Sparse Structure Learning via Graph Neural Networks for Inductive Document Classification

Yinhua Piao, [1] **Sangseon Lee,** [2] **Dohoon Lee,** [3] **Sun Kim** [1,3,4,5]

[1] Department of Computer Science and Engineering, Seoul National University
[2] Institute of Computer Technology, Seoul National University
[3] Bioinformatics Institute, Seoul National University, [4] AIGENDRUG Co., Ltd.
[5] Interdisciplinary Program in Artificial Intelligence, Seoul National University

Abstract

Recently, graph neural networks (GNNs) have been widely used for document classification. However, most existing methods are based on static word co-occurrence graphs without sentence-level information, which poses three challenges:(1) word ambiguity, (2) word synonymity, and (3) dynamic contextual dependency. To address these challenges, we propose a novel GNN-based sparse structure learning model for inductive document classification. Specifically, a document-level graph is initially generated by a disjoint union of sentence-level word co-occurrence graphs. Our model collects a set of trainable edges connecting disjoint words between sentences, and employs structure learning to sparsely select edges with dynamic contextual dependencies. Graphs with sparse structures can jointly exploit local and global contextual information in documents through GNNs. For inductive learning, the refined document graph is further fed into a general readout function for graph-level classification and optimization in an end-to-end manner. Extensive experiments on several real-world datasets demonstrate that the proposed model outperforms most state-of-the-art results, and reveal the necessity to learn sparse structures for each document.

Introduction

Document classification, a task of using algorithms to automatically classify the input document to one or multiple categories, is one of the most fundamental tasks in the field of Natural Language Processing (NLP). The essential part of document classification is to extract features that can represent the documents. Conventional approaches use handcrafted features, e.g., bag-of-words, term frequency-inverse document frequency. With the advent of deep learning technologies, related works such as Word2Vec (Mikolov et al. 2013), utilize contextual information to learn word representations. Considering the word order in a sequence, many models adopt sequence-based models including recurrent neural networks (RNNs) (Mikolov et al. 2010; Tai, Socher, and Manning 2015; Liu, Qiu, and Huang 2016), and convolutional neural networks (CNNs) (Kim 2014; Zhang, Zhao, and LeCun 2015). Although these methods can capture the local contextual information in the document, sequence-based models still have difficulty in capturing long-range word co-occurrence information. With rapid adoption of graph neural networks (GNNs) (Kipf and Welling 2016), GNNs can be designed to capture non-consecutive word dependencies in the document. Therefore, GNNs have recently been used for document classification. TextGCN (Yao, Mao, and Luo 2019) first applies GNNs on one corpus graph for node-level document classification task. Huang et al. (Huang et al. 2019) transform TextGCN to graph-level prediction to reduce the memory consumption during training. To improve generalization performance for new documents, there are also works for inductive document classification. TextING (Zhang et al. 2020) constructs an individual graph for each document where local word interactions can be learned. HyperGAT (Ding et al. 2020) improves expressive power of inductive model by exploiting high-order relations in document-level hypergraphs. These methods, with satisfying results, can empirically prove that graph-based models can indeed capture long-range word dependencies which benefits the model performance.

Nevertheless, almost all graph-based methods are designed to construct static word co-occurrence graph for the whole document without considering sentence-level information. Each unique word in the graph is mapped to only one representation in the latent space, which may bring three potential challenges: (1) *Word ambiguity*. In the real world, most words may have multiple meanings, and in different contexts, a single word may have completely different meanings depending on the context. In the static graph, an anchor word with multiple completely different meanings is connected to all adjacent words as *1-hop* neighbors, which can misguide GNNs to blindly combine global information and confuse syntactic information, as well as degrade local information. (2) *Word synonymity*. Non-consecutive words in the static graph can be mapped to similar positions in the latent space, in many cases owing to their proximity to the same anchor word. However, most words have their synonyms, and words adjacent to synonyms should be mapped similarly. Therefore, some long-range information between synonyms may still not be captured in a static word co-occurrence graph. Last, (3) *Dynamic contextual dependency*. Most GNN-based approaches consider nodes and their neighbors to be homogeneous in the static document graph, allowing simultaneous layer-by-layer message passing. However, syntactic and semantic information should be

passed specifically and develop dynamically, rather than at each hierarchy simultaneously. In conclusion, it is necessary to learn the dynamic graph structure of documents with local syntactic and global semantic information with dynamic contextual dependency.

To address the gaps of aforementioned limitations, we propose a novel sparse graph structure learning for inductive document classification, which constructs learnable and individual graphs for each document. Specifically, nodes in the document graph first pass messages to their intra-sentence neighbors and inter-sentence neighbors, which can be regarded as local syntactic messages and global semantic messages, respectively. Then we apply the proposed sparse structure learning with Gumbel-softmax trick to learn and update the graph structure, aiming for dynamic contextual dependencies with fewer noise from layer to layer. The learned graph with local and global information is further fed into a general readout function for classification and optimization in an end-to-end manner. The contribution of this paper is summarized as follows and all the code publicly available at *https://github.com/qkrdmsghk/TextSSL*:

- We construct a trainable individual graph consisting of sentence-level subgraphs for each document. To our best knowledge, we are the first to construct a trainable graph for inductive document classification.
- We propose a sparse structure learning model via GNNs to learn an effective and efficient structure with dynamic syntactic and semantic information for each document.
- We conduct extensive empirical experiments on several real-world. In the experiments, our model outperforms most existing approaches, which supports the effectiveness of our approach.

