Graph Convolutional Networks for Text Classification

Liang Yao
Zhejiang University
32 PUBLICATIONS **78** CITATIONS
SEE PROFILE

Chengsheng Mao
Northwestern University
41 PUBLICATIONS **94** CITATIONS
SEE PROFILE

Yuan Luo
Nanchang Hangkong University
147 PUBLICATIONS **2,068** CITATIONS
SEE PROFILE

Graph Convolutional Networks for Text Classification

Liang Yao, Chengsheng Mao, Yuan Luo[*]
Northwestern University
Chicago IL 60611
{liang.yao, chengsheng.mao, yuan.luo}@northwestern.edu

Abstract

Text Classification is an important and classical problem in natural language processing. There have been a number of studies that applied convolutional neural networks (convolution on regular grid, e.g., sequence) to classification. However, only a limited number of studies have explored the more flexible graph convolutional neural networks (convolution on non-grid, e.g., arbitrary graph) for the task. In this work, we propose to use graph convolutional networks for text classification. We build a single text graph for a corpus based on word co-occurrence and document word relations, then learn a Text Graph Convolutional Network (Text GCN) for the corpus. Our Text GCN is initialized with one-hot representation for word and document, it then jointly learns the embeddings for both words and documents, as supervised by the known class labels for documents. Our experimental results on multiple benchmark datasets demonstrate that a vanilla Text GCN without any external word embeddings or knowledge outperforms state-of-the-art methods for text classification. On the other hand, Text GCN also learns predictive word and document embeddings. In addition, experimental results show that the improvement of Text GCN over state-of-the-art comparison methods become more prominent as we lower the percentage of training data, suggesting the robustness of Text GCN to less training data in text classification.

Introduction

Text classification is a fundamental problem in natural language processing (NLP). The task is to annotate a given text sequence with one or multiple class label(s) describing its textual content. There are numerous applications of text classification such as document organization, news filtering, spam detection, opinion mining, and computational phenotyping, see (Aggarwal and Zhai 2012) and (Zeng et al. 2018) for reviews. An essential intermediate step for text classification is text representation. Traditional methods represent text with hand-crafted features, such as sparse lexical features (e.g., bag-of-words and n-grams). Recently, deep learning models have been widely used to learn text representations, including convolutional neural networks (CNN) (Kim 2014) and recurrent neural networks (RNN) such as long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997). As CNN and RNN prioritize locality and sequentiality (Battaglia et al. 2018), these deep learning models can capture semantic and syntactic information in local consecutive word sequences well, but may ignore global word co-occurrence in a corpus which carries non-consecutive and long-distance semantics (Peng et al. 2018).

Recently, a new research direction called graph neural networks or graph embeddings has attracted wide attention (Battaglia et al. 2018; Cai, Zheng, and Chang 2018). Graph neural networks have been effective at tasks thought to have rich relational structure and can preserve global structure information of a graph in graph embeddings.

In this work, we propose a new graph neural network-based method for text classification. We construct a single large graph from an entire corpus, which contains word and document as nodes. We model the graph with a Graph Convolutional Network (GCN) (Kipf and Welling 2017), a simple and effective graph neural network which can capture high order neighborhoods information. The edge between two word nodes is built by word co-occurrence information and the edge between a word node and document node is built using word frequency and word's document frequency. We then naturally turn text classification problem into a node classification problem. The method can achieve strong classification performances with a small proportion of labeled documents and learn interpretable word and document node embeddings.

We made our source code publicly available at `https://github.com/yao8839836/text_gcn` for reproducibility. To summarize, our contributions are as follows:

- We propose a novel graph neural network method for text classification. To the best of our knowledge, this is the first study to model a whole corpus as a heterogeneous graph and learn word and document embeddings with graph neural networks jointly.
- Experimental results on several benchmark datasets demonstrate that our method outperforms state-of-the-art text classification methods, without using any pre-trained word embeddings or external knowledge. Our method can also learn predictive word and document embeddings automatically.

[*]Corresponding Author

Related Work

Traditional Text Classification

Traditional text classification studies mainly focus on feature engineering and classification algorithms. For feature engineering, the most commonly used feature is the bag-of-words feature. In addition, some more complex features have been designed, such as n-grams (Wang and Manning 2012) and entities in ontologies (Chenthamarakshan et al. 2011). There are also existing studies on converting texts to graphs and perform feature engineering on graphs (Luo, Uzuner, and Szolovits 2016). (Rousseau, Kiagias, and Vazirgiannis 2015) treated text classification as a graph classification problem. They represented documents as graphs of words and mine frequent subgraphs as features for classification. (Skianis, Rousseau, and Vazirgiannis 2016) use graph-of-words as the regularization of classifiers' training loss to improve classification performance. Unlike these methods, our method can learn text representations as node embeddings automatically.

Deep Learning for Text Classification

Deep learning text classification studies can be categorized into two groups. One group of studies focused on models based on word embeddings (Mikolov et al. 2013; Pennington, Socher, and Manning 2014). Several recent studies showed that the success of deep learning on text classification largely depends on the effectiveness of the word embeddings (Shen et al. 2018; Joulin et al. 2017; Wang et al. 2018). Some authors aggregated unsupervised word embeddings as document embeddings then feed these document embeddings into a classifier (Le and Mikolov 2014; Joulin et al. 2017). Others jointly learned word/document and document label embeddings (Tang, Qu, and Mei 2015; Wang et al. 2018). Our work is connected to these methods, the major difference is that these methods build text representations after learning word embeddings while we learn word and document embeddings simultaneously for text classification.

