InducT-GCN: Inductive Graph Convolutional Networks for Text Classification

Anonymous ACL submission

Abstract

Text classification aims to assign labels to textual units by making use of global information. Recent studies have applied graph neural network (GNN) techniques to capture the global word co-occurrence in a corpus. Most existing approaches require that all the nodes (training and test) in a graph are present during training, which are transductive and do not naturally generalise to unseen nodes. To make those models *inductive*, previous works use extra resources, like pretrained word embedding. However, high-quality resource is not always available and can be hard to train. Under the extreme settings with no extra resource and limited amount of training set, can we still learn an inductive graph-based text classification model? In this paper, we introduce a novel inductive graph-based text classification framework, namely InducT-GCN (**I**nduc**T**ive **G**raph **C**onvolutional **N**etworks for **T**ext classification). Compared to transductive models that require test documents in training, we construct a graph based on the statistics of training documents only and represent document vectors with a weighted sum of word vectors. We then conduct one-directional GCN propagation during testing. Across five text classification benchmarks, our InducT-GCN outperformed state-of-the-art methods that are either transductive in nature or pre-trained additional resources. We also conducted scalability testing by gradually increasing the data size and revealed that our InducT-GCN can reduce the time and space complexity.

1 Introduction

Text classification is one of the most fundamental tasks in the natural language processing research. Assume that we are given a description $d \in \mathcal{X}$ of a document, where $\mathcal{X}$ is the document space; and a fixed set of classes $\mathcal{C} = \{c_1, c_2, ..., c_J\}$, can be called as categories or labels. A document space $\mathcal{X}$ is typically some type of high-dimensional space and the classes are human defined for the needs of an application. Note that we only consider a single-class text classification problem in this research.

With the rise of deep learning, many text classification studies have focused on learning text representations using sequence-based learning models, such as convolutional neural networks (CNN) (Kim, 2014) or recurrent neural networks (RNN) /long short term memory (LSTM) (Zhou et al., 2016). The CNN/RNN-based models focus on the locality and sequence of text, and mainly aim to detect semantic and syntactic information in local consecutive word sequences. It tends to neglect global word co-occurrence in a corpus and ignore non-consecutive and long distance semantic information (Peng et al., 2018). However, those models need relatively large size of training set in order to achieve better performance but most real-world cases (e.g. specific domain or some low resource languages) have very limited amount of training set (limited labeled data). Recently, pre-trained models, like BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), have achieved state-of-the-art performance on several NLP tasks with limited amount of training data. However, those models requires much computation and external resources for pre-training which are not always available.

Yao et al. (2019) proposed TextGCN and achieves state-of-the-art result especially when the percentage of training data is low without using any external resources and low computation costs. It is an initial text graph-based neural network framework, which conducts a straightforward manner of graph construction and applies a GCN-learning(Kipf and Welling, 2017) to deal with complex structured textual data and prioritise global feature exploitation. More recent studies (Wu et al., 2019; Liu et al., 2020; Zhu and Koniusz, 2021) focused on utilising more contextual information or optimising the computation or performance.

However, most graph-based learning mod-

els is intrinsically transductive, meaning that the learned node representations/embeddings for words/documents are not naturally generalisable to unseen words/documents, making it difficult to apply in real world. The transductive nature of these graph-based learning models require relatively large computational space when the corpus size is large. Therefore, an inductive model is required. In order to extend a transdutive graph-based text classification into an inductive model, we mainly consider the following three requirements: 1)The inductive learning model must not include any test set information during the training. 2)The inductive model must not re-train the model on the whole new graph when it learns a new sample. 3)We use corpus-level graph-based text classification in order to make inductive model since it well covers the benefit of graph-based learning, which captures complex global structure of the whole corpus and prioritises global feature exploitation. With these three requirements, in this paper, we propose a novel inductive graph-based text classification framework, called **InducT-GCN** (**InducT**ive **G**raph **C**onvolutional **N**etworks for **T**ext classification). We introduce a new inductive graph framework of graph construction, learning, and testing, and it can expand any tranductive GCN-based text classification model. The paper includes the following contributions:

- To the best of our knowledge, we introduce the first inductive corpus-level GCN-based text classification framework without using any extra resources.
- We compare our InducT-GCN on five benchmark datasets under the limited labeled data settings. InducT-GCN achieves the highest accuracy on four of them, even beating some trasductive baselines integrated by using external resources.
- We introduce a new way to make transductive GCN-based text classification models *inductive*, which improve the performance and reduce the time and space complexity.