Preliminaries

Graph Neural Networks

GNNs use the graph structure and node features to learn representation vectors for each node in the graph to conduct the node-level prediction task, or combine all of them to predict property of the graph. Recent researches have focused on spatial-based GNNs that describe a general framework of message passing networks. The essence of the message passing network is to iteratively propagate and aggregate information across the nodes of the graph. Formally, the k-th iteration of message passing process in a GNN consisting of aggregation operation and update operation that is defined as :

$$h_v^k = \phi\left(f^{(k)}(h_v^{(k-1)}, \{h_u^{(k-1)} : u \in \mathcal{N}_v\})\right), \quad (1)$$

where h_v^k denotes the embedding vector at layer k associated with node v, the function $f^{(k)}(\cdot)$ aggregates and updates the node representations from their neighbor nodes at the previous layer. $\phi(\cdot)$ represents an injection function, such as non-linear activation function. For graph classification, the readout function aggregates node representations to obtain the entire graph's representation h_G:

$$h_G = R(\{h_v^{(K)} | v \in G\}). \quad (2)$$

where $R(\cdot)$ denotes a simple permutation invariant function such as global average pooling or global max pooling after K iterations.

Gumbel-Softmax Distribution

Formally, let a discrete variable π has a distribution of probabilities $(\phi_1, ..., \phi_n)$ with class $C = \{c_1, ..., c_n\}$. Gumbel-max (Gumbel 1954) provides an efficient way for the categorical distribution to sample x_π with:

$$x_\pi = \text{argmax}(\log \phi_i + G_i) \quad (3)$$

where G_i is a Gumbel noise sampled from *Gumbel*(0,1). To solve non-differentiable problem of Gumbel-Max, Jang et al. (Jang, Gu, and Poole 2016) propose Gumbel-Softmax to approximate it as follows:

$$\hat{x}_\pi = \frac{\exp((\log(\phi_i) + G_i)/\tau)}{\sum_{j=1}^{n} \exp((\log(\phi_j) + G_j)/\tau)} \quad (4)$$

where a softmax function with an adjustable temperature τ is to control the argmax operation to make it possible for a differentiable optimization.

The Proposed Model

In this section, we introduce our model for inductive document classification. The proposed model consists of three main parts. We first construct sentence graphs in which the node embedding is learned with a local and global message passing operation. Based on the node embedding, we propose a sparse structure learning for the graph structure refinement. Last, we regularize the graph structure to preserve consistency of the original syntactic information.

Problem Definition

We are given a set of documents $\mathcal{D}$ with a set of document labels $\mathcal{Y}$ where each document $d \in \mathcal{D}$ may have multiple sentences $\mathcal{S}^d = [s_0^d, ..., s_k^d]$ and each sentence $s^d \in \mathcal{S}^d$ is composed of multiple words $\mathcal{W}^{s^d} = [{w_0}^{s^d}, ..., {w_n}^{s^d}]$. We represent each document as an individual graph $\mathcal{G}^d$ using its hierarchical structure for inductive learning. For simplicity, we omit the document index d throughout the paper. The document graph is composed of multiple words in $\mathcal{W}^s$ and their connections. Our goal is to learn the structure and make a prediction for each $\mathcal{G}$.

Graph Construction

Definition 1. Sentence-level Subgraph Given a sentence $s_i \in \mathcal{S}$, a sentence-level subgraph $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ can represent the sentence s_i as a word co-occurrence graph. The node set $\mathcal{V}_i$ contains words in sentence s_i. The edge set $\mathcal{E}_i$ contains all connections between any pair of words in $\mathcal{V}_i$ that can co-occur in the same fixed-size sliding window (Mihalcea and Tarau 2004). Therefore, a preliminary document graph $\tilde{\mathcal{G}} = (\mathcal{V}, \mathcal{E})$ can be represented by taking the disjoint union of all sentence-level subgraphs $\mathcal{G}_\mathcal{S} = \{\mathcal{G}_1, ..., \mathcal{G}_n\}$, where n denotes the number of sentences within the document.