Another group of studies employed deep neural networks. Two representative deep networks are CNN and RNN. (Kim 2014) used CNN for text classification. The architecture is a direct application of CNNs as used in computer vision but with one dimensional convolutions. (Zhang, Zhao, and LeCun 2015) designed a character level CNN and achieved promising results. (Conneau et al. 2017) applied very deep CNNs on characters with convolution and pooling operations on small windows. (Tai, Socher, and Manning 2015) and (Liu, Qiu, and Huang 2016) used LSTM, a specific type of RNN, to learn text representation. To further increase the representation flexibility of such models, attention mechanisms have been introduced as an integral part of models employed for text classification (Yang et al. 2016; Wang et al. 2016). Although these methods were effective and widely used, they mainly focused on local consecutive word sequences, but did not explicitly use global word co-occurrence information in a corpus.

Graph Neural Networks

The topic of Graph Neural Networks has received growing attentions recently (Cai, Zheng, and Chang 2018; Battaglia et al. 2018). A number of authors generalized well-established neural network models like CNN that apply to regular grid structure (2-d mesh or 1-d sequence) to work on arbitrarily structured graphs (Bruna et al. 2014; Henaff, Bruna, and LeCun 2015; Defferrard, Bresson, and Vandergheynst 2016; Kipf and Welling 2017). In their pioneering work, Kipf and Welling presented a simplified graph neural network model, called graph convolutional networks (GCN), which achieved state-of-the-art classification results on a number of benchmark graph datasets (Kipf and Welling 2017). GCN was also explored in several NLP tasks such as semantic role labeling (Marcheggiani and Titov 2017) and machine translation (Bastings et al. 2017), where GCN is used to encode syntactic structure of sentences. Some recent studies explored graph neural networks for text classification (Henaff, Bruna, and LeCun 2015; Defferrard, Bresson, and Vandergheynst 2016; Kipf and Welling 2017; Peng et al. 2018; Zhang, Liu, and Song 2018). However, they either viewed a document or a sentence as a graph of word nodes (Defferrard, Bresson, and Vandergheynst 2016; Peng et al. 2018; Zhang, Liu, and Song 2018) or relied on the not-routinely-available document citation relation to construct the graph (Kipf and Welling 2017). In contrast, when constructing the corpus graph, we regard the documents and words as nodes (hence heterogeneous graph) and do not require inter-document relations.

Method

Graph Convolutional Networks (GCN)

A GCN (Kipf and Welling 2017) is a multilayer neural network that operates directly on a graph and induces embedding vectors of nodes based on properties of their neighborhoods. Formally, consider a graph $G = (V, E)$, where $V(|V| = n)$ and E are sets of nodes and edges, respectively. Every node is assumed to be connected to itself, i.e., $(v, v) \in E$ for any v. Let $X \in \mathbb{R}^{n \times m}$ be a matrix containing all n nodes with their features, where m is the dimension of the feature vectors, each row $x_v \in \mathbb{R}^m$ is the feature vector for v. We introduce an adjacency matrix A of G and its degree matrix D, where $D_{ii} = \sum_j A_{ij}$. The diagonal elements of A are set to 1 because of self-loops. GCN can capture information only about immediate neighbors with one layer of convolution. When multiple GCN layers are stacked, information about larger neighborhoods are integrated. For a one-layer GCN, the new k-dimensional node feature matrix $L^{(1)} \in \mathbb{R}^{n \times k}$ is computed as

$$L^{(1)} = \rho(\tilde{A} X W_0) \tag{1}$$

where $\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ is the normalized symmetric adjacency matrix and $W_0 \in \mathbb{R}^{m \times k}$ is a weight matrix. ρ is an activation function, e.g. a ReLU $\rho(x) = \max(0, x)$. As mentioned before, one can incorporate higher order neighborhoods information by stacking multiple GCN layers:

$$L^{(j+1)} = \rho(\tilde{A} L^{(j)} W_j) \tag{2}$$

where j denotes the layer number and $L^{(0)} = X$.