2 Related Work

2.1 Graph Neural Networks

Graph Neural Network (GNN)s (Cai et al., 2018; Kipf and Welling, 2017) have been effective at tasks to have rich relational structure and can preserve global structure information of a graph in graph embeddings by aggregating first-order neighborhood information. Kipf and Welling (2017) introduced Graph Convolutional Networks (GCN) on semi-supervised and transductive classification tasks using generalized convolutional layers. Then, GraphSage (Hamilton et al., 2017) and FastGCN (Chen et al., 2018) tailored GCNs on inductive representation learning framework with sampling methods. The Attention mechanism was also applied to GCN, called Graph Attention Networks (GAT) (Veličković et al., 2018), in order to specify different weights to different nodes in a neighbourhood. More recent GCN studies for transductive and inductive frameworks have been proposed. For transductive-based GCN, SGC (Wu et al., 2019) was introduced to reduce the complexity and S^2GC (Zhu and Koniusz, 2021) was proposed to solve over-smoothing problems. Some inductive-based models, DeepGL (Rossi et al., 2020) and TGAT (da Xu et al., 2020), were introduced to cover different graph tasks, including transfer learning and topology learning.

2.2 Text Classification Using GNN

GNNs have received lots of attention in various NLP tasks (Bastings et al., 2017; Marcheggiani and Titov, 2017; Tu et al., 2019; Li et al., 2019; Yao et al., 2019; Cao et al., 2019; Yang et al., 2021), including text classification. The GNN-based text classification models can be categorised into two types: Document-level and Corpus-level approaches.

Document-level GNN in Text Classification Several studies proposed a graph-based text classification by building a graph for each document using words as nodes (Defferrard et al., 2016; Peng et al., 2018; Zhang et al., 2018; Nikolentzos et al., 2020; Huang et al., 2019; Zhang et al., 2020). Word nodes are represented by external resources, pre-trained embedding, such as Word2vec (Mikolov et al., 2013), and Glove (Pennington et al., 2014). Hence, they do not consider any global structure information of a corpus/entire dataset during their model training/learning.

Corpus-level GNN in Text Classification Compared to the above Document-level models, Yao et al. (2019) proposed TextGCN, a graph-based text classification that builds a graph for the entire text corpus with documents and words as nodes. Hence, it captures global information of an entire corpus and conduct node(document) classifica-

tion. After that, SGC (Wu et al., 2019) and S^2GC (Zhu and Koniusz, 2021) constructed a graph as TextGCN, but proposed different information propagation approaches. Liu et al. (2020) introduced TensorGCN by explicitly integrating three different aspect-based graphs, including syntactic, semantic, and sequential, for better text classification performance. Li et al. (2021) also proposed the three graphs for covering semantics, syntantic, and contextual aspects. Note that all three graphs are based on an entire corpus and use the same propagation as GCN (Kipf and Welling, 2017). TG-Transformer(Zhang and Zhang, 2020) applied transformer with pretrained GloVe embeddings to the TextGCN, and BERTGCN(Lin et al., 2021) applied BERT embedding to the TextGCN. All the above models are transductive-based approaches as GCN (Kipf and Welling, 2017). However, our model InducT-GCN, an inductive graph-based text classification framework, constructs a corpus-level graph but adopts the nature of inductive learning in order to naturally generalise to unseen nodes[1].

3 Tranductive and Inductive Nature

In this section, we aim to discuss the nature of transductive and inductive GCN learning for text classification, and what inductive learning aspect we would like to explore for text classification. Most GCN models for text classification, including TextGCN(Yao et al., 2019), SGC(Wu et al., 2019), or S^2GC (Zhu and Koniusz, 2021), are inherently transductive, and applied in the whole corpus-level fixed graph. Note that those models use a whole-corpus based textual graph, to create nodes (incl. document, word nodes) and edges (incl. word-word: PMI[2], word-doc: TF-IDF).

In order to extend those transductive models into an inductive learning nature, we fundamentally improve two aspects as follows. First, the transductive GCN-based text classification models include documents not only from the training set but also the test set when constructing a whole-corpus based textual graph for GCN learning. Hence, the learned GCN model will be influenced/generalised by word/document information in the test set, which is supposed to be unseen nodes. Our inductive GCN-based text classification model constructs a graph with only training document information but does not consider any information from the test sets. We just focus on generalising to unseen nodes, and aligning newly observed subgraphs to the node that the model has already optimised on.

Secondly, the transductive models learn the embedding for V_{train}, V_{test}, V_{word} simultaneously by using one-hot input vectors $H^{(0)} \in R^{n \times n}$. For any new test sample, the embedding for that sample should be re-learned by re-training the model on the new graph. In this case, the re-learning/re-training process does not perfectly fulfil the effective generalisation to unseen nodes. Therefore, we come up with new graph construction and training/testing solution for inductive learning, instead of re-learning or re-training.

4 InducT-GCN

We propose an Inductive Graph Convolutional Network (GCN) for text classification, named 'InducT-GCN', which can be an extension of the traditional transductive GCN-based text classification models. We adopt the traditional transductive GCN-based text classification models, including TextGCN(Yao et al., 2019) and SGC(Wu et al., 2019), and focus on expanding those models to efficient inductive learning models. This section demonstrates the proposed inductive learning components[3] applied to TextGCN.

4.1 Revisit TextGCN

TextGCN is a GCN-based text classification model that uses a large and heterogeneous text graph based on the whole corpus. To understand the concept properly, we first explore the GCN process.

