Inductive Light Graph Convolution Network for Text Classification Based on Word-Label Graph

Jinze Shi, Xiaoming Wu, Xiangzhi Liu, Wenpeng Lu, Shu Li

Inductive Light Graph Convolution Network for Text Classification based on Word-Label Graph

Shi Jinze[1], Wu Xiaoming[1*], Liu Xiangzhi[1], Lu Wenpeng[2] and Li Shu[1]

[1*]Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan, Shandong, China.
[2]Qilu University of Technology (Shandong Academy of Sciences), Jinan,Shandong, China.

*Corresponding author(s). E-mail(s): wuxm@sdas.org;
Contributing authors: 626418756@qq.com; liuxzh@sdas.org;
lwp@qlu.edu.cn; lishu04211@163.com;

Abstract

Nowadays, Graph Convolution Networks (GCNs) have flourished in the field of text classification, such as Text Graph Convolution Network (TextGCN). But good performance of those methods is based on building a graph whose nodes consist of an entire corpus, making their models transductive. Meanwhile rich label information has not been utilized in the graph structure. In this paper, we propose a new model named Inductive Light Graph Convolution Networks (ILGCN) with a new construction of graph. This approach uses labels and words to build the graph which removes the dependence between an individual text and entire corpus, and let ILGCN inductive. Besides, we simplify the model structure and only remain the neighborhood aggregation, which is the most important part of GCNs. Experiments on multiple benchmarkshow that our model outperforms existing state-of-the-art models on several text classification datasets.

Keywords: Text classification, graph convolution networks, word-label graph, inductive light graph convolution network

1 Introduction

Text classification is one of the classic problems of Natural Language Processing (NLP), and there are many applications including but not limited to SPAM detection(1), Computational phenotyping(2), and so on. The most critical intermediate step is the text representation learning. Traditional methods of text representation are mostly hand-made, such as Term Frequency - Inverse Document Frequency (TF-IDF)(3), bag-of-words(4) and n-grams(5). With the development of deep learning methods, models such as convolutional neural networks (CNN)(6) and recurrent neural networks (RNN) (7; 8) have been applied to the text representation learning and achieved remarkable results. In recent years, GNN has been gradually explored by researchers due to its effectiveness in capturing global and associated data structure(9; 10; 11), and has been applied in the field of text classification(12; 13; 14; 15) and achieved state-of-the-art results in several text classification datasets. However, many GNN-based models are transductive. They use an entire corpus to construct a graph, including unlabeled texts that need to be predicted and causing the following practical problems:

1. Transductive model is not conducive to large-scale text classification and time-demanding application. Because the model should be retrained again at each time a new text coming in, due to using the whole corpus as nodes.
2. It's not conducive to online test, since the graph structure and model parameters depend on the complete corpus including all the texts whether its labeled or not, and it cannot be modified after training.
3. Rich label information is not utilized in the graph structure, but only used as targets of training.

To solve the above drawbacks, we proposed a new graph neural network for text classification in this paper, called Inductive LightGCN and a new way to construct graph using labels and nodes. Instead of building the whole corpus as a graph, we use words of all texts and corresponding labels to build a graph, making the model inductive and utilizing information carried by labels. Meanwhile, this graph significantly reduces the memory requirements of the graph. Inspired by LightGCN(16), we simplified the original GCN model structure by eliminating nonlinear functions and weight matrices within the layers. Through the empirical assessment on three text classification benchmark datasets, our ILGCN demonstrated comparable results to state-of-the-art models: TextGCN(13), Simplifying Graph Convolutional Networks (SGC)(14) and Simple Spectral Graph Convolution (SSGC)(15), even achieved the SOTA results in two datasets.

To summarize, this work makes the following main contributions:

- We prove that the feature transformation and nonlinear activation impose negative effects on TextGCN.
- We propose a inductive model, ILGCN, which is largely simplified and based on a new construction of graph.

- Our model achieves state-of-the-art results in several text classification datasets and rationality of ILGCN is shown by means of in-depth analyses.

2 Preliminaries

We will introduce TextGCN. Then the ablation experiments, which prove the problems of nonlinear activation and feature transformation in TextGCN, are performed, which will prove the rationality of combining LightGCN's ideas into ours model.