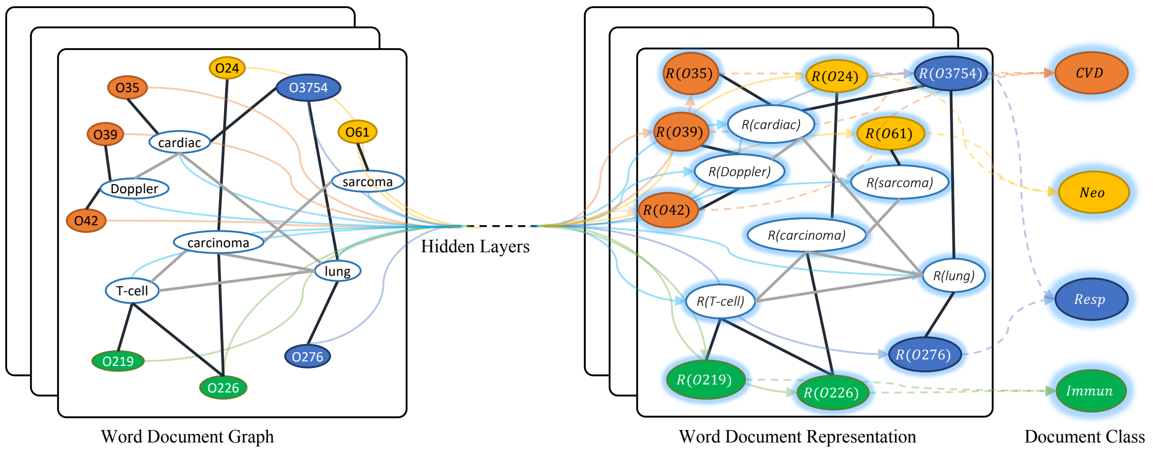


Figure 1: The schematic depiction of our Text GCN. The word document graph example are taken from Ohsumed corpus in the experiment. Nodes begin with "O" are document nodes, others are word nodes. Black bold edges are document-word edges and gray thin edges are word-word edges. $R(x)$ means the representation (embedding) of x. Different colors mean different document classes (only four example classes are shown to avoid clutter). CVD: Cardiovascular Diseases, Neo: Neoplasms, Resp: Respiratory Tract Diseases, Immun: Immunologic Diseases.

Text Graph Convolutional Networks (Text GCN)

We build a large and heterogeneous text graph which contains word nodes and document nodes so that global word co-occurrence can be explicitly modeled and graph convolution can be easily adapted, as shown in Figure 1. The number of nodes in the text graph $|V|$ is the number of documents (corpus size) plus the number of unique words (vocabulary size) in a corpus. We simply set feature matrix $X = I$ as an identity matrix which means every word or document is represented as a one-hot vector as the input to Text GCN. We build edges among nodes based on word occurrence in documents (document-word edges) and word co-occurrence in the whole corpus (word-word edges). The weight of the edge between a document node and a word node is the term frequency-inverse document frequency (TF-IDF) of the word in the document, where term frequency is the number of times the word appears in the document, inverse document frequency is the logarithmically scaled inverse fraction of the number of documents that contain the word. We found using TF-IDF weight is better than using term frequency only. To utilize global word co-occurrence information, we use a fixed size sliding window on all documents in the corpus to gather co-occurrence statistics. We employ point-wise mutual information (PMI), a popular measure for word associations, to calculate weights between two word nodes. We also found using PMI achieves better results than using word co-occurrence count in our preliminary experiments. Formally, the weight of edge between node i and node j is defined as

$$A_{ij} = \begin{cases} \mathrm{PMI}(i,j) & i \text{ and } j \text{ are words and } \mathrm{PMI}(i,j) > 0 \\ \text{TF-IDF}_{ij} & i \text{ is a document and } j \text{ is a word} \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The PMI value of a word pair i, j is computed as

$$\mathrm{PMI}(i,j) = \log \frac{p(i,j)}{p(i)p(j)} \tag{4}$$

$$p(i,j) = \frac{\#W(i,j)}{\#W} \tag{5}$$

$$p(i) = \frac{\#W(i)}{\#W} \tag{6}$$

where $\#W(i)$ is the number of sliding windows in a corpus that contain word i, $\#W(i,j)$ is the number of sliding windows that contain both word i and j, and $\#W$ is the total number of sliding windows in the corpus. A positive PMI value implies a high semantic correlation of words in a corpus, while a negative PMI value indicates little or no semantic correlation in the corpus. Therefore, we only add edges between word pairs with positive PMI values.

After building the text graph, we feed the graph into a simple two layer GCN as in (Kipf and Welling 2017), the second layer node (word/document) embeddings have the same size as the labels set and are fed into a *softmax* classifier:

$$Z = \mathrm{softmax}(\tilde{A}\ \mathrm{ReLU}(\tilde{A}XW_0)W_1) \tag{7}$$

where $\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the same as in equation 1, and $\mathrm{softmax}(x_i) = \frac{1}{\mathcal{Z}}\exp(x_i)$ with $\mathcal{Z} = \sum_i \exp(x_i)$. The loss function is defined as the cross-entropy error over all labeled documents:

$$\mathcal{L} = -\sum_{d \in \mathcal{Y}_D} \sum_{f=1}^{F} Y_{df} \ln Z_{df} \tag{8}$$

where $\mathcal{Y}_D$ is the set of document indices that have labels and F is the dimension of the output features, which is equal to the number of classes. Y is the label indicator matrix. The weight parameters W_0 and W_1 can be trained via gradient descent. In equation 7, $E_1 = \tilde{A}XW_0$ contains the first layer document and word embeddings and $E_2 = \tilde{A}\,\mathrm{ReLU}(\tilde{A}XW_0)W_1$ contains the second layer document and word embeddings. The overall Text GCN model is schematically illustrated in Figure 1.