Graph Convolutional Networks(GCN) Formally, considering a graph $G = (V, E, A)$, where $V(|V| = n)$ is a set of nodes, E is a set of edges, and $A \in R^{n \times n}$ is an adjacent matrix representing the edge values between nodes. The propagation rule of each hidden layer is:

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

where $\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is a normalized symmetric matrix for A and $D_{ii} = \Sigma_j A_{ij}$ as a degree matrix of adjacent matrix A. H^l is l_{th} hidden layer input and W^l is a weight to be learned in this layer. σ is an activation function, e.g. ReLU: $\sigma(x) = max(0, x)$.

[1]The nature of inductive learning is detailed in Section 3

[2]Point-wise Mutual Information(Yao et al., 2019)

[3]We also applied our proposed inductive learning components to SGC in Section 6.2

TextGCN Followed by the GCN(Kipf and Welling, 2017), TextGCN constructs a large corpus-level graph but with textual information, document and word as nodes so it can be modelled the global word-document co-occurrence. Like GCN, the constructed graph includes document and word nodes from training sets, as well as test sets. TextGCN aims to model the global word-document occurrence with two major edges: 1) word-word edge: calculated by point-wise mutual information(PMI), 2) document-word edge: TF-IDF. Note that they use one-hot vectors for word and document nodes. One-hot vectors are fed into a two-layer GCN model to jointly learn the embedding for the documents and words during the training phase. The representations learned from the document nodes in the training set are used for training the text(document) classification model, and those on the document nodes in the test set are used for predicting the document class.

4.2 InducT-GCN Graph Construction

4.2.1 Graph Nodes

As mentioned earlier, our inductive GCN-based text classification model, InducT-GCN, strictly do not consider any information or statistics from the test set, which is supposed to be unseen nodes. Instead, we construct the nodes only the with training document information. Consider a set of nodes $V = \{V_{train}, V_{word}\}$ and the V_{word} are the unique words in the training documents. In order to define input vector $H^{(0)}$ for graph nodes in the InducT-GCN graph, we consider two requirements:

- During the graph propagation, all the input vectors for document nodes and word nodes are multiplied by the same weights so that these vectors should align with each other.
- Unlike TextGCN (transductive model), our InducT-GCN must not use one-hot vectors for representing document nodes. As mentioned, we do not consider or deal with any information of testing documents during the training phase. Hence, the model will neither learn any representation nor make a prediction on testing documents if we directly use the one-hot vector for document node embedding as TextGCN proposed.

With this in mind, we propose a new document representation by focusing on the nature of our proposed inductive learning idea. For the proper alignment between word and document representation, InducT-GCN generates document node representations based on its word nodes vectors. We use a weighted average of word vectors to construct document nodes vectors, and the key idea of this construction is applying TF-IDF weights. Formally, one-hot vectors are used for representing word nodes vectors $H_w^{(0)}, \forall w \in V_{word}$. For representing training documents node vectors $H_i^{(0)}, \forall i \in V_{train}$, we use TF-IDF vectors. The values for each dimension is TF-IDF values for the corresponding word in that specific document:

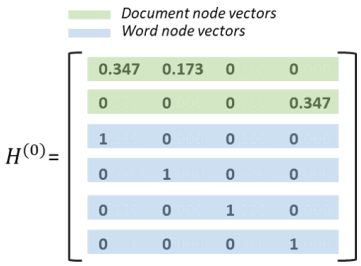


Figure 1: Input Vectors Representations (document and word node vectors) when two input documents are "w_1 w_1 w_2 w_3" and "w_3 w_4".

$$H_{ij}^{(0)} = \text{TF-IDF}(i, j) \tag{2}$$

where i and j are document and word, respectively.

Assume $H^{(0)} \in R^{n \times |V_{word}|}$, where $n = |V_{train}| + |V_{word}|$. When two input documents are "w_1 w_1 w_2 w_3" and "w_3 w_4". Figure 1 visualise the input vector representation, where $H_0^{(0)}$ and $H_1^{(0)}$ are document vectors and $H_i^{(0)}, \forall i \in [2, 5]$ are word vectors.

4.2.2 Graph Edges

We focus on extending corpus-level transductive GCN-based text classification to inductive learning, and select TextGCN(Yao et al., 2019) as one of its kind. Like TextGCN, we define two-edge types for InducT-GCN graph: 1) Word-Word with PMI and 2) Word-Doc edges with TF-IDF. Note that each node also connects to itself. PMI is calculated based on the co-occurrence of a pair of words in a slicing window[4]. Formally, it is calculated by:

$$\text{PMI}(i, j) = log \frac{p(i, j)}{p(i)p(j)} \tag{3}$$

where $p(i, j)$ represents the co-occurrence probability of word i and j and estimated by $p(i, j) =$

[4]The window size is 20, followed by TextGCN

$\frac{\#Co-occurrence}{\#Windows}$, $p(i)$ represents the probability of word i and estimated by $p(i,j) = \frac{\#Occurrence}{\#Windows}$. The graph is un-directed and all the edges are symmetrical. The Adjacent Matrix A is calculated as:

$$A_{ij} = \begin{cases} max(0, \text{PMI}(i,j)) & i, j \text{ are words} \\ \text{TF-IDF}_{i,j} & i \text{ is word}, j \text{ is doc} \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