Table 1: Performance of TextGCN and its variants.

	Accuracy	
	Ohsumed	**R52**
TextGCN	0.685	0.940
TextGCN-n	0.686	0.945
TextGCN-f	0.680	0.934
TextGCN-fn	0.680	0.934
TextGCN-fn-l2	0.685	0.940

2.1 TextGCN Brief

In the first step, TextGCN(13) simply sets feature matrix X = I as an identity matrix, initialize the input variable to a simple identity matrix. Using the Point-wise Mutual Information (PMI)which consisted of word occurrence in texts, as text-word edges and word co-occurrence in the whole corpus as the edge weights between words. Using TF-IDF of the words in the texts as the weight between the words and the texts. The definition of elements in the adjacency matrix:

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

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

$$\text{PMI}(i,j) = log\frac{p(i,j)}{p(i)p(j)} \tag{2}$$

$$p(i,j) = \frac{\#W(i,j)}{\#W} \tag{3}$$

$$p(i) = \frac{\#W(i)}{\#W} \tag{4}$$

where $\#W(i)$ is the number of sliding windows in which word i appears; $\#W(i,j)$ is the number of pair i,j appear in the same window; and $\#W$ is the total number of sliding windows in the whole corpus. TextGCN uses a simple two layers GCN model and is defined as:

$$Y = \text{softmax}(\tilde{A}\text{ReLU}(\tilde{A}XW_0)W_1) \tag{5}$$

where $\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the normalized symmetric adjacency matrix; $W_0 \in R^{(n+m)\times d_0}$ and $W_1 \in R^{d_1 \times d_0}$ respectively denote the trainable feature matrix of two layers; $\text{ReLU}(\cdot)$ is the nonlinear activation; and $\text{softmax}(\cdot)$ is the classifier function. The model structure of TextGCN completely follows GCN(9), including nonlinear activation function and feature transformation matrix.

2.2 Empirical Explorations on TextGCN

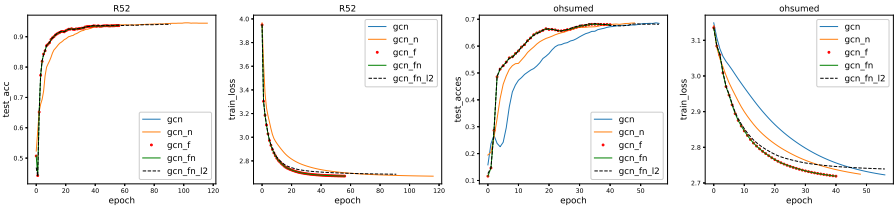


Fig. 1: Training curves (training loss and testing accuracy) of TextGCN and its four simplified variants.

We conduct ablation studies on TextGCN to test the effectiveness of nonlinear activation and feature transformation. By using data and evaluation protocol same as TextGCN. We implement four variants of TextGCN:

- TextGCN-n, which removes the nonlinear activation.
- TextGCN-f, which removes the first layer's feature transformation W_0
- TextGCN-fn, which removes W_0 and he nonlinear activation.
- TextGCN-fn-l2, which removes the nonlinear activation and W_0, but L_2 regularization for W_0 is also added.

Results are shown in Table 1. As can be seen that when nonlinear is removed(i.e.,TextGCN-n), the performance on two different datasets is improved. When the feature transformation of the first layer is removed(i.e.,TextGCN-f, TextGCN-fn), the performance decreased compared with the original model on the two datasets, but better than the model with L_2 regularization(i.e.,TextGCN-fn-l2). Based on these observations, we can draw the following conclusions:

1. Adding nonlinear activation in TextGCN have negative effects, because TextGCN-n has improved the effect in two corpora.
2. Removing W_0 in TextGCN reduces the effect compared with TextGCN. In this case, whether nonlinear activation was removed or not has no significant effect on the results.
3. After removing the W_0, and adding the regularization of the second layer W_1, effect of TextGCN-fn has been improved, indicating that the reason of the lower performance of TextGCN-fn is the overfitting of W_1 to the training set, so removing feature transformation is feasible.