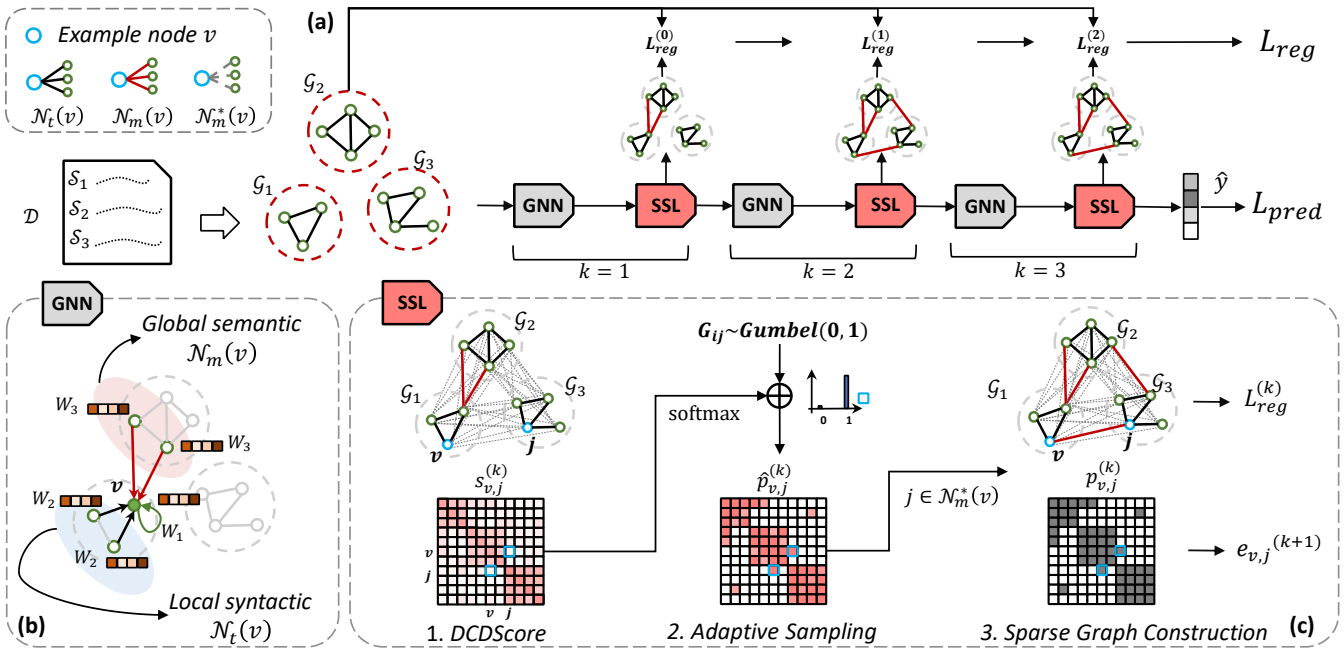

Figure 1: Overview of the proposed model. (a) Model framework. (b) GNN: Local and Global Joint Message Passing. (c) SSL: Sparse Structure Learning contains (c.1) Dynamic Contextual Dependency Score, (c.2) Adaptive Sampling for Sparse Structure, and (c.3) Reconstructing Sparse Graph.

Definition 2. Local Syntactic Neighbor Given a node $v \in \mathcal{V}$ in a preliminary document graph $\tilde{\mathcal{G}}$, we define a local syntactic neighbor $u \in \mathcal{N}_t(v)$ that is adjacent to node v within sentence-level subgraphs $\mathcal{G}_\mathcal{S}$. Since sentence-level subgraphs $\mathcal{G}_\mathcal{S}$ contain relatively more invariant and syntactic information, we constrain the local syntactic neighbors to be static and deterministic during graph structure learning.

Definition 3. Global Semantic Neighbor Given a node $v \in \mathcal{V}$ in a preliminary document graph $\tilde{\mathcal{G}}$, we define a global semantic neighbor $z \in \mathcal{N}_m(v)$ that can have dynamic relation with node v between sentence-level subgraphs $\mathcal{G}_s$. Global semantic neighbors of each node in $\mathcal{V}$ are dynamic and can be learned and selected via structure learning.

A document-level graph $\mathcal{G} = (\mathcal{V}, \{\mathcal{E}_t \cup \mathcal{E}_m\})$ is finally composed of all sentence-level subgraphs $\mathcal{G}_\mathcal{S}$, where edges $\mathcal{E}_t$ connect nodes and their local syntactic neighbors $\mathcal{N}_t(\cdot)$ and edges $\mathcal{E}_m$ connect nodes and their global semantic neighbors $\mathcal{N}_m(\cdot)$. In sparse graph structure learning module, global semantic neighbors can be learned and chosen dynamically during which local syntactic edges can guide the dynamic edge relaxation.

Local and Global Joint Message Passing

Unlike existing GNNs considering that all nodes are homogeneous, we differently aggregate the neighbor messages by distinguishing the neighbor node types (local syntactic neighbor and global semantic neighbor) to update the node representation. The message passing part can be reformulated as:

$$h_v^{(k)} = \phi\left(h_v^{(k-1)}\mathbf{W}_1^{(k)} + t_v^{(k)}\mathbf{W}_2^{(k)} + m_v^{(k)}\mathbf{W}_3^{(k)}\right), \quad (5)$$

where function ϕ denotes an injective function ReLU$(\cdot)$. $h_v^{(k)} \in \mathbb{R}^b$ is the node representation vector and b is the number of hidden dimension. The local syntactic neighbor representations $t_v^{(k)} \in \mathbb{R}^b$ and global semantic neighbor representations $m_v^{(k)} \in \mathbb{R}^b$ can be expressed as:

$$t_v^{(k)} = \sum_{u \in \mathcal{N}_t(v) \cup \{v\}} \frac{e_{u,v}}{\sqrt{\hat{\zeta}_u \hat{\zeta}_v}} h_u^{(k-1)} \quad (6)$$

$$m_v^{(k)} = \sum_{z \in \mathcal{N}_m(v)^{(k-1)}} \frac{e_{z,v}}{\sqrt{\hat{\zeta}_z \hat{\zeta}_v}} h_z^{(k-1)} \quad (7)$$

where $e_{u,v} \in \mathcal{E}_t$ represents edge weight between node v and its local syntactic neighbor u. $e_{z,v} \in \mathcal{E}_m^{(k-1)}$ represents edge weight between node v and its global semantic neighbor z. Here, we normalize the raw edge weight to prevent from the influence of degree bias. Motivated by (Kipf and Welling 2016), we divide both edges $e_{u,v}$ and $e_{z,v}$ by the $\hat{\zeta}_v$ where, $\hat{\zeta}_v = \sum_{j \in \mathcal{N}} \hat{A}_{vj}$ with self-looped adjacency matrix $\hat{A} = A + I$.

