TextGTL: Graph-based Transductive Learning for Semi-Supervised Text Classification via Structure-Sensitive Interpolation

Chen Li[1,2], **Xutan Peng**[3], **Hao Peng**[1,2], **Jianxin Li**[1,2] and **Lihong Wang**[4]
[1]1Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, China
[2]State Key Laboratory of Software Development Environment, Beihang University, China
[3]Department of Computer Science, The University of Sheffield, UK
[4]National Computer Network Emergency Response Technical Team/Coordination Center of China, China
{lichen, lijx}@act.buaa.edu.cn, x.peng@shef.ac.uk, penghao@buaa.edu.cn, wlh@isc.org.cn

Abstract

Compared with traditional sequential learning models, graph-based neural networks exhibit excellent properties when encoding text, such as the capacity of capturing global and local information simultaneously. Especially in the semi-supervised scenario, propagating information along the edge can effectively alleviate the sparsity of labeled data. In this paper, beyond the existing architecture of heterogeneous word-document graphs, for the first time, we investigate how to construct lightweight non-heterogeneous graphs based on different linguistic information to better serve free text representation learning. Then, a novel semi-supervised framework for text classification that refines graph topology under theoretical guidance and shares information across different text graphs, namely Text-oriented Graph-based Transductive Learning (TextGTL), is proposed. TextGTL also performs attribute space interpolation based on dense substructure in graphs to predict low-entropy labels with high-quality feature nodes for data augmentation. To verify the effectiveness of TextGTL, we conduct extensive experiments on various benchmark datasets, observing significant performance gains over conventional heterogeneous graphs. In addition, we also design ablation studies to dive deep into the validity of components in TextTGL.

1 Introduction

Text classification is a fundamental and long-standing task of Natural Language Processing, which has been shown useful in countless intelligent services, such as sentiment analysis [Wang *et al.*, 2020a] and fake news detection [Wang *et al.*, 2020b]. Different from traditional sequential learning methods [Iacobacci and Navigli, 2019; Kim, 2014; Blei *et al.*, 2003], some recently proposed studies adopt graph learning approaches to model free text, aiming to improve classification performance based on text representation with higher quality. More concretely, all these works utilize different media (e.g., words [Yao *et al.*, 2019; Liu *et al.*, 2020] and topics [Wang *et al.*, 2019; Hu *et al.*, 2019]) as bridges to associate free text, form heterogeneous text graph, and introduce graph neural networks to capture global and local information simultaneously.

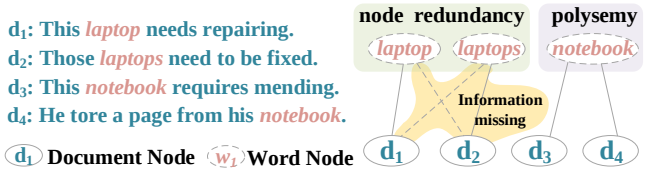

Figure 1: An example heterogeneous graph for propagating information among documents. Some linguistic phenomena will obviously limit the reasonable information propagation among nodes.

Although these graph-based methods have yielded promising results on free text classification, they are mostly intuition-based and lack theoretical guidance, leading to limitations in real-world setups. On the one hand, as previously mentioned, they all rely on constructing heterogeneous graphs which not only hugely expand the scale of model parameters [Yao *et al.*, 2019; Liu *et al.*, 2020], but also introduce subsequent problems such as node redundancy, information missing, and error cascade propagation (illustrated in Figure 1). On the other hand, while the motivation of incorporating graph structures for text modeling is to promote information sharing along edges and break the independence across samples (nodes), thus alleviating the data sparsity and over-fitting issues, existing methods can merely achieve this goal on datasets where edges naturally have meanings. More particularly, this line of algorithms has only been successfully applied on data with explicit graph structures (e.g., citation [Kipf and Welling, 2017; Velickovic *et al.*, 2018] and molecular bond [Fout *et al.*, 2017]) but have never been used to model free text under more strict semi-supervised conditions.

In this work, we propose a semi-supervised pipeline framework, namely *Text-oriented Graph-based Transductive Learning* (TextGTL), which explores attacking the problem of semi-supervised classification for free text with a graph-based architecture. Moreover, to the best of our knowledge, we are the first to construct non-heterogeneous graphs for text classification, which greatly reduces the parameter complexity and makes the trained model lightweight.

Specifically, following the theoretical insights from document similarity research [Pörner and Schütze, 2019], we first design a set of novel strategies to respectively construct text graphs based on multiple linguistic information sources (i.e., semantics, syntax, and sequential context). Next, inspired by the works of [Li *et al.*, 2018; Yang *et al.*, 2019; Li *et al.*, 2020], TextGTL refines the graph topology by strengthening the connection density and information consistency within the same dense substructure of the text graph, so that it can better serve the information propagation. We also obtain high-quality features through attribute space interpolation according to the structural tightness and estimate their low-entropy labels as new training samples, which substantially mitigates the data sparsity issue through data augmentation. Finally, to boost the performance of semi-supervised text classification, we design multi-graph parallel Graph Convolutional Networks [Kipf and Welling, 2017] to share hidden layer signals between multiple text graphs.