We plot the changes of training loss and testing accuracy as Figure 1 for deep insights. And the empirical experiments show that TextGCN's nonlinear activation and feature transformation will increase the difficulty of training, and it is necessary to suppress the overfitting of feature transformation to the training set by dropout or L_2 regularization, otherwise the performance will become worse. Based on the above viewpoints, we propose the ILGCN.

3 Method

In this section we will describe our method in detail. First, describing how to construct a text-label graph for a whole corpus. Then the ILGCN model structure is showed.

3.1 Building graph

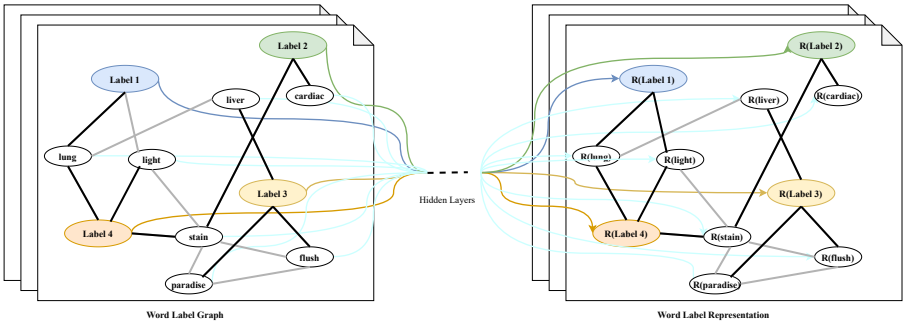


Fig. 2: Schematic layout of graph convolution part of ILGCN. Including 4 label nodes and some word nodes. Black bold edges are word-label edges and gray edges are word-word edges. $R(x)$ means the representation(embedding) of x. Different labels are filled with different colors.

We normally define an undirected graph as $G = (V, A)$, nodes $W = w_1, ..., w_m$ and label nodes $L = l_1, ..., l_n$, which together constitute the node set $V = v_1, ..., v_{m+n}$. Each node v_i corresponds to a d-dimensional feature vector $x_i \in R^d$ and $X \in R^{(m+n)} \times d$, which represents feature matrix of all nodes.

We set $X = I$, which is the same as TextGCN. And $A \in R^{(n+m)\times(n+m)}$ is the symmetric adjacency matrix where a_{ij} represents the weight of edges between node v_i and v_j. We set $a_{ij} = 0$ if there is no edge between v_i and v_j; edge weight between two word nodes w_i and w_j is calculated by Point-wise Mutual Information (PMI); the edge weight between a word node w_i and a label node l_j is obtained by TF-IDF of the word in the label, in which term frequency is the number of times the word appears in the label, inverse label frequency is logarithmically scaled inverse fraction of the number of labels that contain the word. Finally, the weight between node i and node j is defined as:

$$A_{ij} = \begin{cases} \text{PMI}(i,j) & \text{i,j are words, PMI}(i,j) > 0 \\ \text{TF-IDF}_{ij} & \text{i is label, j is word} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

The PMI value of a word pair i, j is the same as (2), (3), (4). The difference of A between ILGCN and TextGCN is that if we don't set $a_{ij} = 0$ not 1 when $i = j$, because with I as X, adding a node self-connection is equal to the weighted sum of the embeddings propagated at each layer of ILGCN. This is similar to LightGCN.

Compared with the previous methods for constructing the graph, our method can effectively reduce the number of nodes in the graph, especially when the size of the corpus is very large. This means that the word-label graph will consume less memory. Moreover, because the graph does not use the texts, it is more friendly to the new test data. Because the graph is only related to the words contained in the datasets, the model becomes a decoupled model.

3.2 ILGCN

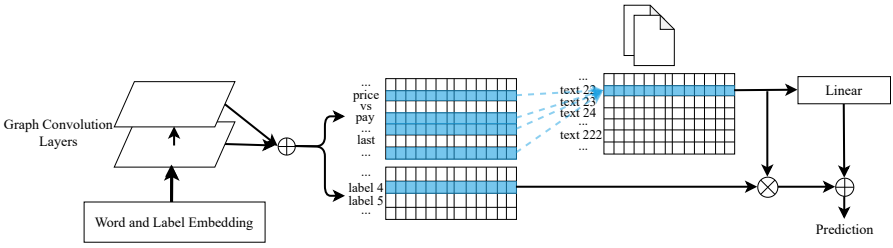


Fig. 3: Overall layout of ILGCN. The black bold arrows are representations of information transfer. The blue dashed lines represent the constructing process of text embeddings.