With the iteration of local and global message passing operation on $\mathcal{G}$, nodes combine high-order neighbor nodes within sentences $\mathcal{G}_\mathcal{S}$ and also select and combine dynamic neighbor nodes between sentences. In this way, the local syntactic information can be combined with global semantic information in each layer, which can learn the rich contextual information in the document.

Sparse Structure Learning

Since relationships among sentences are not known prior, it is vital to refine the document-level graph $\mathcal{G}$ by exploiting local and global contextual dependencies. One feasible approach is learning from a *Complete Graph* $\mathcal{G}^* = (\mathcal{V}, \{\mathcal{E}_t \cup \mathcal{E}_m^*\})$ where nodes between subgraphs $\mathcal{G}_\mathcal{S}$ are fully connected. However, fully connected words between sentences usually bring both necessary and unnecessary information, a form of noisy information. Therefore, in this section, we perform a sparse structure learning, which can be divided into two parts. (1) Calculating the relative score of global semantic candidate neighbors of each node, which has dynamic contextual dependency. (2) Conducting adaptive hard selection for global semantic candidate neighbors using Gumbel-softmax approach. Finally, global semantic neighbor set is updated and the document-level graph $\mathcal{G}$ can obtain a sparse structure from $\mathcal{G}^*$.

Dynamic Contextual Dependency Score Given a node $v \in \mathcal{V}$ in a complete graph $\mathcal{G}^*$, all neighbors of node v are in $\mathcal{N}^*(v)$, where we can obtain $\mathcal{N}_m^*(v) = \mathcal{N}^*(v) - \mathcal{N}(v)^{(k-1)}$ that contains all global semantic candidate neighbors of node v. We first calculate *attention coefficient score* between each neighbor $j \in \mathcal{N}^*(v)$ and node v as follows:

$$a_{v,j}^{*(k)} = \psi\left(\mathbf{a}^{(k)\top}[h_v^{(k)}\mathbf{W}^{(k)} || h_j^{(k)}\mathbf{W}^{(k)}]\right) \tag{8}$$

where $\mathbf{W}^{(k)} \in \mathbb{R}^{b\times b}$ denotes the projection for node features $h_v \in \mathbb{R}^{1\times b}$ and $h_j \in \mathbb{R}^{n\times b}$. k denotes the current layer of our model. We adopt function ψ as LeakyReLU$(\cdot)$ activation function, and $\mathbf{a} \in \mathbb{R}^{b\times 1}$ is a learnable vector. To consider the correlation between current local and global contextual information, we normalize $a_{v,j}^{*(k)}$ with softmax function across the nodes to calculate dynamic context dependency score:

$$s_{v,j}^{(k)} = \frac{\exp(a_{v,j}^{*(k)})}{\sum_{u\in\mathcal{N}^*(v)}\exp(a_{v,u}^{*(k)})}. \tag{9}$$

We adopt normalization operation on $\mathcal{N}^*(v)$ which both contains: existing neighbors $\mathcal{N}(v)^{(k-1)}$ and the global semantic candidate neighbors $\mathcal{N}_m^*(v)$ that are going to be selected in current layer. The existing neighbors consist of both local syntactic neighbors $\mathcal{N}_t(v)$ and global semantic neighbors $\mathcal{N}_m(v)^{(k-1)}$ that are selected in previous layers. Therefore the score $s_{v,j}^{(k)}$ of global semantic candidate neighbors can elucidate a relative difference compared to existing dynamic contextual dependency of node v.

Sampling Adaptive Neighbors for Sparse Structure Based on the dynamic contextual dependency score $s_{v,j}^{(k)}$, we perform an adaptive sampling on the $\mathcal{G}^*$. To determine sparse edge, we set a threshold to select meaningful global semantic neighbors j for each node v via argmax operation. However, this operation is not be differentiable during back-propagation process to optimize model. Inspired from (Jang, Gu, and Poole 2016), we first generate a neighbor selector $p_{v,j}^{(k)} \in \{0, 1\}$ from the Bernoulli distribution $\{\pi_1 := s_{v,j}^{(k)}, \pi_0 := 1 - s_{v,j}^{(k)}\}$ and adopt Gumbel-Softmax approach to generate differentiable probability $\hat{p}_{v,j}^{(k)}$ of selector samples $p_{v,j}^{(k)}$ as follows:

$$\hat{p}_{v,j}^{(k)} = \frac{\exp((\log\pi_1 + g_1)/\tau)}{\sum_{i\in\{0,1\}}\exp((\log\pi_i + g_i)/\tau)}, \tag{10}$$

where g_1 and g_0 are i.i.d variables sampled from Gumbel distribution, and $\tau \in (0, \infty)$ denotes temperature parameter. As $\tau \to 0$, $\hat{p}_{v,j}^{(k)}$ can be annealed to categorical distribution. Then we can get a discrete neighbor selector $p_{v,j}^{(k)}$ by setting a threshold T.