In order to demonstrate the effectiveness of our proposed method, we conducted extensive experiments on five popular text classification datasets and benchmarked TextGTL against seven strong baselines. Empirical results show that TextGTL outperforms its counterparts on all tested datasets, especially in the challenging corpus with long sentences. We further performed ablation studies to investigate how each component affects the final model performance as well as the relationship between text graphs via different linguistic information.

The main contributions of this paper can be summarized as follows:

- By proposing novel non-heterogeneous graph construction methods for free text, we significantly reduce the parameter scale and optimize the graph quality.
- Based on our unsupervised text graph topology refinement and data augmentation schemes, we successfully strengthen the useful information propagation among nodes and achieved graph-based semi-supervised free text classification.
- Through joint training on multiple text graphs, the proposed TextGTL system outperforms state-of-the-art methods in extensive experimental setups.

2 Related Work

2.1 Graph-Based Text Classification

Different from the existing sequence learning model, the graph-based methods expect to establish label or feature associations for independent samples by constructing a graph structure. In the initial stage, such methods are more common in text graph topology with real meaning, e.g., citation networks [Kipf and Welling, 2017; Velickovic *et al.*, 2018], protein networks [Fout *et al.*, 2017], and social networks [Liu and Wu, 2018]. They can learn higher-quality text representations through smoothing attributes along the edges. With the development of multiple text graph construction methods, some works began to lean toward free text as the object of analysis. More methods focus on constructing heterogeneous graphs, using words, topics, and other elements as bridges to realize the information propagation [Yao *et al.*, 2019; Liu *et al.*, 2020; Hu *et al.*, 2019; Chen *et al.*, 2020; Ye *et al.*, 2020]. In addition, the work of [Zhang *et al.*, 2020] use graph neural networks to capture the information of the internal structure of the text (e.g., co-occurrence). Some studies also construct text or online word graphs to alleviate the problems of sample imbalance and poor scalability [Tian *et al.*, 2020; Huang *et al.*, 2019].

However, existing methods often use heterogeneous structures to model free text, which greatly increases the parameter scale. Meanwhile, the construction method is mainly based on empirical design, without theoretical guidance.

2.2 Semi-Supervised Text Classification

The prior semi-supervised methods can be roughly divided into two categories: latent variable model and embedding-based model. The former expands the topic model based on labeled samples and then relies on the posterior topic-label distribution for label prediction. The latter uses the given labeled samples as a seed to derive the embedding and labeling of the pending text. Among them, some semi-supervised methods introduce graphs as the basis for derivation. For example, PTE [Tang *et al.*, 2015] first represented the labeled information and different levels of word co-occurrence information as a large-scale heterogeneous text network, which is then embedded into a low dimensional space through an efficient algorithm. HAN [Wang *et al.*, 2019] and HGAT [Hu *et al.*, 2019] designed heterogeneous graph to achieve more flexible feature capture. EGNN-Proto [Lyu *et al.*, 2020] utilize an edge-labeling graph neural network to implicitly models the intra-cluster similarity and the inter-cluster dissimilarity of the documents

However, graph-based semi-supervised methods generally do not consider the derivation strategy [Gururangan *et al.*, 2019; Meng *et al.*, 2018; Miyato *et al.*, 2019] in classic semi-supervised methods. Hence, we propose a novel graph-based data augmentation technique and corresponding graph neural network structure to fill this gap.

3 Methodology

In this section, we will introduce the proposed pipeline framework TextGTL, as shown in Figure 2.

3.1 Problem Definitions & Notions

Semi-supervised text classification is a common natural language processing task. Given a text corpus $D = \{d_1, d_2, \cdots, d_n\}$, where $n = |D|$, and d_i denotes a document. Among D, every text d_i has a corresponding label $y_j \in C$, where $C = \{c_1, c_2, \cdots, c_{n_c}\}$ is the label set, and $n_c = |C|$. $X \in \mathbb{R}^{n \times dem}$ is the attribute set of D, $x_i \in X$ denotes the initial attributes of d_i, and dem is the deminsion of attributes. $Y \in \mathbb{R}^n$ is the label set of D, and due to the semi-supervised setting, $|Y_{obs}| \ll |Y - Y_{obs}|$, where Y_{obs} is the training set.