After the construction of the graph is completed, the adjacency matrix A and the feature matrix X are taken as the input of the ILGCN model. In our proposed model, feature transformations W_0 and W_1 in the graph convolution part of TextGCN are abandoned as well as the activation function, and

we only remain the adjacency matrix and trainable embedding matrix. The convolutional layer of ILGCN is defined as:

$$Z_i = \tilde{A} Z_{i-1} \tag{7}$$

where Z_i represents the embedding of the nodes trained at layer i. $\tilde{A}$ is defined in the same way as formula (5). $\tilde{A}$ restricts the growth of embedding and gathers information from different neighbor nodes to the central node with different weights. It should be noted that the $\tilde{A}$ here does not be added self-connection, so we adopt the weighted sum which plays the same role. Through weighted sum, ILGCN fuses the embedding representation extracted by graph convolution structure at each layer to represent the embedding of the trained words and the labels in the end. The formula is as follows:

$$Z = \sum_{l=0}^{L} \alpha_i Z_i \tag{8}$$

where Z_i represents the embedding matrix of the i-th layer,X is used as the initialization feature matrix of Z_0; α_i represents the weighting coefficient; L represents the total number of layers of the graph convolution model. Since different convolution layers extract different information and can reduce oversmoothing, we decided to use layer combination to get the final representations.

We can get text embeddings through multiplying trained word embeddings with $D \in R^{t \times (n+m)}$ matrix without combine the texts in the graph. D represents text vector of unclassified texts; t represents the number of unclassified texts. The value of text i and word j in the matrix D is calculated by the ratio of number of word j in the text i and total number of words in the text i. we need to note that the pair of label and text is set as 0 and get the text embeddings as:

$$Z_{text} = DZ \tag{9}$$

Finally, the prediction part of the model is composed of two parts. One is to map the text embedding of $n+m$ dimension to the number of classes C through a single-layer perceptron. The other part is to get the correlation between text and labels by multiplying the text embedding and label embedding. The final model prediction consists of two parts through weighted sum as:

$$Y = \beta Z_{text} W + (1 - \beta) Z_{text} Z_{labels}^{T} \tag{10}$$

where β represents the trainable weight; $W \in R^{(n+m) \times C}$ is a parameter matrix, which plays the role of mapping a high-dimensional vector to a low-dimensional vector; $Z_{labels}^{T} \in R^{C \times (n+m)}$ represents the trained transpose of label embedding part of Z. The goal of training is to minimize the cross entropy

loss between ground truths label and predicted labels:

$$Loss = -\sum_{d \in Y_D} \sum_{c=1}^{C} Y_{dc} (\beta \ln Z_{dc}^{mlp} + (1-\beta) \ln Z_{dc}^{label}) \tag{11}$$

where Y_D is the set of text indices that have labels; C is the dimension of the output features, which is equal to the number of classes; Y is the label indicator matrix; $Z^{mlp} = Z_{text}W$ and Z_{labels} represents the prediction results of the two parts that make up the prediction results respectively. The overall ILGCN model is schematically illustrated in Figure 3.

4 Experiment

4.1 Setting

4.1.1 Datasets

Table 2: Statistics of the datasets.

Corpus	Texts	Train	Test	Word	Nodes	Classes	Average Length
R8	7674	5485	2189	7688	7696	8	65.72
R52	9100	6532	2568	8892	8944	52	69.82
Ohsumed	7400	3357	4043	14157	14180	23	135.82
MR	10662	5331	5331	18764	18766	2	20.39

For experiemnts, we utilize four widely used datasets including Ohsumed, R52 and R8 of Reuters 21578 and Movie Review (MR).

