Every Document Owns Its Structure: Inductive Text Classification via Graph Neural Networks

Yufeng Zhang[1*], **Xueli Yu**[1*], **Zeyu Cui**[1], **Shu Wu**[1], **Zhongzhen Wen**[2] **and Liang Wang**[1]
[1]Institute of Automation, Chinese Academy of Sciences
[2]Xi'an Jiaotong University
{yufeng.zhang,xueli.yu}@cripac.ia.ac.cn
{zeyu.cui,shu.wu}@nlpr.ia.ac.cn
burning21@stu.xjtu.edu.cn, wangliang@nlpr.ia.ac.cn

Abstract

Text classification is fundamental in natural language processing (NLP), and Graph Neural Networks (GNN) are recently applied in this task. However, the existing graph-based works can neither capture the contextual word relationships within each document nor fulfil the inductive learning of new words. In this work, to overcome such problems, we propose TextING[1] for inductive text classification via GNN. We first build individual graphs for each document and then use GNN to learn the fine-grained word representations based on their local structures, which can also effectively produce embeddings for unseen words in the new document. Finally, the word nodes are incorporated as the document embedding. Extensive experiments on four benchmark datasets show that our method outperforms state-of-the-art text classification methods.

1 Introduction

Text classification is one of the primary tasks in the NLP field, as it provides fundamental methodologies for other NLP tasks, such as spam filtering, sentiment analysis, intent detection, and so forth. Traditional methods for text classification include Naive Bayes (Androutsopoulos et al., 2000), k-Nearest Neighbor (Tan, 2006) and Support Vector Machine (Forman, 2008). They are, however, primarily dependent on the hand-crafted features at the cost of labour and efficiency.

There are several deep learning methods proposed to address the problem, among which Recurrent Neural Network (RNN) (Mikolov et al., 2010) and Convolutional Neural Network (CNN) (Kim, 2014) are essential ones. Based on them, extended models follow to leverage the classification performance, for instance, TextCNN (Kim, 2014), TextRNN (Liu et al., 2016) and TextRCNN (Lai et al., 2015). Yet they all focus on the locality of words and thus lack of long-distance and non-consecutive word interactions. Graph-based methods are recently applied to solve such issue, which do not treat the text as a sequence but as a set of co-occurrent words instead. For example, Yao et al. (2019) employ Graph Convolutional Networks (Kipf and Welling, 2017) and turns the text classification problem into a node classification one (TextGCN). Moreover, Huang et al. (2019) improve TextGCN by introducing the message passing mechanism and reducing the memory consumption.

However, there are two major drawbacks in these graph-based methods. First, the contextual-aware word relations within each document are neglected. To be specific, TextGCN (Yao et al., 2019) constructs a single graph with global relations between documents and words, where fine-grained text level word interactions are not considered (Wu et al., 2019; Hu et al., 2019a,b). In Huang et al. (2019), the edges of the graph are globally fixed between each pair of words, but the fact is that they may affect each other differently in a different text. Second, due to the global structure, the test documents are mandatory in training. Thus they are inherently *transductive* and have difficulty with *inductive* learning, in which one can easily obtain word embeddings for new documents with new structures and words using the trained model.

Therefore, in this work, we propose a novel Text classification method for INductive word representations via Graph neural networks, termed TextING. In contrast to previous graph-based approaches with global structure, we train a GNN that can depict the detailed word-word relations using only training documents, and generalise to new documents in test. We build individual graphs by applying the sliding window inside each doc-

[*] The first two authors contribute equally to this work.
[1] https://github.com/CRIPAC-DIG/TextING

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 334–339
July 5 - 10, 2020.

ument (Rousseau et al., 2015). The information of word nodes is propagated to their neighbours via the Gated Graph Neural Networks (Li et al., 2015, 2019), which is then aggregated into the document embedding. We also conduct extensive experiments to examine the advantages of our approach against baselines, even when words in test are mostly unseen (21.06% average gain in such inductive condition). Noticing a concurrent work (Nikolentzos et al., 2020) also reinforces the approach with a similar graph network structure, we describe the similarities and differences in the method section. To sum up, our contributions are threefold:

- We propose a new graph neural network for text classification, where each document is an individual graph and text level word interactions can be learned in it.
- Our approach can generalise to new words that absent in training, and it is therefore applicable for inductive circumstances.
- We demonstrate that our approach outperforms state-of-the-art text classification methods experimentally.

2 Method

TextING comprises three key components: the graph construction, the graph-based word interaction, and the readout function. The architecture is illustrated in Figure 1. In this section, we detail how to implement the three and how they work.