4.3 InducT-GCN Learning and Testing

After building the graph, we train it using a two-layer Graph Convolutional Network as in (Kipf and Welling, 2017). The first GCN layer learns the word embeddings. The dimension of the second GCN layer is the number of classes of the dataset, and the output is fed into a softmax activation function. The node representations on the training documents are used for cross-entropy loss calculation and back-propagation. Formally, the propagation can be described as:

$$H^{(1)} = \sigma(\tilde{A}H^{(0)}W^{(0)}) \quad (5)$$

$$Z = softmax(\tilde{A}H^{(1)}W^{(1)}) \quad (6)$$

where $W^{(0)}$ is a learned word embedding lookup table. Loss is calculated by using cross-entropy between Z_i and Y_i, $\forall i \in V_{train}$.

In GCN, the propagation for each layer is conducted by updating the nodes with the weighted sum of their first-order neighbours and the node itself. In order to make predictions on the test set, the first-order and second-order neighbours' representation for each test document should be aggregated. Note that we utilise the test documents during testing phase only so there is no need to update all the nodes in the graph during the propagation.

Instead, we conduct an one-direction propagation and only update test document node $t, \forall t \in V_{test}$. Firstly, $H_i^{(1)}, \forall i \in V_{word}$, $W^{(0)}$ and $W^{(1)}$ are recorded after training phase. Storing $H_i^{(1)}$ enables InducT-GCN not to work on the training document nodes during the test phase so it saves computation resources. During the testing phase, InducT-GCN supports batch testing(Hamilton et al., 2017). For each batch of test document node $B \in V_{test}, |B| = b$, the edges E_B between B and V_{word} are calculated using TF-IDF with the document frequency of the training set.

Test document input $H_B^{\prime(0)}$ is also calculated using TF-IDF. The testing phase can be described as:

$$A_B = \text{concat}(E_B, I) \quad (7)$$

$$H_{word,B}^{\prime(0)} = \text{concat}(H_{word}^{(0)}, H_B^{\prime(0)}) \quad (8)$$

$$H_B^{\prime(1)} = \sigma(A_B H_{word,B}^{\prime(0)} W^{(0)}) \quad (9)$$

$$H_{word,B}^{\prime(1)} = \text{concat}(H_{word}^{(1)}, H_B^{\prime(1)}) \quad (10)$$

$$Z_B = \text{softmax}(A_B H_{word,B}^{\prime(1)} W^{(1)}) \quad (11)$$

where $A_B \in R^{b \times (|V_{word}|+b)}$ stands for the weights of the weighted sum calculation, and $H^{\prime(0)} \in R^{(|V_{word}|+b) \times |V_{word}|}$ stands for the input vectors in the subgraph and will be used to update the test document node using word nodes information. $H_B^{\prime(1)} \in R^{b \times h}$ is the updated test document node embedding after the first layer of GCN and h is the hidden dimension size. Then, the stored $\{H_w^{(1)}, \forall w \in V_{word}\}$, represented as $H_{word}^{(1)}$, are used to propagate training documents information to the test document nodes for the second layer. [5]

4.4 Generalization

In summary, we successfully replace the input vectors with InducT-GCN way of node input vectors and use one-directional propagation on a subgraph during the test phase. With our Inductive graph construction and learning framework, it is possible to extend any corpus-level and tranductive GCN-based text classification models, such as TextGCN(Yao et al., 2019), SGC(Wu et al., 2019), TensorGCN(Liu et al., 2020), and S^2GC (Zhu and Koniusz, 2021). Section 6.2 shows the detailed generalization evaluation results.

4.5 Space and Time Analysis

Comparing with TextGCN, InducT-GCN is more efficient both in time and space. For the space complexity:(1)Number of Parameters of InducT-GCN is $|V_{word}| * h + h * c$ while TextGCN requires $(|V_{train}| + |V_{word}| + |V_{test}|) * h + h * c$ parameters. Meanwhile, $|V_{word}|$ in InducT-GCN is smaller than that in TextGCN. (2)Graph Space complexity of InducT-GCN is $O(|V_{word}|^2 + |V_{train}| * |V_{word}|)$ and for TextGCN, it is $O(|V_{word}|^2 + (|V_{train}| + |V_{test}|) * |V_{word}|)$. Similarly, $|V_{word}|$ in InducT-GCN is smaller than TextGCN.

Comparing with TextGCN, our model is faster in three ways: (1)When constructing the graph, the time complexity of PMI is $O(|V_{word}|^2 *$

[5] We formally describe the overall algorithms for training and testing phase of InducT-GCN in Appendix A.