- R8 and R52[1] are two subsets of Reuters 21578 database. The R8 dataset has 8 different classifications and contains 7674 texts, including 5485 training texts and 2189 test texts. R52 has 52 different classifications, with a total of 9100 texts, which are divided into 6532 training texts and 2568 test texts. The data distributions of these two data sets are uneven.
- Ohsumed[2] is obtained from Medline Database, containing 7400 texts in 23 classifications, which are split into 3357 training texts and 4043 test texts.
- MR[3] is a binary corpus of film reviews, with a total of 10,662 film reviews as text data, divided equally into 5,331 text data in each of the two categories as what Tang, Qu, and Mei do.

In order to ensure the consistency of the experimental results, we used the same preprocess method as what TextGCN do for the four datasets. We first

[1] https://www.cs.umb.edu/smimarog/textmining/datasets/
[2] http://disi.unitn.it/moschitti/corpora.htm
[3] https://github.com/mnqu/PTE/tree/master/data/mr

preprocessed all the datasets by cleaning and tokenizing texts as (6). We then removed stop words defined in NLTK6 and low frequency words appearing less than 5 times for R8, R52 and Ohsumed.

4.1.2 Baselines

For baselines, we compared our proposed approach with the following three state-of-the-art models: TextGCN, SGC and SSGC. And the results of these baseline models are obtained directly from TextGCN(13), SGC(14), SSGC(15).

- TextGCN. Proposed by Yao et al.2019, a graph-based text classification model, performing graph convolutions on a single large graph for whole corpus.
- SGC. Defined by Wu et al.2019, a graph-based model, reducing the excess complexity through successively removing nonlinearities and collapsing weight matrices between consecutive layers.
- SSGC. Proposed by Zhu et al.2021, a variant of GCN which uses a modified Markov Diffusion Kernel.

4.1.3 Hyperparameter details

The layer combination coefficient $_i$ is uniformly set as $1/(L+1)$ where K is number of layers. After testing K from 1 to 5, we choose K equal to 2 for 2-layer ILGCN and 4 for 3-layer ILGCN. is initialized to 0.5 and can not be trained. Dropout ratio is set as 0.5 and Dropout layer is applied before the dense layer, because feature transformation is not used in the graph convolution layer, so it is not necessary to add dropout to control its overfitting. We used the Adam optimizer (18) and set the learning rate to 0.002 for MR and 0.02 for others, and randomly selected 10% of the training set as validation. As designed in TextGCN, we stop training if the validation loss value is less than the average value of the past ten epochs or reaches 200 epochs.

4.2 Performance Comparison with Different Layers

Table 3: The comparison of accuracy between ILGCNs at different layers.

Layer	R8	R52	Ohsumed	MR
1 Layers	96.7 ± 0.2	94.1 ± 0.2	63.2 ± 0.8	73.1 ± 3.5
2 Layers	97.5 ± 0.1	94.1 ± 0.1	66.1 ± 0.6	77.4 ± 0.1
3 Layers	97.5 ± 0.1	94.4 ± 0.1	66.3 ± 0.2	77.3 ± 0.3
4 Layers	97.4 ± 0.1	94.3 ± 0.2	——	——

We study the influence of models with different layers and record the accuracy of ILGCN on all four datasets, so as to study the performance of ILGCN

against oversmoothing. Due to hardware limitations, the results of the 4 layers model are only tested and compared on two datasets–R8 and R52.The experimental results are shown in Table 3.

We can see that the results of the 1-layer ILGCN are generally bad, with 1 to 3% lower accuracy compared to 2-layer ILGCN. Main reason is only using one layer ILGCN to aggregate node information is not enough. The word could not gather other words' information through the label connected directly, and label information is difficult to get information for two-hops through the adjacent words. Through the results of the three layers and four layers ILGCN, we can clearly see that increasing the number of layers does not have a significant impact on the results, indicating that ILGCN has a very strong ability to resist oversmoothing. But through the four-layer ILGCN, we can see a slight decrease of accuracy in the results. The intuitive explanation is that the two-layer model is enough to aggregate all the required information, while more layers would learn a lot of redundant and useless knowledge, which leads to a decrease in the accuracy of the model.