Reconstructing Sparse Graph Thus, we can select informative neighbors for node v using $p_{v,j}^{(k)}$. It is worth mentioning that despite $p_{v,j}^{(k)}$ is obtained from $s_{v,j}^{(k)}$ that is calculated by both existing neighbors and global semantic candidate neighbors, we only select the neighbors from the candidate neighbor set to maintain the local syntactic topology of the document graph. Specifically, we update the global semantic neighbors $\mathcal{N}_m(v)^{(k)}$ for node v with selected candidate neighbors as follows:

$$\mathcal{N}_m(v)^{(k)} = \mathcal{N}_m(v)^{(k-1)} \cup \{j\,[|]\,\forall j \to p_{v,j}^{(k)} = 1\}. \tag{11}$$

where $j \in \mathcal{N}_m^*(v)$. In addition, for static local syntactic neighbors $\mathcal{N}_t(v)$, we compute the entropy to preserve consistency of the original syntactic information and prevent too much structure variation in the graph.

$$L_{reg}^{(k)} = \sum_{v\in\mathcal{V}}\sum_{j\in\mathcal{N}_t(v)} -\hat{p}_{v,j}^{(k)}\log(\hat{p}_{v,j}^{(k)}), \tag{12}$$

At the last iteration, all of the nodes in the graph $\mathcal{G}$ are fed into readout function with simple summation operation and linear operation. We use cross entropy loss function $l(\cdot,\cdot)$ to measure prediction and true label y of the document.

$$L_{pred} = l(R(h_v), y), \tag{13}$$

We therefore minimize the loss by summing prediction loss L_{pred} and averaged regularization loss $\lambda\sum_k L_{reg}^{(k)}$ for each document classification tasks, where λ is a hyperparameter to adjust the trade-offs between newly learned structure and original structure.

Experiments

Datasets

For a fair and comprehensive evaluation, we use the same benchmark datasets that are used in (Yao, Mao, and Luo 2019). There are five datasets in three different domains, including sentiment analysis, news classification, and topic classification domains. We use MR dataset for binary sentiment analysis with either positive or negative polarity. We use three datasets in news classification. 20NG is a newsgroup file dataset with 20 categories and reasonably balanced. R8 and R52 are two subsets of Reuters 21578 (Moschitti 2003) datasets with 8 and 52 categories respectively and both two datasets are extremely imbalanced. Ohsumed is a topic classification dataset consisting of medical abstracts with 23 categories, such as cardiovascular disease.

Dataset	#Docs	#Training	#Test	#Classes (ρ)	#Vocab.	Avg.#Length	Avg.#Sentence	#Prop.NW
MR	10,662	7,108	3,554	2 (1.0)	18,764	20.39	1.17	30.09%
R8	7,674	5,485	2,189	8 (84.7)	7,688	65.72	4.03	2.60%
R52	9,100	6,532	2,568	52 (1666.7)	8,892	69.82	4.34	2.63%
Ohsumed	7,400	3,357	4,034	23 (62.5)	14,157	135.82	8.59	8.46%
20NG	18,846	11,314	7,532	20 (1.6)	42,757	221.26	6.06	7.40%

Table 1: Statistics of the datasets. ρ denotes class imbalance ratio (the sample size of the most frequent class divided by that of the least frequent class). The Avg.#Length and the Avg.#Sentence mean the number of words and the number of sentences in a document, respectively. The #Prop.NW denotes the proportion of new words in test.

Categories	**Baselines**	**MR**	**R8**	**R52**	**Ohsumed**	**20NG**
Word-based	fastText	72.17±1.30	86.04±0.24	71.55±0.42	14.59±0.00	11.38±1.18
	SWEN	76.65±0.63	95.32±0.26	92.94±0.24	63.12±0.55	85.16±0.29
Sentence-based	CNN-non-static	77.75±0.72	95.71±0.52	87.59±0.48	58.44±1.06	82.15±0.52
	LSTM (pretrain)	77.33±0.89	96.09±0.19	90.48±0.86	51.10±1.50	75.43±1.72
	Bi-LSTM	77.68±0.86	96.31±0.33	90.54±0.91	49.27±1.07	73.18±1.85
Graph-based (Tr)	TextGCN	76.74±0.20	97.07±0.10	93.56±0.18	68.36±0.56	86.34±0.09
	Huang et al.	-	97.80±0.20	94.60±0.30	69.40±0.60	-
	TensorGCN	77.91±0.07	**98.04±0.08**	95.05±0.11	70.11±0.24	**87.74±0.05**
	DHTG	77.21±0.11	97.33±0.06	93.93±0.10	68.80±0.33	87.13±0.07
Graph-based (Ind)	TextING	78.93±0.65	97.34±0.25	93.73±0.47	67.95±0.52	OOM
	HyperGAT	77.36±0.22	96.82±0.21	94.15±0.18	66.39±0.65	84.65±0.31
	Our proposal	**79.74±0.19**	97.81±0.14	**95.48±0.26**	**70.59±0.38**	85.26±0.28

Table 2: Test accuracies of various models on five benchmark datasets. The mean ± standard deviation of all models are reported an average of 10 executions of each model. Graph-based (Tr) means transductive graph-based methods and Graph-based (Ind) means inductive graph-based methods.