There exist settings for this task, namely transductive learning and inductive learning, which are different in information visibility. While the former can fully observe and utilize X in both learning and inference stages, the latter is blocked from

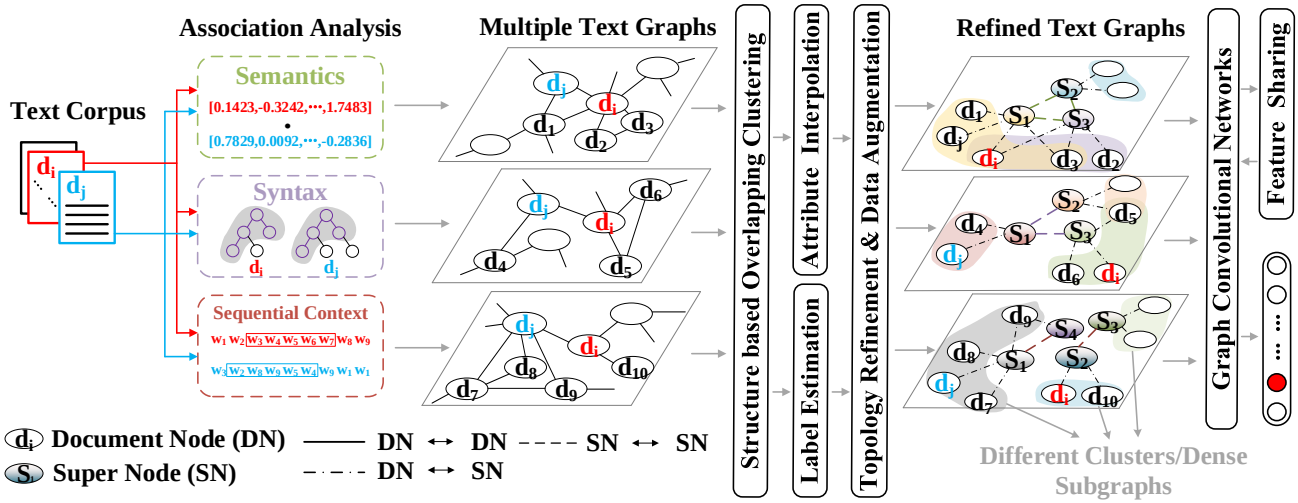

Figure 2: The overall architecture of TextGTL. From left to right, TextGTL first constructs multiple text graphs based on different linguistic factors (see section 3.2). Next, TextGTL is guided by **criterion** $\mathcal{C}$ to refine the graph topology and perform data augmentation (see section 3.3 and Figure 3). Finally, TextGTL feeds the refined multiple text graphs into a 2-layer GCN that can share hidden layer information to achieve semi-supervised text classification (see section 3.4).

partial information during the learning stage but are fed with the complete dataset during testing. In this paper, we focus on transductive learning.

3.2 Non-Heterogeneous Text Graph Construction

In this section, we will explore how to construct meaningful text graphs and propose novel strategies to associate text. Unlike heterogeneous graphs that use words, topics, etc., as bridges to associate documents, we intend to use different linguistic information to consider whether the information is propagated between documents. To capture various information, TextGTL evaluates the propagation probability from three different linguistic perspectives as follows.

Semantics Text Graph

Inspired by the task of unsupervised semantic textual similarity, we design meta-document embedding to integrate multiple pre-trained document vectors based on various methods to represent the documents, and then determine if the association exists [Pörner and Schütze, 2019]. Given zero-mean random vectors v_1, and v_2, Canonical Correlation Analysis (CCA) finds linear projections θ_1 and θ_2 such that $\theta_1^\top v_1$ and $\theta_2^\top v_2$ are maximally correlated. [Bach and Jordan, 2002] showed that CCA reduces to a generalized eigenvalue problem, and Generalized CCA (GCCA) generalizes CCA to K random vectors $v_1, v_2, \cdots, v_K \in \mathbb{R}^{dem_k}$. Then, the generalized eigenvalue problem finds scalar-vector pairs (ρ, θ) that satisfy

$$\begin{vmatrix} \mathbf{0} & \Sigma_{\cdots} & \Sigma_{1,K} \\ \Sigma_{\cdots} & \mathbf{0} & \Sigma_{\cdots} \\ \Sigma_{K,1} & \Sigma_{\cdots} & \mathbf{0} \end{vmatrix} \theta = \rho \begin{vmatrix} \Sigma_{1,1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \Sigma_{\cdots} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Sigma_{K,K} \end{vmatrix} \theta, \quad (1)$$

where $\Sigma_{K,K}$ is the covariance matrix and $\Sigma_{K,1}$ is the cross-covariance matrices. For stability, we add $\tau\sigma_k \mathrm{I}_k, k \in [1, K]$ to every $\Sigma_{k,k}$, where $\tau = 0.1$ is a hyperparameter, I_k is the identity matrix and σ_k is the average variance of v_k. We stack the eigenvectors of the top-dem eigenvalues into operator matrices $\Theta_k \in \mathbb{R}^{dem \times dem_k}$ and define the GCCA meta-embedding of document d_m as (i.e., $v_k = \begin{bmatrix} h_{d_{0,k}} \\ \cdots \\ h_{d_{n,k}} \end{bmatrix}$):