4.3 Experimental Results

Table 4: The comparison of results between ILGCN and the state-of-the-art models.

Model	R8	R52	Ohsumed	MR
TextGCN	97.0 ± 0.2	93.8 ± 0.2	68.2 ± 0.4	76.3 ± 0.3
SGC	97.2 ± 0.1	94.0 ± 0.2	**68.5 ± 0.3**	75.9 ± 0.3
SSGC	97.4 ± 0.1	**94.5 ± 0.2**	**68.5 ± 0.1**	76.7 ± 0.0
ILGCN	**97.5 + 0.1**	94.4 ± 0.1	66.3 ± 0.2	**77.4 ± 0.1**

Table 4 shows the experimental results of using identity matrix as input for our model compared with other models. We can see that our model outperforms SGC, TextGCN and SSGC on multiple datasets, which is showing the competitiveness of ILGCN in text corpora. It can be seen from Table 4, compared with other models, our model performs better in R8, R52 and MR datasets, but has some shortcomings in ohsumed. We believe that main reason our model can perform better than TextGCN in some datasets is that TextGCN contains some unnecessary linear or nonlinear transformations, which increase the training complexity of the model. Therefore, our model achieves better performance than TextGCN by removing some nonlinear activation and feature transformations.The key difference between our model and SGC or SSGC is the use of word-label graph which helps us to obtain additional information and accomplishes more accurate predictions. By analyzing the accuracy and F1-score of the model in the different categories of the

ohsumed dataset, we believe that the main reason for the poor performance of ILGCN in the ohsumed data set is that there is considerable overlap between the words contained in the different categories.

4.4 Memory Consumption

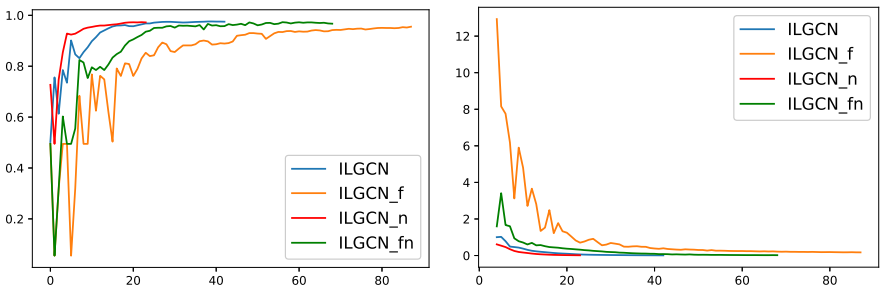
As shown in the Table 5, our model has a significant advantage over TextGCN in memory consumption.

As discussed in 3.1, we do not use texts as our nodes which constructing our graph, instead use the label. And in general, the number of labels is much smaller than the number of the texts. So comparing with TextGCN, which including all texts as its nodes, ours not only reduces a lot of memory consumption, but also solves the problems of transductive of using the entire corpus. The more texts in the corpus, the better our method can reflect the optimization in memory consumption.

Table 5: Comparison of memory consuming.

Datasets	R8	R52	Ohsumed	MR
TextGCN	1800M	2470M	3545M	6606M
Ours	452M	610M	1534M	2687M

4.5 Ablation Study



(a) Testing accuracy on R8

(b) Training loss on R8

Fig. 4: Training curves(training loss and testing accuracy) of ILGCN and its three variants.

We perform ablation studies on R8 dataset to further analyze the effectiveness of removing feature transformation and nonlinear activation in ILGCN. Table 6 shows the results and Figure 4 shows the training curves of ILGCN

Table 6: Performance of ILGCN and its variants on R8.

Mode	ILGCN	ILGCN-f	ILGCN-n	ILGCN-fn
Accuracy	97.5 + 0.1	96.2 ± 0.4	97.2 ± 0.1	96.8 ± 0.2

and its three variants.Since loss values fluctuates sharply in some models at the beginning, we set learning rate of ILGCN-f as 0.07 and others as 0.02, then keep other hyper-parameters same. Figure 4 shows the results after the fourth epoch.

In ILGCN-f, we add a feature transformation matrix at second layer. As shown in Table 6, ILGCN-f performs much worse than ILGCN, which demonstrates the effectiveness of removing it. In our opinion, the main reason is that feature transformation matrix increases the difficulty of training process.