Experimental Settings

For quantitative evaluation, we follow the same train/test splits and data preprocessing for MR, Ohsumed and 20NG datasets as (Kim 2014; Yao, Mao, and Luo 2019). For R8 and R52 datasets, they are only provided by a preprocessed version that lack punctuations and do not have explicit sample names. Since we use documents with sentence segmentation information to construct graph, we re-extract the data from original Reuters-21578 dataset. More details on the preprocessing of R8 and R52 dataset are provided in Appendix. In each experiment, we randomly select 10% documents from the training set to build validation set. A statistics of the benchmark dataset is listed in Table 1.

Baseline Methods

In our experiments, the baselines are divided into three categories: word based methods, sequence based methods and graph based methods. In *word-based* methods, we use fastText (Joulin et al. 2016) and SWEM (Shen et al. 2018). In *sequence-based* methods, we use CNN (Kim 2014) with pretrained word embedding, and RNN (Liu, Qiu, and Huang 2016) with pretrained word embedding and its variant models LTSM from (Yao, Mao, and Luo 2019). *Graph-based* models for document classification can be categorized into transductive learning and inductive learning. We compare a series of GNN-based transductive models, such as TextGCN (Yao, Mao, and Luo 2019), TensorGCN (Liu et al. 2020), DHTG (Wang et al. 2020), Huang et al. (Huang et al. 2019). We also compare with recent published inductive models, such as HyperGAT (Ding et al. 2020), TextING (Zhang et al. 2020). Detail of these methods are provided in related works.

Parameter Settings

In this part, we describe the hyperparameter ranges for model training. First, we search GNN layers from {2, 3}. Batch size is chosen from {16, 64, 128, 256}. We set the initial node dimension to 300, and then we search the hidden node dimension from {96, 256, 512}. The hyperparameters of all datasets are reported in the Appendix. We use Adam (Kingma and Ba 2014) to optimize the model. We use PyTorch (Paszke et al. 2019) to implement our architecture. For Texting and HyperGAT baselines, we use the same datasets for fair comparisons. All models are trained on a single NVIDIA GeForce RTX 3080 GPU. For baseline models, we either show the results reported in previous research (Yao, Mao, and Luo 2019) or run the codes provided by the authors using the parameters described in the original papers. More details can be found in the Appendix.

Experiment Results

Table 2 shows performance comparisons of the different methods on five benchmark datasets. Firstly, most graph-based methods outperform both word-based and sequence-based baselines, indicating the long-range dependencies captured by graph-based models benefit document classification. Next, we compare our model with transductive and inductive graph-based models, respectively. Overall, our model achieves best results among all inductive learning models, indicating that the sparse graph structure learned from our model using dynamic contextual information has a positive effect on inductive learning. To summarize, our observations are as follows:

Unseen words. It is noted in Table 2 that our model significantly outperforms in MR dataset. According to the #Prop.NW in Table 1, we can find that there are many unseen words in the test set, which indicates that the sparse structure of the documents learned by our model using inductive learning favors the generalization ability.

Document length. From Table 1 and Table 2, we find a trend that inductive models perform better on short documents (MR, R8, and R52), while most transductive methods perform relatively well on long documents (Osumed, 20NG). It seems that long documents own denser structures that can benefit message passing for the transductive methods. For inductive learning, the dense structure introduces additional noise, which makes the learning of the model difficult. Even so, our model combines syntax and global semantics to learn sparse graphs for documents. Therefore the proposed model outperforms all baselines on Ohsumed dataset and all existing inductive methods on 20NG dataset.

Dynamic contextual dependency. Most notably, our model and TensorGCN achieve the best performance in the inductive and transductive models, respectively. TensorGCN, like our model, also takes into account both syntactic and sequential information. This suggests that considering sequential, syntactic and semantic information in these datasets can help a lot in document classification. Unlike TensorGCN, our model is able to perform inductive learning and can leverage the learned sparse dynamic contextual dependencies from rich structured document to improve generalization performance in more complex classification tasks, e.g., unbalanced (R52) and domain-specific (Ohsumed) datasets.

Issues in Constructing Document-level Graph

In this subsection, we analyze the effectiveness of different ways to construct document-level graphs for the document classification task: (1) Word co-occurrence graph, (2) disjoint graph, (3) complete graph and (4) our graphs. The word co-occurrence graph is created by a simple sliding window method that is not considering the sentence information. Then we learn a disjoint graph in the model by setting threshold T (in the equation 10) equals to 1, which can lead to none of edges are sampled during structure learning, thus the inter sentence information is completely ignored and the graph only focuses on the intra-sentence information. On the contrary, we may set threshold to 0 to learn a complete graph, where all words inter-sentence are connected to each other. Ideally, this complete graph would have been able to learn relatively global information, however, the sharp increase in the number of edges dissolves the information within sentences, and it cannot learn informative features to perform the task effectively.

Graph	R8	R52	Ohsumed
WordCooc	97.20±0.29	93.82±0.15	68.08±0.32
Disjoint	97.29±0.21	94.80±0.20	69.72±0.27
Complete	97.40±0.25	94.35±0.10	67.57±0.30
Ours	97.76±0.16	95.32±0.21	70.53±0.30
Ours w/ reg	**97.81±0.14**	**95.48±0.26**	**70.59±0.38**

Table 3: Comparison with different constructions of document-level graphs. (1) WordCooc denotes word co-occurrence graph. (2) Disjoint means a disjoint union of sentence-level subgraphs. (3) Complete graph means disjoint graph with fully connected edges between sentences. (4) Ours graph is constructed by sentence-level subgraphs and learned by sparse structure learning(w/ reg means we add regularization to our model).