A two-layer GCN can allow message passing among nodes that are at maximum two steps away. Thus although there is no direct document-document edges in the graph, the two-layer GCN allows the information exchange between pairs of documents. In our preliminary experiment. We found that a two-layer GCN performs better than a one-layer GCN, while more layers did not improve the performances. This is similar to results in (Kipf and Welling 2017) and (Li, Han, and Wu 2018).

Experiment

In this section we evaluate our Text Graph Convolutional Networks (Text GCN) on two experimental tasks. Specifically we want to determine:

- Can our model achieve satisfactory results in text classification, even with limited labeled data?
- Can our model learn predictive word and document embeddings?

Baselines. We compare our Text GCN with multiple state-of-the-art text classification and embedding methods as follows:

- **TF-IDF + LR** : bag-of-words model with term frequency-inverse document frequency weighting. Logistic Regression is used as the classifier.
- **CNN**: Convolutional Neural Network (Kim 2014). We explored CNN-rand which uses randomly initialized word embeddings and CNN-non-static which uses pre-trained word embeddings.
- **LSTM**: The LSTM model defined in (Liu, Qiu, and Huang 2016) which uses the last hidden state as the representation of the whole text. We also experimented with the model with/without pre-trained word embeddings.
- **Bi-LSTM**: a bi-directional LSTM, commonly used in text classification. We input pre-trained word embeddings to Bi-LSTM.
- **PV-DBOW**: a paragraph vector model proposed by (Le and Mikolov 2014), the orders of words in text are ignored. We used Logistic Regression as the classifier.
- **PV-DM**: a paragraph vector model proposed by (Le and Mikolov 2014), which considers the word order. We used Logistic Regression as the classifier.
- **PTE**: predictive text embedding (Tang, Qu, and Mei 2015), which firstly learns word embedding based on heterogeneous text network containing words, documents and labels as nodes, then averages word embeddings as document embeddings for text classification.
- **fastText**: a simple and efficient text classification method (Joulin et al. 2017), which treats the average of word/n-grams embeddings as document embeddings, then feeds document embeddings into a linear classifier. We evaluated it with and without bigrams.
- **SWEM**: simple word embedding models (Shen et al. 2018), which employs simple pooling strategies operated over word embeddings.
- **LEAM**: label-embedding attentive models (Wang et al. 2018), which embeds the words and labels in the same joint space for text classification. It utilizes label descriptions.
- **Graph-CNN-C**: a graph CNN model that operates convolutions over word embedding similarity graphs (Defferrard, Bresson, and Vandergheynst 2016), in which Chebyshev filter is used.
- **Graph-CNN-S**: the same as Graph-CNN-C but using Spline filter (Bruna et al. 2014).
- **Graph-CNN-F**: the same as Graph-CNN-C but using Fourier filter (Henaff, Bruna, and LeCun 2015).

Datasets. We ran our experiments on five widely used benchmark corpora including 20-Newsgroups (20NG), Ohsumed, R52 and R8 of Reuters 21578 and Movie Review (MR).

- The 20NG dataset[1] (bydate version) contains 18,846 documents evenly categorized into 20 different categories. In total, 11,314 documents are in the training set and 7,532 documents are in the test set.
- The Ohsumed corpus[2] is from the MEDLINE database, which is a bibliographic database of important medical literature maintained by the National Library of Medicine. In this work, we used the 13,929 unique Cardiovascular diseases abstracts in the first 20,000 abstracts of the year 1991. Each document in the set has one or more associated categories from the 23 disease categories. As we focus on single-label text classification, the documents belonging to multiple categories are excluded so that 7,400 documents belonging to only one category remain. 3,357 documents are in the training set and 4,043 documents are in the test set.
- R52 and R8[3] (all-terms version) are two subsets of the Reuters 21578 dataset. R8 has 8 categories, and was split to 5,485 training and 2,189 test documents. R52 has 52 categories, and was split to 6,532 training and 2,568 test documents.
- MR is a movie review dataset for binary sentiment classification, in which each review only contains one sentence (Pang and Lee 2005)[4]. The corpus has 5,331 posi-

[1]`http://qwone.com/~jason/20Newsgroups/`

[2]`http://disi.unitn.it/moschitti/corpora.htm`

[3]`https://www.cs.umb.edu/~smimarog/textmining/datasets/`

[4]`http://www.cs.cornell.edu/people/pabo/movie-review-data/`

Table 1: Summary statistics of datasets.

Dataset	# Docs	# Training	# Test	# Words	# Nodes	# Classes	Average Length
20NG	18,846	11,314	7,532	42,757	61,603	20	221.26
R8	7,674	5,485	2,189	7,688	15,362	8	65.72
R52	9,100	6,532	2,568	8,892	17,992	52	69.82
Ohsumed	7,400	3,357	4,043	14,157	21,557	23	135.82
MR	10,662	7,108	3,554	18,764	29,426	2	20.39

tive and 5,331 negative reviews. We used the training/test split in (Tang, Qu, and Mei 2015)[5].

We first preprocessed all the datasets by cleaning and tokenizing text as (Kim 2014). We then removed stop words defined in NLTK[6] and low frequency words appearing less than 5 times for 20NG, R8, R52 and Ohsumed. The only exception is MR, we do not remove words after cleaning and tokenizing raw text, as the documents are very short. The statistics of the preprocessed datasets are summarized in Table 1.