In ILGCN-n, we add a nonlinear activation function at first layer which imposes negative effect compared with ILGCN. Adding nonlinear activation, such as ReLU, make embedding matrix unnecessarily sparser and imposes negative effect to ILGCN though training speed is much faster because of simpler embedding matrix.

In ILGCN-fn, we add both nonlinear activation and feature transformation at first layer and second layer separately. It's performance is better than ILGCN-f but worse than ILGCN-n.We believe nonlinear activation decreases the training difficulty which still higher than ILGCN-n due to feature transformation.

Therefore, it's effective to remove feature transformation and nonlinear activation.

5 Related Work

5.1 GNN

Bruna et al. (19) firstly proposed graph convolutional network on the frequency domain, which can apply the model to graphs. Then, M. Henaff et al. (20) and M. Defferrard et al. (21) further refined the model. The former extends model to large-scale, high-dimensional tasks and simplifies the formulas and the latter designs a local filter to extract local features. Finally, with the help of Kipf(9) and Welling, the most famous model of graph convolution network, is finally proposed. And its simple intuitive idea has inspired many spatial GCNs, including Neural Networks for Graph (NN4G)(22) with sum as the clustering idea; Diffusion-Convolutional Neural Networks (DCNN)(23), DCRNN(24) and Graph Sample and Aggregate (GraphSAGE)(11) with mean as the clustering idea, and Mixture Model Networks (MoNet) (25); Graph Attention Network (GAT) (10) and Graph Isomorphism Network (GIN) (26) use weighted sum as the clustering idea. The first industrial recommendation system based on GCN(9); Chen et.al. (27) further studies the oversmoothing problem of GCN; Ma et al.(28) and Wang et al.(29) propose Disentangled Graph Collaborative

Filtering (DGCF) and Neural Graph Collaborative Filtering (NGCF) that let multiple dimensions partitioned artificially into a few parts that are as orthogonal as possible to extract information.

5.2 Text Classification

Text classification is a classic problem in NLP. Traditional machine learning methods solve text classification relying on feature extraction, such as TF-IDF (Term Frequency-Inverse Document Frequency) (3), N-gram(5), Bag-of-Words(5) and Topic Model(30). Then combines extracted features with machine learning methods such as support vector machine(31), Naive Bayes(32), K-NearestNeighbor (KNN)(33) and other methods to complete classification. With the development of deep learning, Yoon Kim proposes TextCNN(6) which solves NLP problem using CNN firstly; then Long Short-Term Memory Network (LSTM) model (7; 8) has been designed and applied to NLP issues; more recently, Transformer(34) and Bidirectional Encoder Representations from Transformers (BERT)(35), have further enhanced the effectiveness of text classification. With the development of GNN, some graph-based classification models are gradually emerging(10; 11; 36). Yao et al. (13) proposes Text-GCN and achieved state-of-the-art results on several mainstream datasets. Subsequently, the SGC(14), which removes the nonlinear activation and simplify the feature transformation matrix, is proposed to take text classification as a downstream task and achieved good results; SSGC(15), a variant of GCN which uses a modified Markov Diffusion Kernel, also achieves the state-of-the-art results on several text classification datasets.

6 Conclusion and Future Work

In this paper, we proved the redundant part of TextGCN and propose a new text classification model named ILGCN, which simplifies the graph structure and only remain the most important part–neighbor aggregation– to get a better performance. We construct a word-label graph which makes model inductive and let model can leverage neglected information of labels. Experiments show that our model is competitive in many mainstream datasets with many state-of-the-art models.

The future work includes but is not limited to creating other models based on word-label graph, analyzing what kind of structured dataset is suitable for constructing word-label graph to solve problems, and how to make more reasonable use of label information to improve the effect of the model.

References

[1] Jindal, Nitin, and Bing Liu. "Review spam detection." Proceedings of the 16th international conference on World Wide Web. 2007.