$$h_{d_m}^{GCCA} = \sum_{k=1}^{K} \Theta_k (h_{d_{m,k}} - \mathbb{E}_{i \in |D|}[h_{d_{i,k}}]), \quad (2)$$

where $h_{d_{m,k}} \in \mathbb{R}^{dem_k}$ is the kth kind of pre-training initial document representaiton of d_m. For each document pair (d_i, d_j) to be determined for establishing an association or not, we use a simple classifier as follows:

$$y_{(d_i,d_j)} = \mathrm{ReLU}(\alpha[h_{d_i}^{GCCA} || h_{d_j}^{GCCA}]), \quad (3)$$

where $y_{(d_i,d_j)}$ denotes adding an edge between (d_i, d_j) or not, $\mathrm{ReLU}(\cdot)$ is an activation function, α is a trainable parameter, and $||$ denotes concatenation. The positive and negative examples used in training are derived from the classification training set sample combination. Then, we can minimize the cross-entropy loss for data distribution for training

$$\mathcal{L} = -\sum_{k=1}^{n} (y_k \log(\widehat{y_k}) + (1 - y_k)\log(1 - \widehat{y_k})). \quad (4)$$

When the edge between (d_i, d_j) is determined to be added, we will use $w_{i,j}^{sem} = \sigma(h_{d_i}^{GCCA} \cdot h_{d_j}^{GCCA\top})$ as its initial weight and normalize it on whole graph.

Syntax Text Graph

Dependency parsing is considered to be useful for guiding the understanding of sentences in a document, so we are ready to calculate the contact ratio of syntax dependency between

documents. First, for the document d_i, we use the parser of Stanford CoreNLP to get the smallest dependency unit (i.e., [word, dependency-relation, word]) that exists in the document and generate a set. Next, like the bag-of-words, we represent all documents as a vector $h_{d_i}^{syn} = [0, \cdots, n_j, \cdots, 0]$, where n_j is the number of jth dependency unit in d_i. Similar to the construction method of the semantics text graph, we also build a classifier to add edges and determine the weight of edges $w_{i,j}^{syn}$, so we omit it.

Context Text Graph

The sequential context describes the language characteristics of local co-occurrence between words, which has been widely used in text representation learning. In TextGTL, to evaluate the sequential context similarity between two documents, we design a sliding-window-based calculation method as

$$w_{i,j}^{seq} = \sum_{m,n \in d_i \bigcap d_j} \log(p(w_m, w_n)/p(w_m)p(w_n)), \quad (5)$$

where w_m and w_n are a pair of words which simultaneously appear in both d_i and d_j, and $p(w_m)$ is the probability of the word w_m appeared in a sliding window. $p(w_m, w_n)$ is the probability of the word pair (w_m, w_n) co-occurring in the same sliding window, which can be estimated as the fraction of the total number of sliding windows over d_i and d_j and the number of times that w_m and w_n co-occur in the same sliding window.

In the end, as shown in the Figure 2, we obatin three independent text graphs constructed based on different linguistic information.

3.3 Topology Refinement & Data Augmentation

Based on the widely-adopted perspective of existing graph-based semi-supervised models, i.e., attributes exhibit smoothness along the graph edges [Stretcu *et al.*, 2019; Xu *et al.*, 2020; Li *et al.*, 2018; Li *et al.*, 2020], some works propose a criterion to assess both training samples and given graph topology, which is highly correlated to the subsequent classification performance. **Criterion** $\mathcal{C}$: *The higher the consistency of the node information (i.e., attribute and label) within the same dense subgraph, the more reasonable and sufficient the information propagation.* This criterion is highly consistent with the basic assumptions (i.e., graph homogeneity) of traditional label propagation algorithms, and has been proved by mass experiments, so we use it as our guide to refine text graph topology.

Subgraph Density

Specifically, when we refine the three text graphs' topology respectively, we first need to reasonably strengthen the density of the subgraphs in each text graph (All operation in each of the three text graphs is independent, and we take one as an example.). As shown in Figure 3, through the structure-based overlapping clustering algorithm [Epasto *et al.*, 2017], we obtain the dense subgraphs in the text graph, which are regarded as the range that the information propagation should cover. However, due to the existence of long-distance dependence, information often cannot be propagated adequately in GNNs that only stack about two or three layers. This is the reason why some nodes' representations are under-smoothing.