Dataset	# Full Train	# Limited Train	# Test	# Word	# Node	# Class	Avg Len
R8	5,485	274	2,189	1,878	2,152	8	62.22
R52	6,532	326	2,568	2,568	2,894	52	66.98
Ohsumed	3,357	167	4,043	2,667	2,834	23	123.7
20NG	11,314	113	7,532	2,839	2,952	20	163.5
MR	7,108	355	3,554	605	960	2	7.25

Table 1: Summary statistics of datasets. The limited environment description can be found in the Section 5.1

$\#Windows$), and it is smaller for InducT-GCN. (2)The training time is shorter for InducT-GCN since the graph is smaller. (3)When new test samples show, TextGCN requires retraining while InducT-GCN can make predictions without retraining. The testing result can be found in Section B.

5 Evaluation Setup

We evaluate the performance of our InducT-GCN on text classification, and examine the effectiveness of the proposed inductive learning approach.

5.1 Dataset[6]

We first evaluate InducT-GCN on 5(five) publicly available and widely used text classification benchmark datasets, including R8, R52, Ohsumed, 20NG, and MR. In order to test in the limited environment (small size training set), we select 5% of the full training set (1% for 20NG, due to the size). Note that the size of the test set is the same as the original. The detailed statistics of datasets can be found in Table 1. Secondly, we also extend the evaluation on the larger test sets. We apply the data augmentation methods on R8 dataset, called R8A. The detailed procedure can be found as follows.

R8, **R52** are two subsets of the Reuters dataset, and focus on the topic classification with 8 and 52 classes respectively. **Ohsumed** is produced by the MEDLINE database, which is a bibliographic database of life sciences and biomedical information. It focuses on the medical text classification using 23 classes for different cardiovascular diseases abstracts. **20NG**(20 NewsGroup) is a 20 class-based news classification dataset. **MR**(Movie Review) is a binary (positive and negative) sentiment polarity analysis, in which each movie review only contains one sentence. **R8A**: To evaluate our InducT-GCN scalability in the larger test set, we apply a data augmentation technique. Nlpaug[7], the python library for data augmentation in NLP, is applied for augmenting the R8 test set. To achieve this, we randomly choose one of the four options: (1) randomly deleting a word, (2) adding a word based on Word2Vec embedding similarity, (3) substituting a word using Synonyms in WordNet (Miller, 1995), (4) randomly swapping two words. With the above procedure, we evaluate our InducT-GCN with 1-5 times (2,189, 4,378, 6,567, 8,756, 10,945) of the R8 original test set size. The detailed evaluation can be found in Section 6.3.

All dataset are preprocessed and tokenised based on Kim (2014). We remove the words if shown less than 2 times[8] in the training documents or if listed in NLTK stop word list.

5.2 Baselines

We compare InducT-GCN with baselines, mainly those models with no external knowledge and learning in an inductive way. However, due to the limited number of baselines, we include the baselines with pretrained word embeddings, such as CNN-non-static, LSTM (pretrained), and TextING. We also add transductive models, including TextGCN and SGC.

- **TF-IDF + SVM/LR** applies TF-IDF vectors for the input representation, and uses a traditional machine learning models, Support Vector Machine (SVM) or Logistic Regression (LR), as classifiers respectively.
- **CNN** Kim (2014) applies Convolutional Neural Network on text classification. We evaluate two types: 1) CNN-rand with randomly initialised word embeddings and 2) CNN-non-static with pretrained GloVe(Pennington et al., 2014) embeddings separately.
- **LSTM** Liu et al. (2016) applies Long Short-Term Memory, and uses last hidden state for the text representation. We use 1) randomly initialised word embedding or 2) pretrained Glove(Pennington et al., 2014).

[6]The datasets will be shared via the github for reproducibility.

[7]https://github.com/makcedward/nlpaug

[8]Words only shown once can not work as a bridge between two document nodes.

Method	PT.	R8 5%	R52 5%	Ohsumed 5%	20NG 1%	MR 5%
TF-IDF + SVM	✗	0.8054 ± 0.0000	0.6830 ± 0.0000	0.1476 ± 0.0000	0.1289 ± 0.0000	0.5537 ± 0.0000
TFIDF + LR	✗	0.8090 ± 0.0000	0.6869± 0.0000	0.1813 ± 0.0000	0.1860 ± 0.0000	0.5967 ± 0.0000
CNN-rand	✗	0.8107 ± 0.0110	0.6854 ± 0.0100	0.1586 ± 0.0079	0.1390 ± 0.0179	0.5485 ± 0.0122
CNN-non-static	✓	0.9052 ± 0.0097	0.7708 ± 0.0181	0.3411 ± 0.0370	0.2969 ± 0.0277	**0.7009 ± 0.0060**
LSTM	✗	0.7392 ± 0.0146	0.6364 ± 0.0060	0.1614 ± 0.0085	0.0766 ± 0.0063	0.5301 ± 0.0191
LSTM (Pretrain)	✓	0.7916 ± 0.0499	0.6667 ± 0.0303	0.2486 ± 0.0392	0.1010 ± 0.0220	0.6680 ± 0.0198
TextGCN (Yao et al., 2019)	✗	0.9116 ± 0.0127	0.7885 ± 0.0751	0.2225 ± 0.1138	0.2198 ± 0.1293	0.5341 ± 0.0216
SGC (Wu et al., 2019)	✗	0.8955 ± 0.0098	0.7725 ± 0.0189	0.2474 ± 0.0392	0.2948 ± 0.0342	0.6015 ± 0.0051
TextING (Zhang et al., 2020)	✓	0.8648 ± 0.0414	0.7465 ± 0.0298	0.3026 ± 0.0235	N/A	0.6117 ± 0.0342
InducT-GCN	✗	**0.9155 ± 0.0051**	**0.8135 ± 0.0384**	**0.3563 ± 0.0078**	**0.3461 ± 0.0337**	0.6037 ± 0.0038