Settings. For Text GCN, we set the embedding size of the first convolution layer as 200 and set the window size as 20. We also experimented with other settings and found that small changes did not change the results much. We tuned other parameters and set the learning rate as 0.02, dropout rate as 0.5, L_2 loss weight as 0. We randomly selected 10% of training set as validation set. Following (Kipf and Welling 2017), we train Text GCN for a maximum of 200 epochs using Adam (Kingma and Ba 2015) and stop training if the validation loss does not decrease for 10 consecutive epochs. For baseline models, we use default parameter settings as in their original papers or implementations. For baseline models which use pre-trained word embeddings, we use 300-dimensional GloVe word embeddings (Pennington, Socher, and Manning 2014)[7].

Test Performance. Table 2 presents test accuracy of each model. Text GCN performs the best and significantly outperforms all baseline models ($p < 0.05$ based on student t-test) on four datasets, which showcases the effectiveness of the proposed method on long text datasets. For more in-depth performance analysis, we note that TF-IDF + LR performs well on long text datasets like 20NG and can outperform CNN with randomly initialized word embeddings. When pre-trained GloVe word embeddings are provided, CNN performs much better, especially on Ohsumed and 20NG. CNN also achieves the best results on short text dataset MR with pre-trained word embeddings, which shows it can model consecutive and short-distance semantics well. Similarly, LSTM-based models also rely on pre-trained word embeddings and tend to perform better when documents are shorter. PV-DBOW achieves comparable results to strong baselines on 20NG and Ohsumed, but the results on shorter text are clearly inferior to others. This is likely due to the fact that word orders are important in short text or sentiment classification. PV-DM performs worse than PV-DBOW, the only comparable results are on MR, where word orders are more essential. The results of PV-DBOW and PV-DM indicate that unsupervised document embeddings are not very discriminative in text classification. PTE and fastText clearly outperform PV-DBOW and PV-DM because they learn document embeddings in a supervised manner so that label information can be utilized to learn more discriminative embeddings. The two recent methods SWEM and LEAM perform quite well, which demonstrates the effectiveness of simple pooling methods and label descriptions/embeddings. Graph-CNN models also show competitive performances. This suggests that building word similarity graph using pre-trained word embeddings can preserve syntactic and semantic relations among words, which can provide additional information in large external text data.

The main reasons why Text GCN works well are two fold: 1) the text graph can capture both document-word relations and global word-word relations; 2) the GCN model, as a special form of Laplacian smoothing, computes the new features of a node as the weighted average of itself and its second order neighbors (Li, Han, and Wu 2018). The label information of document nodes can be passed to their neighboring word nodes (words within the documents), then relayed to other word nodes and document nodes that are neighbor to the first step neighboring word nodes. Word nodes can gather comprehensive document label information and act as bridges or key paths in the graph, so that label information can be propagated to the entire graph. However, we also observed that Text GCN did not outperform CNN and LSTM-based models on MR. This is because GCN ignores word orders that are very useful in sentiment classification, while CNN and LSTM model consecutive word sequences explicitly. Another reason is that the edges in MR text graph are fewer than other text graphs, which limits the message passing among the nodes. There are only few document-word edges because the documents are very short. The number of word-word edges is also limited due to the small number of sliding windows. Nevertheless, CNN and LSTM rely on pre-trained word embeddings from external corpora while Text GCN only uses information in the target input corpus.

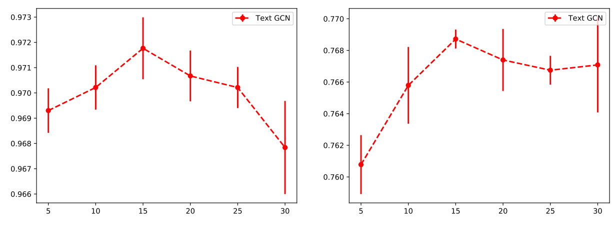
[5]https://github.com/mnqu/PTE/tree/master/data/mr

[6]http://www.nltk.org/

[7]http://nlp.stanford.edu/data/glove.6B.zip

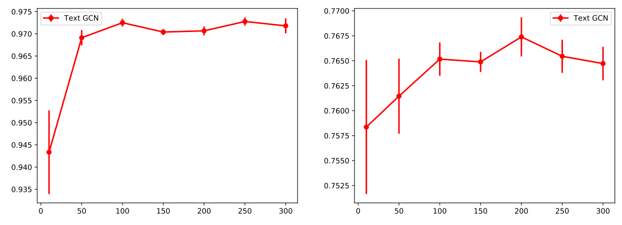
Table 2: Test Accuracy on document classification task. We run all models 10 times and report mean $\pm$ standard deviation. Text GCN significantly outperforms baselines on 20NG, R8, R52 and Ohsumed based on student t-test ($p < 0.05$).