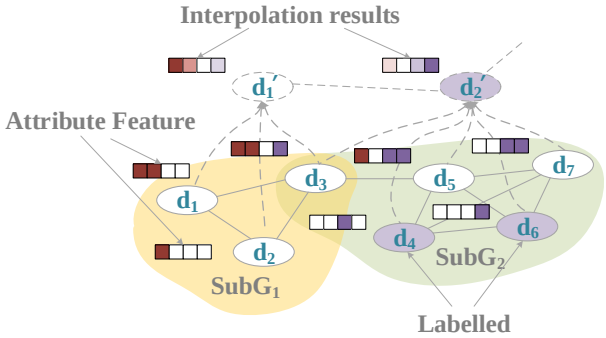

Figure 3: This is a toy example of TextGTL's topology refinement and data augmentation. TextGTL generates super nodes d_1' and d_2' based on the dense subgraphs $SubG_1$, and $SubG_2$ discovered by the structure-based overlapping clustering. It refines the graph topology by associating the super nodes with the original nodes in the cluster (dotted line). Since a super node can be regarded as the central node in a cluster, its attributes (d_1', and d_2') and label (only d_2') can be obtained through attribute interpolation and label estimation, which realizes data augmentation.

Therefore, by generating super nodes and connecting them with corresponding subgraph members, we limit the maximum distance of nodes in the subgraph to 2, which will effectively alleviate the phenomenon of under-smoothing. In particular, Dropedge [Rong *et al.*, 2020] shows that too dense the graph structure is also not conducive to deep node representation learning. This is because in a graph with too high connectivity, information propagation becomes easy, which can easily lead to the convergence of nodes in the entire graph, i.e., the problem of over-smoothing. Hence, we randomly remove some edges in the text graphs. The weights of new edges are also calculated from section 3.2.

Information Consistency (Data Augmentation)

From another perspective, improving the consistency of node information within the subgraph can also improve the performance of the downstream tasks. As the sum of all nodes of the corresponding cluster, the super node can be regarded as multiple sampling of the same label distribution. This operation is similar to *mixup* [Zhang *et al.*, 2018], but different from augmenting the marginal sample to improve the model in a supervised setting, our goal is to obtain samples with higher class separability in a semi-supervised setting. Specifically, according to the *Law of Large Numbers* [Prékopa, 1972], it should fit the original distribution more closely than the original node [Li *et al.*, 2020]. On this basis, as shown in Figure 3, some super nodes can be directly labeled based on their containing training samples in clusters. By introducing a simple classifier trained by such high-quality labeled super nodes, we can obtain more super nodes with high-quality features and low-entropy labels, and provide more training data for downstream models, which realizes data augmentation.

3.4 Joint Training on Multiple Text Graphs

After obtaining the three updated text graphs and the augmented training sample set, we focus on exploring an efficient

graph neural network framework to learn document representation and achieve classification. We consider that, on the one hand, crudely compressing three different text graphs together will inevitably confuse the various structure information and lose the freedom of the model. On the other hand, the basic principle of graph neural network learning is that nodes propagated information to each other and update their feature embeddings by propagating information among other neighboring nodes. Therefore, we attempt to transmit information within the text graph independently while sharing information between different text graphs to promote a more flexible propagation of information throughout the model.

We stack a 2-layer GCN. Among them, the first layer is a general GCN [Kipf and Welling, 2017], and three graphs are fed independently. The first layer of each text graph is the same and independent, as follows:

$$H^{(1)} = \sigma(\tilde{A} \cdot H^{(0)} \cdot W^{(1)}), \tag{6}$$

where $H^{(1)}$ denotes the hidden representation matrix of D at the 1st layer, $H^{(0)}$ is the initial embedding of D, $W^{(1)}$ is a learnable linear transformation matrix, and $\sigma(\cdot)$ is an element-wise nonlinear activation function. Among them, $\tilde{A} = M^{-\frac{1}{2}}(A + I)M^{-\frac{1}{2}}$ represents the symmetric normalized adjacency matrix, A, and M are the weighted adjacency matrix and degree matrix. Next, by combining hidden representations from three text graphs, the second layer as follows:

$$H^{(2)} = \sigma(\tilde{A} \cdot [H^{(1)}_{sem}||H^{(1)}_{syn}||H^{(1)}_{seq}] \cdot W^{(2)}), \tag{7}$$

In the last layer of TextGTL, we perform a mean pooling over graphs to obtain the final representation of documents for classification.

4 Experiment

4.1 Datasets & Baselines

We choose a suite of recently widely used benchmark datasets for experiments and analysis. We conduct extensive experiments on 5 benchmark text datasets: 20-Newsgroups dataset, Ohsumed dataset, R52 Reuters dataset, and R8 Reuters dataset, and Movie Review dataset. These datasets involve many life genres, such as movie reviews, medical literature, and news documents, etc. The Movie Review dataset is designed for binary sentiment classification. The 20-Newsgroups dataset, R52 Reuters dataset, R8 Reuters dataset are news classification datasets. The Ohsumed dataset is medical literature. To evaluate the performance of the semi-supervised learning model, we gave 20 labeled data per class. We split the same number of samples of the verification set as the training set from the dataset, and ensure that there is no sample imbalance in the testing set. A summary statistics of the benchmark datasets is presented in Table 1.