Table 2: Comparison with Baseline on Limited Labeled Data. For 20NG dataset, TextING(Zhang et al., 2020) has out of Memory issue so the TextING authors also have not tested on the 20NG either. *The column PT. refers to the model applied any pre-trained embedding.

- **TextGCN/SGC** are the graph-based transductive learning models. Yao et al. (2019) introduced TextGCN, a graph-based transductive and corpus-level text classification model. Inspired by TextGCN, Wu et al. (2019) propose SGC (Simplifying Graph Convolutional networks) using the same graph construction and improves the performance.

- **TextING**, proposed by Zhang et al. (2020), and a document-level GNN for text classification. It uses pretrained GloVE embedding and applies graph embedding to classify the documents in an inductive way.

5.3 Settings

For a fair comparison, we apply the same set of hyper-parameters to all dataset without hyper-parameter tuning. As described in Yao et al. (2019), we applied two layers graph convolutional model to all graph-based baseline models, including TextGCN (Yao et al., 2019), SGC (Wu et al., 2019), our InducT-GCN. Whereas the embedding size in the first GCN layer is 200, that in the second layer is the number of classes of each dataset. The dropout rate for each layer is 0.5. Adam optimizer with a learning rate of 0.02 is used for training, and no weight decay is applied. For each experiment, followed by the original GCN paper (Kipf and Welling, 2017), 200 epochs are set to be the maximum number of epochs, and an early stopping mechanism is applied. 10% of the training set is randomly selected as the validation set. The model would stop training when the minimum validation loss in the recent 10 epochs is larger than the minimum validation loss 10 epochs ago, which indicates the loss does not decrease for 10 consecutive epochs. For other baseline models, early stopping mechanism is also applied by using the default hyper-parameters. For all experiments, the models are trained using 16 Intel(R) Core(TM) i9-9900X CPU @ 3.50GHz and NVIDIA Titan RTX 24GB on Ubuntu 20.04.1 with Pytorch 1.7. Followed by Yao et al. (2019); Wu et al. (2019); Liu et al. (2020); Zhang et al. (2020), we use the accuracy as an evaluation metric. For each testing result, we produce the average and standard derivation of the ten-time running results.

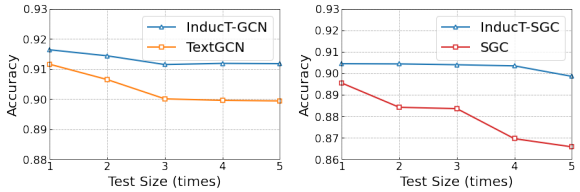
6 Result

6.1 Performance Evaluation

A comprehensive experiment is conducted on the 5 benchmark datasets in the limited environment as mentioned in Section 5.1. The result presented in Table 2 shows that our proposed InducT-GCN significantly outperforms all baselines (including transductive graph-based models, such as TextGCN and SGC, and CNN, LSTM, TextING use external knowledge, pretrained word embeddings) in terms of average accuracy on four datasets in R8, R52, Ohsumed, 20NG. Meanwhile, the standard derivation is smaller than most of baseline models, showing the robustness of our model. For more in-depth performance analysis, we can highlight that this result shows the effectiveness of the proposed InducT-GCN on long text datasets. While the average lengths of 20NG and Ohsumed are longer than 100 and those of R8 and R52 are still longer than 60, MR is less than 10 which can be considered as an extremely short text dataset. We found that models with pre-trained word embeddings GloVe, achieve better performance on the short text documents. This is mainly because it would be very difficult to recognise the global word co-occurrence with this short length of text documents, which leads to a fewer connection (bridging) between word nodes in

Method	R8 5%	R52 5%	Ohsumed 5%	20NG 1%	MR 5%
SGC	0.8955 ± 0.0098	0.7725 ± 0.0189	0.2474 ± 0.0392	0.2948 ± 0.0342	0.6015 ± 0.0051
InducT-SGC	**0.9045 ± 0.0046**	**0.8046 ± 0.0066**	**0.3106 ± 0.0061**	**0.2990 ± 0.0251**	**0.6017 ± 0.0048**

Table 3: Transductive Graph Text Classification Model Test



(a) TextGCN (b) SGC

Figure 2: Test accuracy with different test size on R8A by using both TextGCN and SGC with our proposed Inductive model.

corpus-level text graph. Nevertheless, our InducT-GCN performs the best among the models, which do not have any pre-trained embeddings[9].

