Language*, 2014.

[14] J. Tang, M. Qu, and Q. Mei, "Pte: Predictive text embedding through large-scale heterogeneous text networks," pp. 1165–1174, 2015.

[15] Y. Kim, "Convolutional neural networks for sentence classification," pp. 1746–1751, 2014.

[16] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," vol. 1, pp. 1107–1116, 2017.

[17] Y. Zhang, Q. Liu, and L. Song, "Sentence-state lstm for text representation," vol. 1, pp. 317–327, 2018.

[18] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *arXiv: Learning*, 2018.

[19] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel, "Gated graph sequence neural networks," *arXiv: Learning*, 2016.

[20] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," vol. 4, no. 11, pp. 992–1003, 2011.

[21] B. Pang and L. Lee, "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)*. Ann Arbor, Michigan: Association for Computational Linguistics, Jun. 2005, pp. 115–124.

[22] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," vol. 2, pp. 427–431, 2017.