To comprehensively evaluate our proposed method for semi-supervised text classification, we compare it with the following three groups of state-of-the-art methods: **SVM** [Blei *et al.*, 2003]: SVM classifiers using TF-IDF features. **CNN** [Kim, 2014]: CNN classifiers using averaged word embeddings as sentence embedding, whose word embeddings are pre-trained with Wikipedia Corpus based on GloVe. **LSTM** [Iacobacci and Navigli, 2019]: A LSTM sentence encoder using pre-trained word embeddings as CNN. **PTE** [Tang *et al.*, 2015]: A graph-based semi-supervised representation learning method for text data. PTE firstly learns word embedding based on the heterogeneous text networks containing three bipartite networks of words, documents, and labels, then averages word embeddings as document embeddings for text classification. **TextGCN, TensorGCN** [Yao *et al.*, 2019; Liu *et al.*, 2020]: A GCN-based methods on sequential contextual graph and multiple text graph. **HAN** [Wang *et al.*, 2019]: It embeds heterogeneous information network by first converting a heterogeneous information network to several homogeneous subnetworks through pre-defined meta-paths and then applying graph attention networks. To be fair, all of our models use a uniform pre-training vector as the initial representation of the document. The results of models with $*$ come from original code or replication.

	#Nodes	#Classes	#Avg. Length
20NG	18,846	20	221.26
Ohsumed	7,400	23	135.82
R52	9,100	52	69.82
R8	7,674	8	65.72
MR	10,662	2	20.39

Table 1: Summary of the text datasets.

4.2 Experiment Settings

We use $K = 4$ kinds of document initial embeddings in the semantics text graph that are derived from [Pörner and Schütze, 2019], set the length of sliding-window as 20, and the dependency parser used in the syntax text graph is Stanford CoreNLP. We equip TextGTL with the Ego-Splitting [Epasto *et al.*, 2017] as an overlapping clustering algorithm (resolution=1.0). During data augmentation, we select GBDT as a simple classifier, and iterated 10 rounds to obtain more labeled super nodes to add to the training set. As mentioned in section 3.3, in this study, we adopt a two-layer GCN to achieve. The initial features of the first-layer nodes in the three different text graphs are all TF-IDF attributes from TfIdfcounter in sklearn, the output feature dimension is 200, the input feature dimension of the second-layer nodes is 200×3, and the output feature is the number of classes. During the training process, the dropout rate is 0.5, and the L2 loss weight is 5e-6. We use Adam optimizer with a maximum of 200 epochs and a learning rate of 0.002, when the verification loss is not reduced for 10 consecutive epochs, an early stop will be executed. All results reported in this study are the average of 10 independent tests.

4.3 Results Analysis

A comprehensive experiment is conducted on the benchmark datasets. The results presented in Table 2 show that our proposed TextGTL significantly outperforms all baselines (including some state-of-the-art sequential learning and graph-based models). Among them, we notice that the sequential learning method is generally lower than the graph-based method in corpus sets with a larger average text length, which we suspect may be due to the limitation of the representa-

Datasets	SVM	CNN	LSTM	PTE*	TextGCN*	TensorGCN*	HAN*	TextGTL(Ours)
20NG	65.94±0.29	77.84±0.23	76.84±0.19	68.54±0.33	77.63±0.21	78.63±0.31	71.36±0.19	**80.14±0.26**
Ohsumed	40.04±0.27	35.64±0.36	31.55±0.32	43.84±0.41	50.88±0.27	52.17±0.39	41.76±0.35	**54.11±0.24**
R52	79.74±0.31	81.84±0.16	81.79±0.25	74.62±0.22	81.38±0.09	82.29±0.20	73.26±0.28	**83.23±0.25**
R8	80.72±0.17	84.79±0.18	86.46±0.30	80.73±0.26	85.57±0.12	86.15±0.18	78.53±0.15	**87.06±0.30**
MR	57.64±0.18	59.19±0.20	62.37±0.36	56.89±0.36	60.47±0.18	61.25±0.37	58.84±0.26	**62.42±0.28**

Table 2: Test accuracy (%) of the semi-supervised text classification benchmark. The highest performance per dataset is highlighted in **bold**. The ± error bar denotes the standard deviation in 10 independent trials.

text graph	20NG	Ohsumed	R52	R8	MR
Semantics	78.17	51.56	81.05	84.88	60.11
Syntax	77.29	49.74	79.37	83.09	58.93
Sequential	78.85	52.78	81.44	86.01	60.76
Semantics(w/o)	79.15	53.20	81.94	86.33	61.42
Syntax(w/o)	79.76	53.52	82.24	86.84	62.04
Sequential(w/o)	78.44	52.05	81.33	85.12	60.53

Table 3: Analysis of the effectiveness of a single text graph. In the upper half table, the first three rows are the results of only using a single graph, while the lower half table is the results of the complementary experiment, where (w/o) means TextGTL without using corresponding text graph.