6.2 Generalization Evaluation

In this section, we report the generalisation capability of our inductive graph construction and learning framework. As mentioned earlier, our framework is applied to TextGCN, the most widely known transductive graph-based text classification model. We now extend our inductive framework to another corpus-level graph-based model, SGC (Wu et al., 2019), and called InducT-SGC. Table 3 visualises the comparison of the original transductive SGC models and our InducT-SGC. As can be seen in the table, our InducT-SGC produces much higher performance than the original SGC when the labeled data are limited. The performance improvement between both pair of the original transductive and our inductive model, TextGCN-to-InducT-GCN and SGC-to-InductSGC, clearly shows the generalisation capability of our proposed inductive framework. It can be also applied to other corpus-level graph-based text classification models in the future.

6.3 Impact of Test Size

As mentioned earlier, we use the R8A (R8 with a data augmentation) to show the scalability of our proposed Inductive learning framework by comparing TextGCN and InducT-GCN in the larger text set. Figure 2a shows the comparison of TextGCN and InducT-GCN on different test sizes, with 1 to 5 times (2,189, 4,378, 6,567, 8,756, 10,945) of the R8 original test set size. The larger the test size, the larger the gap between TextGCN and InducT-GCN. TextGCN produces worse performance with the largest test size. This is mainly because there would be only a small proportion of the document nodes contributing to the gradient in TextGCN with a larger test set. Especially during the training phase, it is difficult for TextGCN to learn embeddings of those test document nodes having fewer connections with word nodes by backpropagation. Moreover, we found that the performance of our InducT-GCN decreased only a little (less than 0.5) when the test size grows. We also conducted the exact same evaluation based on SGC by applying our Inductive graph construction and learning framework to SGC, named InducT-SGC. Like the result that our InducT-GCN produced, the InducT-SGC produces much higher performance than the original SGC. The performance trend shows how our Inductive framework fits perfectly on the inductive learning nature.

7 Conclusion

In this study, we propose a novel inductive graph-based text classification framework, named InducT-GCN, which makes heavy and transductive GCN-based text classification models inductive. We construct a graph only using training set statistics. InducT-GCN can efficiently capture global information with fewer parameters and smaller space complexity. Our InducT-GCN significantly outperformed all graph-based text classification baselines, and even better than the models using pre-trained embeddings. We also demonstrated the generalisation capability of our inductive graph construction and learning framework by applying and expanding different transductive graph-based text classification models, like TextGCN and SGC. The performance and computation time surprisingly improved, compared to the original models. It is hoped that this paper provides some insight into the future integration of the lighter and faster inductive graph neural networks on different NLP tasks.

[9] Apart from our main goal, we also conducted an evaluation on R8 and R52 in the full environment for testing the superiority of our Inductive learning framework. The result can be found in the appendix C

References

Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967.

Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637.

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

Jie Chen, Tengfei Ma, and Cao Xiao. 2018. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*.

da Xu, chuanwei ruan, evren korpeoglu, sushant kumar, and kannan achan. 2020. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations (ICLR)*.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*.

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

William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035.

Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and WANG Houfeng. 2019. Text level graph neural network for text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3435–3441.

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

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

Chen Li, Xutan Peng, Hao Peng, Jianxin Li, and Lihong Wang. 2021. Textgtl: Graph-based transductive learning for semi-supervised text classification via structure-sensitive interpolation. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2680–2686.

Y Li, R Jin, and Y Luo. 2019. Classifying relations in clinical narratives using segment graph convolutional and recurrent neural networks (seg-gcrns). *Journal of the American Medical Informatics Association: JAMIA*, 26(3):262–268.

Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li, and Fei Wu. 2021. Bertgcn: Transductive text classification by combining gcn and bert. *arXiv preprint arXiv:2105.05727*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2873–2879. IJCAI/AAAI Press.

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Giannis Nikolentzos, Antoine Tixier, and Michalis Vazirgiannis. 2020. Message passing attention networks for document understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8544–8551.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 world wide web conference*, pages 1063–1072.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Ryan A Rossi, Rong Zhou, and Nesreen K Ahmed. 2020. Deep inductive graph representation learning. *IEEE Computer Architecture Letters*, 32(03):438–452.

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

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations*.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR.

Tianchi Yang, Linmei Hu, Chuan Shi, Houye Ji, Xiaoli Li, and Liqiang Nie. 2021. Hgat: Heterogeneous graph attention networks for semi-supervised short text classification. *ACM Transactions on Information Systems*, 39(3).

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377.

Haopeng Zhang and Jiawei Zhang. 2020. Text graph transformer for document classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8322–8327.

Yue Zhang, Qi Liu, and Linfeng Song. 2018. Sentence-state lstm for text representation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 317–327.

Yufeng Zhang, Xueli Yu, Zeyu Cui, Shu Wu, Zhongzhen Wen, and Liang Wang. 2020. Every document owns its structure: Inductive text classification via graph neural networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 334–339.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3485–3495.