Model	**20NG**	**R8**	**R52**	**Ohsumed**	**MR**
TF-IDF + LR	0.8319 $\pm$ 0.0000	0.9374 $\pm$ 0.0000	0.8695 $\pm$ 0.0000	0.5466 $\pm$ 0.0000	0.7459 $\pm$ 0.0000
CNN-rand	0.7693 $\pm$ 0.0061	0.9402 $\pm$ 0.0057	0.8537 $\pm$ 0.0047	0.4387 $\pm$ 0.0100	0.7498 $\pm$ 0.0070
CNN-non-static	0.8215 $\pm$ 0.0052	0.9571 $\pm$ 0.0052	0.8759 $\pm$ 0.0048	0.5844 $\pm$ 0.0106	**0.7775** $\pm$ **0.0072**
LSTM	0.6571 $\pm$ 0.0152	0.9368 $\pm$ 0.0082	0.8554 $\pm$ 0.0113	0.4113 $\pm$ 0.0117	0.7506 $\pm$ 0.0044
LSTM (pretrain)	0.7543 $\pm$ 0.0172	0.9609 $\pm$ 0.0019	0.9048 $\pm$ 0.0086	0.5110 $\pm$ 0.0150	0.7733 $\pm$ 0.0089
Bi-LSTM	0.7318 $\pm$ 0.0185	0.9631 $\pm$ 0.0033	0.9054 $\pm$ 0.0091	0.4927 $\pm$ 0.0107	0.7768 $\pm$ 0.0086
PV-DBOW	0.7436 $\pm$ 0.0018	0.8587 $\pm$ 0.0010	0.7829 $\pm$ 0.0011	0.4665 $\pm$ 0.0019	0.6109 $\pm$ 0.0010
PV-DM	0.5114 $\pm$ 0.0022	0.5207 $\pm$ 0.0004	0.4492 $\pm$ 0.0005	0.2950 $\pm$ 0.0007	0.5947 $\pm$ 0.0038
PTE	0.7674 $\pm$ 0.0029	0.9669 $\pm$ 0.0013	0.9071 $\pm$ 0.0014	0.5358 $\pm$ 0.0029	0.7023 $\pm$ 0.0036
fastText	0.7938 $\pm$ 0.0030	0.9613 $\pm$ 0.0021	0.9281 $\pm$ 0.0009	0.5770 $\pm$ 0.0049	0.7514 $\pm$ 0.0020
fastText (bigrams)	0.7967 $\pm$ 0.0029	0.9474 $\pm$ 0.0011	0.9099 $\pm$ 0.0005	0.5569 $\pm$ 0.0039	0.7624 $\pm$ 0.0012
SWEM	0.8516 $\pm$ 0.0029	0.9532 $\pm$ 0.0026	0.9294 $\pm$ 0.0024	0.6312 $\pm$ 0.0055	0.7665 $\pm$ 0.0063
LEAM	0.8191 $\pm$ 0.0024	0.9331 $\pm$ 0.0024	0.9184 $\pm$ 0.0023	0.5858 $\pm$ 0.0079	0.7695 $\pm$ 0.0045
Graph-CNN-C	0.8142 $\pm$ 0.0032	0.9699 $\pm$ 0.0012	0.9275 $\pm$ 0.0022	0.6386 $\pm$ 0.0053	0.7722 $\pm$ 0.0027
Graph-CNN-S	–	0.9680 $\pm$ 0.0020	0.9274 $\pm$ 0.0024	0.6282 $\pm$ 0.0037	0.7699 $\pm$ 0.0014
Graph-CNN-F	–	0.9689 $\pm$ 0.0006	0.9320 $\pm$ 0.0004	0.6304 $\pm$ 0.0077	0.7674 $\pm$ 0.0021
Text GCN	**0.8634** $\pm$ **0.0009**	**0.9707** $\pm$ **0.0010**	**0.9356** $\pm$ **0.0018**	**0.6836** $\pm$ **0.0056**	0.7674 $\pm$ 0.0020



(a) R8 (b) MR

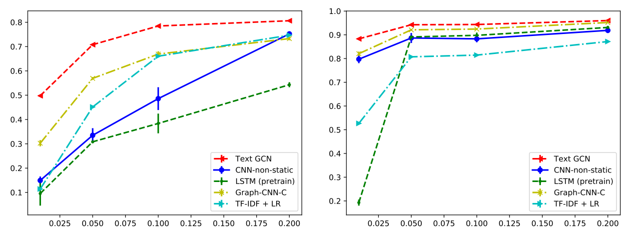
Figure 2: Test accuracy with different sliding window sizes.



(a) R8 (b) MR

Figure 3: Test accuracy with different dimensions of first layer embeddings.

Parameter Sensitivity. Figure 2 shows test accuracies with different sliding window sizes on R8 and MR. We can see that test accuracy first increases as window size becomes larger, but the average accuracy stops increasing when window size is larger than 15. This suggests that too small window sizes could not generate sufficient global word co-

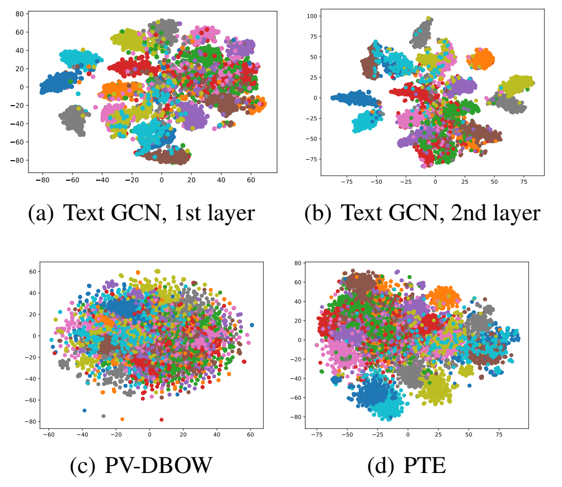


(a) 20NG (b) R8

Figure 4: Test accuracy with different proportions of the training data.