Component	20NG	Ohsumed	R52	R8	MR
Only Text Graphs	75.13	48.43	77.97	82.84	57.13
+Topology Refinement	75.84	49.18	78.61	83.42	57.86
+Data Augmentation	79.52	52.84	82.44	86.77	61.63
+Joint Training	80.14	54.11	83.23	87.06	62.42

Table 4: Ablation analysis of TextGTL. + indicates that the corresponding components are added to the model from top to bottom in the table.

tional ability of the sequential model when processing multiple sentences.

4.4 Discussion of Text Graphs

We discuss the specific performance of the text graph proposed in this paper by testing the effectiveness of the three text graphs. The results are shown in Table 3. We notice that the performance of any single text graph is lower than the result of joint training as shown in Table 2, which shows that the three text graphs are complementary in information propagation. In addition, after statistics, the edges and nodes in the text graphs of TextGTL are far less than the existing heterogeneous graph methods.

4.5 Ablation Analysis

Finally, we performed an ablation analysis of the components in our proposed TextGTL to confirm the improvement of model performance brought by different components. The experimental results are shown in Table 4. We notice that the performance of the model based on only three original text graphs is only competitive, which is slightly lower than TextGCN. We guess the main reason is the undersmoothing of information propagation and noise in the text graphs. However, each of the remaining components has brought certain improvements to the performance of the model, especially topology refinement and data augmentation. It proves that **Criterion** $\mathcal{C}$ correctly guides to topology refinement.

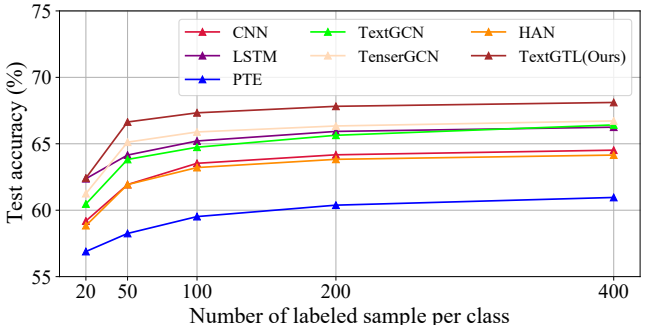

Figure 4: The accuracy with different number of labeled documents.

4.6 Impact of Number of Labeled Data

To explore the performance of different models given different amounts of labeled samples, we experiment with TextGTL and baseline methods on the MR dataset. The experimental results are shown in Figure 4. With the increase of labeled documents, all methods have achieved better results in terms of test accuracy. And under different settings, TextGTL always outperforms all other methods. If fewer labeled documents are provided, the performance of the baseline will be significantly reduced, and our model can still achieve higher performance. It illustrates that our proposed TextGTL can more effectively use limited label data for text classification.

5 Conclusion

In this work, we propose a pipeline framework *Text-oriented Graph-based Transductive Learning* to build text graphs, refine topologies, data augmentation, and design GCNs that share hidden layer information for semi-supervised text classification. Experimental results show that TextGTL can effectively improve the performance of semi-supervised text classification. In addition, we further discussed the performance improvement brought by different strategies in ablation analysis. In the future, we will continue to explore more development possibilities for graph-based text classification tasks.

Acknowledgements

The corresponding author is Jianxin Li. This work was supported by the NSFC program (No. U20B2053, 62002007 and 61772151), S&T Program of Hebei through grant 20310101D, and SKLSDE-2020ZX-12.

References

[Bach and Jordan, 2002] Francis R. Bach and Michael I. Jordan. Kernel independent component analysis. *J. Mach. Learn. Res.*, 3:1–48, 2002.

[Blei *et al.*, 2003] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[Chen *et al.*, 2020] Boyan Chen, Guangquan Lu, Bo Peng, and Wenzhen Zhang. DGRL: text classification with deep graph residual learning. In *ADMA*, volume 12447 of *Lecture Notes in Computer Science*, pages 83–97, 2020.

[Epasto *et al.*, 2017] Alessandro Epasto, Silvio Lattanzi, and Renato Paes Leme. Ego-splitting framework: from non-overlapping to overlapping clusters. In *KDD*, pages 145–154, 2017.

[Fout *et al.*, 2017] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In *NIPS*, pages 6530–6539, 2017.

[Gururangan *et al.*, 2019] Suchin Gururangan, Tam Dang, Dallas Card, and Noah A. Smith. Variational pretraining for semi-supervised text classification. In *ACL*, pages 5880–5894, 2019.