τ	R8	R52	Ohsumed
0.01	97.50±0.29	95.16±0.18	**70.59±0.38**
0.1	97.34±0.13	**95.48±0.26**	70.21±0.40
0.2	97.44±0.39	95.03±0.16	70.33±0.32
0.5	**97.81±0.14**	94.56±0.33	70.34±0.37
1.0	97.35±0.24	95.09±0.32	70.22±0.29

Table 4: Test accuracy with different temperatures τ for adaptive sampling.

The experimental results of these graphs and our graphs are shown in Table 3. We note that our graphs perform the best. This suggests that (1) it is useful to use sentence information to construct document-level graph for classification tasks, allowing for word sense disambiguation and capturing synonyms. And (2) sparse structures learned from dynamic contexts in documents can help improve the generalization of document classification. It is worth noting that disjoint and complete graph have different results on different datasets, indicating that each documents owns its characteristics and they need to be learned adaptively according to the goal of document classification.

Adaptive Sampling Analysis

To obtain a proper sampling temperature for each dataset, we set five temperatures of each dataset for experiments in table 4. During sampling the adaptive neighbors for each nodes via Gumbel-softmax with an adjustable temperature, the smaller the temperature, the more the samples tend to be categorically distributed, which in our model represents the document graph learning to a more sparse structure. From the results, we can see that different datasets have different proper temperatures, and the Ohsumed dataset reaches the

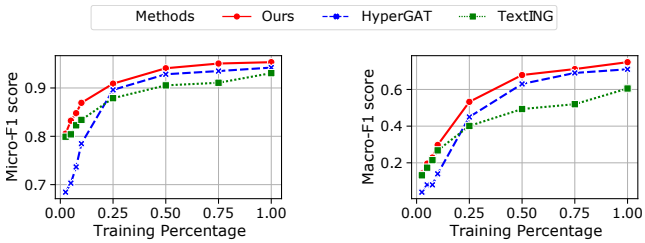

Figure 2: Micro F1 score and Macro F1 score with different percent of training data from 0.025 to 1 on R52 dataset.

appropriate temperature at a very small value, which meets the same conclusion as the ablation study in the previous section, i.e., the learned sparse structures are adaptive for topic label classification for Ohsumed dataset. In Appendix, more details for the temperatures of the other datasets and a real case study of visualizing important sparse connections that are learned by our model are provided.

Generalization on Imbalanced Unlabeled Data

To estimate the generalization ability of our model over imbalanced dataset, we use different proportions of training data on R52 dataset to compare with the inductive learning baselines. As the number of training sets decreases, it becomes more challenging to train model on extremely imbalanced datasets. Figure 2 shows that all inductive methods achieve improved performance as the number of labeled training data increases. Our method significantly outperforms other inductive baselines in all cases, indicating that the sparse document structure learned using sentence information allows our model to generalize well even in extremely imbalanced dataset.

Related Works

Document classification

Document classification is one of the most fundamental tasks in the field of natural language processing (NLP). Document classification is widely used in many downstream applications, such as spam filtering (Wu et al. 2020a), news classification (Liu and Wu 2018), sentiment analysis (Medhat, Hassan, and Korashy 2014), etc. An important part of document classification is feature extraction. The traditional approach is to use word-based statistical models to compute features of documents and apply them to downstream classifiers, such as support vector machines (Suykens and Vandewalle 1999), naive Bayes (McCallum, Nigam et al. 1998), random forests (Svetnik et al. 2003), etc. With the rapid development of deep learning, many feature-based deep learning models have been proposed, such as the study of word representation learning models (Mikolov et al. 2013; Grover and Leskovec 2016). Considering the word order in sequences, many models use sequence-based models, including recurrent neural networks (RNNs) (Mikolov et al. 2010; Liu, Qiu, and Huang 2016), convolutional neural networks (CNNs) (Kim 2014). In addition, there are also models, such as (Yang et al. 2016; Peng et al. 2021), leverage document hierarchical structures to jointly consider word order-sentence order information. Since the above models are studied based on sequence data, the dependencies between long sentences may not be taken into account. Inspired by the semi-supervised GNN proposed by (Kipf and Welling 2016), the study of transforming documents into data structured as graphs and optimizing GNN parameter learning model on the document graphs has rapidly gained attention.