occurrence information, while too large window sizes may add edges between nodes that are not very closely related. Nevertheless, the classification performance is not as sensitive to window size. For example, the difference between 15 and 20 are not statistically significant. Figure 3 depicts the classification performance on R8 and MR with different dimensions of the-first layer embeddings. We observed similar trends as in Figure 2. Too low dimensional embeddings may not propagate label information to the whole graph well, while high dimensional embeddings do not improve classification performances and may cost more training time.

Effects of the Size of Labeled Data. In order to evaluate the effect of the size of the labeled data, we tested several best performing models with different proportions of the training data. Figure 4 reports test accuracies with 1%, 5%, 10% and 20% of original 20NG and R8 training set. We note that Text GCN can achieve higher test accuracy with limited


(a) Text GCN, 1st layer (b) Text GCN, 2nd layer

(c) PV-DBOW (d) PTE

Figure 5: The t-SNE visualization of test set document embeddings in 20NG.

Table 3: Words with highest values for several classes in 20NG. Second layer word embeddings are used. We show top 10 words for each class.

comp.graphics	sci.space	sci.med	rec.autos
jpeg	space	candida	car
graphics	orbit	geb	cars
image	shuttle	disease	v12
gif	launch	patients	callison
3d	moon	yeast	engine
images	prb	msg	toyota
rayshade	spacecraft	vitamin	nissan
polygon	solar	syndrome	v8
pov	mission	infection	mustang
viewer	alaska	gordon	eliot

labeled documents. For instance, Text GCN achieves a test accuracy of 0.8063 ± 0.0025 on 20NG with only 20% training documents and a test accuracy of 0.8830 ± 0.0027 on R8 with only 1% training documents which are higher than some baseline models with even the full training documents. These encouraging results are similar to results in (Kipf and Welling 2017) where GCN can perform quite well with low label rate, which again suggests that GCN can propagate document label information to the entire graph well and our word document graph preserves global word co-occurrence information.

Document Visualization. We give an illustrative visualization of the document embeddings leaned by Text GCN. We use t-SNE tool (Maaten and Hinton 2008) to visualize the learned document embeddings. Figure 5 shows the visualization of 200 dimensional 20NG test document embeddings learned by GCN (first layer), PV-DBOW and PTE. We also show 20 dimensional second layer test document embeddings of Text GCN. We observe that Text GCN can learn more discriminative document embeddings, and the second layer embeddings are more distinguishable than the first layer.

Word Visualization. We also qualitatively visualize word embeddings learned by Text GCN. Figure 6 shows the t-SNE visualization of the second layer word embeddings learned from 20NG. We set the dimension with the highest value as a word's label. We can see that words with the same label are close to each other, which means most words are closely related to some certain document classes. We also show top 10 words with highest values under each class in Table 3. We note that the top 10 words are interpretable. For example, "jpeg", "graphics" and "image" in column 1 can represent the meaning of their label "comp.graphics" well. Words in other columns can also indicate their label's meaning.

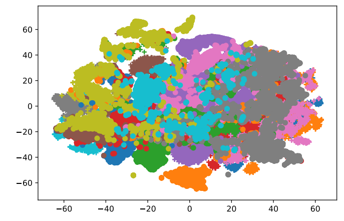


Figure 6: The t-SNE visualization of the second layer word embeddings (20 dimensional) learned from 20NG. We set the dimension with the largest value as a word's label.

Discussion. From experimental results, we can see the proposed Text GCN can achieve strong text classification results and learn predictive document and word embeddings. However, a major limitation of this study is that the GCN model is inherently transductive, in which test document nodes (without labels) are included in GCN training. Thus Text GCN could not quickly generate embeddings and make prediction for unseen test documents. Possible solutions to the problem are introducing inductive (Hamilton, Ying, and Leskovec 2017) or fast GCN model (Chen, Ma, and Xiao 2018).

Conclusion and Future Work

In this study, we propose a novel text classification method termed Text graph convolutional networks (Text-GCN). We build a heterogeneous word document graph for a whole corpus and turn document classification into a node classification problem. Text GCN can capture global word co-occurrence information and utilize limited labeled documents well. A simple two-layer Text GCN demonstrates promising results by outperforming numerous state-of-the-art methods on multiple benchmark datasets.

In addition to generalizing Text GCN model to inductive settings, some interesting future directions include improving the classification performance using attention mechanisms (Veličković et al. 2018) and developing unsupervised text GCN framework for representation learning on large-scale unlabeled text data.

References

[Aggarwal and Zhai 2012] Aggarwal, C. C., and Zhai, C. 2012. A survey of text classification algorithms. In *Mining text data*. Springer. 163–222.