[Hu *et al.*, 2019] Linmei Hu, Tianchi Yang, Chuan Shi, Houye Ji, and Xiaoli Li. Heterogeneous graph attention networks for semi-supervised short text classification. In *EMNLP-IJCNLP*, pages 4820–4829, 2019.

[Huang *et al.*, 2019] Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. Text level graph neural network for text classification. In *EMNLP-IJCNLP*, pages 3442–3448, 2019.

[Iacobacci and Navigli, 2019] Ignacio Iacobacci and Roberto Navigli. Lstmembed: Learning word and sense representations from a large semantically annotated corpus with long short-term memories. In *ACL*, pages 1685–1695, 2019.

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, 2014.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Li *et al.*, 2018] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, pages 3538–3545, 2018.

[Li *et al.*, 2020] Chen Li, Xutan Peng, Hao Peng, Jianxin Li, Lihong Wang, and Philip S. Yu. Forming an electoral college for a graph: a heuristic semi-supervised learning framework. *CoRR*, abs/2006.06469, 2020.

[Liu and Wu, 2018] Yang Liu and Yi-fang Brook Wu. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In *AAAI*, pages 354–361, 2018.

[Liu *et al.*, 2020] Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. Tensor graph convolutional networks for text classification. In *AAAI*, pages 8409–8416, 2020.

[Lyu *et al.*, 2020] Chen Lyu, Weijie Liu, and Ping Wang. Few-shot text classification with edge-labeling graph neural network-based prototypical network. In *COLING*, pages 5547–5552, 2020.

[Meng *et al.*, 2018] Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. Weakly-supervised neural text classification. In *CIKM*, pages 983–992, 2018.

[Miyato *et al.*, 2019] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(8):1979–1993, 2019.

[Pörner and Schütze, 2019] Nina Pörner and Hinrich Schütze. Multi-view domain adapted sentence embeddings for low-resource unsupervised duplicate question detection. In *EMNLP-IJCNLP*, pages 1630–1641, 2019.

[Prékopa, 1972] András Prékopa. Laws of large numbers for random linear programs. *Math. Syst. Theory*, 6(3):277–288, 1972.

[Rong *et al.*, 2020] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *ICLR*, 2020.

[Stretcu *et al.*, 2019] Otilia Stretcu, Krishnamurthy Viswanathan, Dana Movshovitz-Attias, Emmanouil A. Platanios, Sujith Ravi, and Andrew Tomkins. Graph agreement models for semi-supervised learning. In *NeurIPS*, pages 8710–8720, 2019.

[Tang *et al.*, 2015] Jian Tang, Meng Qu, and Qiaozhu Mei. PTE: predictive text embedding through large-scale heterogeneous text networks. In *KDD*, pages 1165–1174, 2015.

[Tian *et al.*, 2020] Jiachen Tian, Shizhan Chen, Xiaowang Zhang, and Zhiyong Feng. A graph-based measurement for text imbalance classification. In *ECAI*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2188–2195, 2020.

[Velickovic *et al.*, 2018] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[Wang *et al.*, 2019] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. Heterogeneous graph attention network. In *WWW*, pages 2022–2032, 2019.

[Wang *et al.*, 2020a] Hao Wang, Shuai Wang, Sahisnu Mazumder, Bing Liu, Yan Yang, and Tianrui Li. Bayes-enhanced lifelong attention networks for sentiment classification. In *COLING*, pages 580–591, 2020.

[Wang *et al.*, 2020b] Yaqing Wang, Weifeng Yang, Fenglong Ma, Jin Xu, Bin Zhong, Qiang Deng, and Jing Gao. Weak supervision for fake news detection via reinforcement learning. In *AAAI*, pages 516–523, 2020.

[Xu *et al.*, 2020] Chunyan Xu, Zhen Cui, Xiaobin Hong, Tong Zhang, Jian Yang, and Wei Liu. Graph inference learning for semi-supervised classification. In *ICLR*, 2020.

[Yang *et al.*, 2019] Liang Yang, Zesheng Kang, Xiaochun Cao, Di Jin, Bo Yang, and Yuanfang Guo. Topology optimization based graph convolutional network. In *IJCAI*, 2019.

[Yao *et al.*, 2019] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *AAAI*, pages 7370–7377, 2019.

[Ye *et al.*, 2020] Zhihao Ye, Gongyao Jiang, Ye Liu, Zhiyong Li, and Jin Yuan. Document and word representations generated by graph convolutional network and BERT for short text classification. In *ECAI*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2275–2281, 2020.

[Zhang *et al.*, 2018] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*. OpenReview.net, 2018.

[Zhang *et al.*, 2020] Yufeng Zhang, Xueli Yu, Zeyu Cui, Shu Wu, Zhongzhen Wen, and Liang Wang. Every document owns its structure: Inductive text classification via graph neural networks. In *ACL*, pages 334–339, 2020.