[2] Zeng, Zexian, et al. "Natural language processing for EHR-based computational phenotyping." IEEE/ACM transactions on computational biology and bioinformatics 16.1 (2018): 139-153.
[3] Joachims, Thorsten. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.
[4] Zhang, Yin, Rong Jin, and Zhi-Hua Zhou. "Understanding bag-of-words model: a statistical framework." International Journal of Machine Learning and Cybernetics 1.1-4 (2010): 43-52.
[5] Wang, Sida I., and Christopher D. Manning. "Baselines and bigrams: Simple, good sentiment and topic classification." Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 2012.
[6] Kim Y . Convolutional Neural Networks for Sentence Classification[J]. Eprint Arxiv, 2014.x
[7] Kalchbrenner, Nal, Edward Grefenstette, and Phil Blunsom. "A convolutional neural network for modelling sentences." arXiv preprint arXiv:1404.2188 (2014).
[8] Gehring, Jonas, et al. "Convolutional sequence to sequence learning." International Conference on Machine Learning. PMLR, 2017.
[9] Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016).
[10] Veličković, Petar, et al. "Graph attention networks." arXiv preprint arXiv:1710.10903 (2017).
[11] Hamilton, William L., Rex Ying, and Jure Leskovec. "Inductive representation learning on large graphs." arXiv preprint arXiv:1706.02216 (2017).
[12] Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering." arXiv preprint arXiv:1606.09375 (2016).
[13] Yao, Liang, Chengsheng Mao, and Yuan Luo. "Graph convolutional networks for text classification." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. No. 01. 2019.
[14] Wu, Felix, et al. "Simplifying graph convolutional networks." International conference on machine learning. PMLR, 2019.
[15] Zhu H, Koniusz P. Simple spectral graph convolution[C]//International Conference on Learning Representations. 2021.
[16] He, Xiangnan, et al. "Lightgcn: Simplifying and powering graph convolution network for recommendation." Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020.
[17] Wang, Xiang, et al. "Neural graph collaborative filtering." Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval. 2019.

[18] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).
[19] Bruna, Joan, et al. "Spectral networks and locally connected networks on graphs." arXiv preprint arXiv:1312.6203 (2013).
[20] Henaff, Mikael, Joan Bruna, and Yann LeCun. "Deep convolutional networks on graph-structured data." arXiv preprint arXiv:1506.05163 (2015).x
[21] Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering." arXiv preprint arXiv:1606.09375 (2016).
[22] Micheli, Alessio. "Neural network for graphs: A contextual constructive approach." IEEE Transactions on Neural Networks 20.3 (2009): 498-511.
[23] Atwood, James, and Don Towsley. "Diffusion-convolutional neural networks." Advances in neural information processing systems. 2016.
[24] Li, Yaguang, et al. "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting." arXiv preprint arXiv:1707.01926 (2017).
[25] Monti, Federico, et al. "Geometric deep learning on graphs and manifolds using mixture model cnns." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
[26] Xu, Keyulu, et al. "How powerful are graph neural networks?." arXiv preprint arXiv:1810.00826 (2018).
[27] Chen, Deli, et al. "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. No. 04. 2020.
[28] Ma, Jianxin, et al. "Disentangled graph convolutional networks." International Conference on Machine Learning. PMLR, 2019.
[29] Wang, Xiang, et al. "Disentangled Graph Collaborative Filtering." Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020.
[30] Wallach, Hanna M. "Topic modeling: beyond bag-of-words." Proceedings of the 23rd international conference on Machine learning. 2006.
[31] Burges, Christopher JC. "A tutorial on support vector machines for pattern recognition." Data mining and knowledge discovery 2.2 (1998): 121-167.
[32] Leung, K. Ming. "Naive bayesian classifier." Polytechnic University Department of Computer Science/Finance and Risk Engineering 2007 (2007): 123-156.
[33] Cover, Thomas, and Peter Hart. "Nearest neighbor pattern classification." IEEE transactions on information theory 13.1 (1967): 21-27.
[34] Vaswani, Ashish, et al. "Attention is all you need." arXiv preprint arXiv:1706.03762 (2017).
[35] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
[36] Peng, Hao, et al. "Large-scale hierarchical text classification with recursively regularized deep graph-cnn." Proceedings of the 2018 world wide web conference. 2018.