[Bastings et al. 2017] Bastings, J.; Titov, I.; Aziz, W.; Marcheggiani, D.; and Simaan, K. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *EMNLP*, 1957–1967.

[Battaglia et al. 2018] Battaglia, P. W.; Hamrick, J. B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

[Bruna et al. 2014] Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2014. Spectral networks and locally connected networks on graphs. In *ICLR*.

[Cai, Zheng, and Chang 2018] Cai, H.; Zheng, V. W.; and Chang, K. 2018. A comprehensive survey of graph embedding: problems, techniques and applications. *IEEE Transactions on Knowledge and Data Engineering* 30(9):1616–1637.

[Chen, Ma, and Xiao 2018] Chen, J.; Ma, T.; and Xiao, C. 2018. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *ICLR*.

[Chenthamarakshan et al. 2011] Chenthamarakshan, V.; Melville, P.; Sindhwani, V.; and Lawrence, R. D. 2011. Concept labeling: Building text classifiers with minimal supervision. In *IJCAI*.

[Conneau et al. 2017] Conneau, A.; Schwenk, H.; Barrault, L.; and Lecun, Y. 2017. Very deep convolutional networks for text classification. In *EACL*.

[Defferrard, Bresson, and Vandergheynst 2016] Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 3844–3852.

[Hamilton, Ying, and Leskovec 2017] Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NIPS*, 1024–1034.

[Henaff, Bruna, and LeCun 2015] Henaff, M.; Bruna, J.; and LeCun, Y. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*.

[Hochreiter and Schmidhuber 1997] Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

[Joulin et al. 2017] Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2017. Bag of tricks for efficient text classification. In *EACL*, 427–431. Association for Computational Linguistics.

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

[Kingma and Ba 2015] Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

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

[Le and Mikolov 2014] Le, Q., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *ICML*, 1188–1196.

[Li, Han, and Wu 2018] Li, Q.; Han, Z.; and Wu, X. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*.

[Liu, Qiu, and Huang 2016] Liu, P.; Qiu, X.; and Huang, X. 2016. Recurrent neural network for text classification with multi-task learning. In *IJCAI*, 2873–2879. AAAI Press.

[Luo, Uzuner, and Szolovits 2016] Luo, Y.; Uzuner, Ö.; and Szolovits, P. 2016. Bridging semantics and syntax with graph algorithmsstate-of-the-art of extracting biomedical relations. *Briefings in bioinformatics* 18(1):160–178.

[Maaten and Hinton 2008] Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *JMLR* 9(Nov):2579–2605.

[Marcheggiani and Titov 2017] Marcheggiani, D., and Titov, I. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *EMNLP*, 1506–1515.

[Mikolov et al. 2013] Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, 3111–3119.

[Pang and Lee 2005] Pang, B., and Lee, L. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, 115–124.

[Peng et al. 2018] 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 *WWW*, 1063–1072. International World Wide Web Conferences Steering Committee.

[Pennington, Socher, and Manning 2014] Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *EMNLP*, 1532–1543.

[Rousseau, Kiagias, and Vazirgiannis 2015] Rousseau, F.; Kiagias, E.; and Vazirgiannis, M. 2015. Text categorization as a graph classification problem. In *ACL*, volume 1, 1702–1712.

[Shen et al. 2018] Shen, D.; Wang, G.; Wang, W.; Renqiang Min, M.; 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. In *ACL*.

[Skianis, Rousseau, and Vazirgiannis 2016] Skianis, K.; Rousseau, F.; and Vazirgiannis, M. 2016. Regularizing text categorization with clusters of words. In *EMNLP*, 1827–1837.

[Tai, Socher, and Manning 2015] Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, volume 1, 1556–1566.

[Tang, Qu, and Mei 2015] Tang, J.; Qu, M.; and Mei, Q.

2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *KDD*, 1165–1174. ACM.

[Veličković et al. 2018] Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph attention networks. In *ICLR*.

[Wang and Manning 2012] Wang, S., and Manning, C. D. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*, 90–94. Association for Computational Linguistics.

[Wang et al. 2016] Wang, Y.; Huang, M.; Zhao, L.; et al. 2016. Attention-based lstm for aspect-level sentiment classification. In *EMNLP*, 606–615.

[Wang et al. 2018] Wang, G.; Li, C.; Wang, W.; Zhang, Y.; Shen, D.; Zhang, X.; Henao, R.; and Carin, L. 2018. Joint embedding of words and labels for text classification. In *ACL*, 2321–2331. Association for Computational Linguistics.

[Yang et al. 2016] Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; and Hovy, E. 2016. Hierarchical attention networks for document classification. In *NAACL*, 1480–1489.

[Zeng et al. 2018] Zeng, Z.; Deng, Y.; Li, X.; Naumann, T.; and Luo, Y. 2018. Natural language processing for ehr-based computational phenotyping. *IEEE/ACM transactions on computational biology and bioinformatics* 10.1109/TCBB.2018.2849968.

[Zhang, Liu, and Song 2018] Zhang, Y.; Liu, Q.; and Song, L. 2018. Sentence-state lstm for text representation. In *ACL*, 317–327.

[Zhang, Zhao, and LeCun 2015] Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. In *NIPS*, 649–657.