GNNs for Document Classification

The popularity of GNNs have grown rapidly in recent years (Kipf and Welling 2016; Veličković et al. 2017; Hamilton, Ying, and Leskovec 2017; Xu et al. 2018). In natural language domain, GNN can better capture the non-consecutive phrases and long-distance word dependency in the documents (Wu et al. 2020b; Li et al. 2020). Recent works on GNNs for document classification can be divided into two categories. One is transductive learning. TextGCN (Yao, Mao, and Luo 2019) first applies GNNs on a whole corpus graph. Huang et al. (Huang et al. 2019) build graphs for each document with global shared structure to support effective online learning. TensorGCN (Liu et al. 2020) jointly learns syntactic, semantic and sequential on a document graph tensor based on a whole corpus. DHTG (Wang et al. 2020) propose a novel hierarchical topic graph to learn an corpus graph with meaningful node embeddings and semantic edges. While transductive learning models have to be retrained once evaluating an unseen documents, which is unrealistic in the real world. On the other hand, inductive models can solve this problem. Peng et al. (Peng et al. 2018) present a graph-based model to perform hierarchical text classification. TextING (Zhang et al. 2020) builds individual word co-occurrence graphs for each document and shows a better generalization performance on unseen documents. HyperGAT (Ding et al. 2020) proposes novel document-level hypergraphs and inject topic information to obtain high-order semantic information in each document. However,the hypergraphs with pre-defined latent topics lacks local syntactic information. Our proposed model makes first attempt to leverage sequence information to construct novel document-level graphs that can jointly aggregate local syntactic and global semantic information to learn fine-grained word representations for autonomous inductive document classification.

Conclusion

In the real world, each document has its own rich sentence structure, where the intra-sentence context contains local information, and the inter-sentence context captures long-range word dependencies. To this end, we construct a novel trainable document-level graph to jointly capture local and global contextual information. We propose a sparse structure learning via GNNs to refine the structure of the graph with learned dynamic context dependencies. Experimental results show that our proposed approach of combining and learning local and global information is effective for inductive document classification.

Acknowledgements

This research was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) [NO.2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University)] (IITP-2021-0-01343) and by the Bio & Medical Technology Development Program of the National Research Foundation (NRF) funded by the Ministry of Science & ICT (NRF-2019M3E5D4065965) and by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2021R1A6A3A01086898).

References

Ding, K.; Wang, J.; Li, J.; Li, D.; and Liu, H. 2020. Be more with less: Hypergraph attention networks for inductive text classification. *arXiv preprint arXiv:2011.00387*.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 855–864.

Gumbel, E. J. 1954. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 1025–1035.

Huang, L.; Ma, D.; Li, S.; Zhang, X.; and Wang, H. 2019. Text level graph neural network for text classification. *arXiv preprint arXiv:1910.02356*.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

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

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Li, Q.; Peng, H.; Li, J.; Xia, C.; Yang, R.; Sun, L.; Yu, P. S.; and He, L. 2020. A survey on text classification: From shallow to deep learning. *arXiv preprint arXiv:2008.00364*.

Liu, P.; Qiu, X.; and Huang, X. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Liu, X.; You, X.; Zhang, X.; Wu, J.; and Lv, P. 2020. Tensor graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 8409–8416.

Liu, Y.; and Wu, Y.-F. B. 2018. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In *Thirty-second AAAI conference on artificial intelligence*.

McCallum, A.; Nigam, K.; et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, 41–48. Citeseer.

Medhat, W.; Hassan, A.; and Korashy, H. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4): 1093–1113.

Mihalcea, R.; and Tarau, P. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, 404–411.

Mikolov, T.; Karafiát, M.; Burget, L.; Černockỳ, J.; and Khudanpur, S. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Moschitti, A. 2003. Text Categorization Corpora. http://disi.unitn.it/moschitti/corpora.htm. Accessed: 2021-09-08.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32: 8026–8037.

Peng, H.; Li, J.; He, Y.; Liu, Y.; Bao, M.; Wang, L.; Song, Y.; and Yang, Q. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 world wide web conference*, 1063–1072.

Peng, H.; Li, J.; Wang, S.; Wang, L.; Gong, Q.; Yang, R.; Li, B.; Yu, P. S.; and He, L. 2021. Hierarchical Taxonomy-Aware and Attentional Graph Capsule RCNNs for Large-Scale Multi-Label Text Classification. *IEEE Transactions on Knowledge and Data Engineering*, 33(6): 2505–2519.

Shen, D.; Wang, G.; Wang, W.; Min, M. R.; Su, Q.; Zhang, Y.; Li, C.; Henao, R.; and Carin, L. 2018. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. *arXiv preprint arXiv:1805.09843*.

Suykens, J. A.; and Vandewalle, J. 1999. Least squares support vector machine classifiers. *Neural processing letters*, 9(3): 293–300.

Svetnik, V.; Liaw, A.; Tong, C.; Culberson, J. C.; Sheridan, R. P.; and Feuston, B. P. 2003. Random forest: a classification and regression tool for compound classification and QSAR modeling. *Journal of chemical information and computer sciences*, 43(6): 1947–1958.

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wang, Z.; Wang, C.; Zhang, H.; Duan, Z.; Zhou, M.; and Chen, B. 2020. Learning dynamic hierarchical topic graph with graph convolutional network for document classification. In *International Conference on Artificial Intelligence and Statistics*, 3959–3969. PMLR.

Wu, Y.; Lian, D.; Xu, Y.; Wu, L.; and Chen, E. 2020a. Graph convolutional networks with markov random field reasoning for social spammer detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 1054–1061.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020b. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; and Hovy, E. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 1480–1489.

Yao, L.; Mao, C.; and Luo, Y. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 7370–7377.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28: 649–657.

Zhang, Y.; Yu, X.; Cui, Z.; Wu, S.; Wen, Z.; and Wang, L. 2020. Every document owns its structure: Inductive text classification via graph neural networks. *arXiv preprint arXiv:2004.13826*.