Hao Zhu and Piotr Koniusz. 2021. Simple spectral graph convolution. In *International Conference on Learning Representations*.

A Learning and Inferencing Algorithm

Alorightm 1 shows the training phase and testing phase of InducT-GCN.

Algorithm 1 InducT-GCN Training and Testing Phase

Input: Training Graph $G(V, \tilde{A}), V = \{V_{train}, V_{word}\}$;Test Adjacent Matrix $\{A'_{B,w}, B \in V_{test}, \forall w \in V_{word}\}$; Training input vectors $\{H_i^{(0)}, \forall i \in V_{train}\}$; Test input vectors $\{H_B'^{(0)}, B \in V_{test}\}$; Training Labels $\{Y_i, \forall i \in V_{train}\}$

Parameter: Weight matrices $W^{(0)}$ and $W^{(1)}$

Output: Prediction Results for Test Samples $\{Y_B, B \in V_{test}\}$

for $epoch = 1, 2, \ldots$ **do**
  $H^{(1)} \leftarrow \sigma(\tilde{A}H^{(0)}W^{(0)})$
  $Z \leftarrow softmax(\tilde{A}H^{(1)}W^{(1)})$
  $L \leftarrow \text{CrossEntropy}(Y_i, Z_i), \forall i \in V_{train}$
  Update $W^{(0)}$ and $W^{(1)}$
end for
for Batch B in V_{test} **do**
  $G' \leftarrow \{A'_{B,w}, H_w^{(0)}, H_B'^{(0)}\}, \forall w \in V_{word}$
  $H_B'^{(1)} \leftarrow \text{GCN}(G', v_B)$
  $G'' \leftarrow \{A'_{B,w}, H_w^{(1)}, H_B'^{(1)}\}, \forall w \in V_{word}$
  $Y_B \leftarrow argmax(\text{GCN}(G'', v_B))$
end for

B Computation Time Results

Table 6 visualises the superiority of our inductive learning framework by reducing the computation time, including graph construction and training/testing. It compared the original transductive TextGCN with our model on R8A; The larger the gap between TextGCN and our InducT-GCN, the larger the test size.

C Performance in Full Dataset

We also evaluated the performance of our InducT-GCN with the full dataset, like the TextGCN was evaluated (Yao et al., 2019). As can be seen in Table 7, the performance of InducT-GCN and TextGCN on R8 and R52 are comparable when using the entire dataset. We can conclude that

# Parameters	R8 5%	R52 5%	Ohsumed 5%	20NG 1%	MR 5%
1st Layer	375,600	513,600	533,400	567,800	121,000
2nd Layer	1,600	10,400	4,600	4,000	400
Total	377,200	524,000	538,000	571,800	121,400

Table 4: Number of Parameters

Embedding	R8 5%	R52 5%	Ohsumed 5%	20NG 1%	MR 5%
RAND	0.9155 ± 0.0051	0.8135 ± 0.0384	**0.3563 ± 0.0078**	0.3461 ± 0.0337	0.6037 ± 0.0038
Word2Vec	0.9124 ± 0.0043	**0.8290 ± 0.0084**	0.3544 ± 0.0305	0.3476 ± 0.0086	0.6003 ± 0.0045
GloVe	**0.9159 ± 0.0102**	0.8266 ± 0.0090	0.3514 ± 0.0186	**0.3662 ± 0.0197**	**0.6051 ± 0.0055**

Table 5: Pretrained Embedding

Test Size	TextGCN		InducT-GCN	
	Graph	Training	Graph	Training
×1	6.29	2.75	0.89	0.52
×2	11.90	3.20	1.11	0.53
×3	16.60	3.54	1.30	0.53
×4	21.10	4.13	1.52	0.55
×5	27.50	4.98	1.68	0.51

Table 6: Graph Construction Time and Training Time comparison on R8A (sec)

InducT-GCN has superior than the TextGCN in terms of the performance and computation, which is not only in smaller space with fewer parameter numbers, but also in the whole dataset setting.

Method	R8 Full	R52 Full
TextGCN	0.9629 ± 0.0010	0.9295 ± 0.0020
InducT-GCN	**0.9653 ± 0.0017**	**0.9323 ± 0.0015**

Table 7: Comparison with TextGCN on Full Data

D Number of Parameters

The number of parameters of the first GCN layer is $|V_{word}| * h$, which is the number of unique words times first layer hidden dimension. In the second layer, $h * |\mathcal{C}|$ parameters are trained, and $|\mathcal{C}|$ is the number of classes for each dataset. In total, the number of parameters is $(|V_{word}| + |\mathcal{C}|) * h$, and details are shown in Table 4.

E Pretrained Embeddings

(Draft) The first layer of InducT-GCN actullay learns the word embeddings, and in InducT-GCN the weights were randomly initialized. We also tested with initializing with pretrained word embeddings, shown in Tabel 5. In most of the cases, the performance improves, showing the potential of InducT-GCN when combined with external resources.