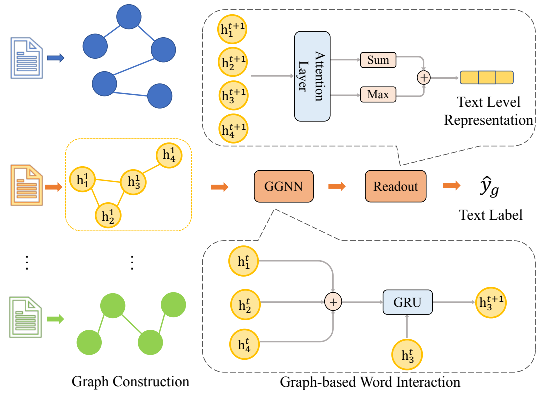


Figure 1: The architecture of TextING. As an example, upon a graph of document, every word node updates itself from its neighbours and they aggregate to the ultimate graph representation.

Graph Construction

We construct the graph for a textual document by representing unique words as vertices and co-occurrences between words as edges, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ the edges. The co-occurrences describe the relationship of words that occur within a fixed-size sliding window (length 3 at default) and they are undirected in the graph. Nikolentzos et al. (2020) also use a sliding window of size 2. However, they include a particular master node connecting to every other node, which means the graph is densely connected and the structure information is vague during message passing.

The text is preprocessed in a standard way, including tokenisation and stopword removal (Blanco and Lioma, 2012; Rousseau et al., 2015). Embeddings of the vertices are initialised with word features, denoted as $\mathbf{h} \in \mathbb{R}^{|\mathcal{V}| \times d}$ where d is the embedding dimension. Since we build individual graphs for each document, the word feature information is propagated and incorporated contextually during the word interaction phase.

Graph-based Word Interaction

Upon each graph, we then employ the Gated Graph Neural Networks (Li et al., 2015) to learn the embeddings of the word nodes. A node could receive the information $\mathbf{a}$ from its adjacent neighbours and then merge with its own representation to update. As the graph layer operates on the first-order neighbours, we can stack such layer t times to achieve high-order feature interactions, where a node can reach another node t hops away. The formulas of the interaction are:

$$\mathbf{a}^t = \mathbf{A}\mathbf{h}^{t-1}\mathbf{W}_a, \tag{1}$$

$$\mathbf{z}^t = \sigma\left(\mathbf{W}_z\mathbf{a}^t + \mathbf{U}_z\mathbf{h}^{t-1} + \mathbf{b}_z\right), \tag{2}$$

$$\mathbf{r}^t = \sigma\left(\mathbf{W}_r\mathbf{a}^t + \mathbf{U}_r\mathbf{h}^{t-1} + \mathbf{b}_r\right), \tag{3}$$

$$\tilde{\mathbf{h}}^t = \tanh\left(\mathbf{W}_h\mathbf{a}^t + \mathbf{U}_h(\mathbf{r}^t \odot \mathbf{h}^{t-1}) + \mathbf{b}_h\right), \tag{4}$$

$$\mathbf{h}^t = \tilde{\mathbf{h}}^t \odot \mathbf{z}^t + \mathbf{h}^{t-1} \odot \left(1 - \mathbf{z}^t\right), \tag{5}$$

where $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ is the adjacency matrix, σ is the sigmoid function, and all $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{b}$ are trainable weights and biases. $\mathbf{z}$ and $\mathbf{r}$ function as the update gate and reset gate respectively to determine to what degree the neighbour information contributes to the current node embedding.

Readout Function

After the word nodes are sufficiently updated, they are aggregated to a graph-level representation for the document, based on which the final prediction

Table 1: The statistics of the datasets including both short (sentence) and long (paragraph) documents. The vocab means the number of unique words in a document. The Prop.NW denotes the proportion of new words in test.

Dataset	# Docs	# Training	# Test	# Classes	Max.Vocab	Min.Vocab	Avg.Vocab	Prop.NW
MR	10,662	7,108	3,554	2	46	1	18.46	30.07%
R8	7,674	5,485	2,189	8	291	4	41.25	2.60%
R52	9,100	6,532	2,568	52	301	4	44.02	2.64%
Ohsumed	7,400	3,357	4,043	23	197	11	79.49	8.46%

is produced. We define the readout function as:

$$\mathbf{h}_v = \sigma\left(f_1(\mathbf{h}_v^t)\right) \odot \tanh\left(f_2(\mathbf{h}_v^t)\right), \quad (6)$$

$$\mathbf{h}_{\mathcal{G}} = \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}} \mathbf{h}_v + \text{Maxpooling}\left(\mathbf{h}_1 ... \mathbf{h}_{\mathcal{V}}\right), \quad (7)$$

where f_1 and f_2 are two multilayer perceptrons (MLP). The former performs as a soft attention weight while the latter as a non-linear feature transformation. In addition to averaging the weighted word features, we also apply a max-pooling function for the graph representation $\mathbf{h}_{\mathcal{G}}$. The idea behind is that every word plays a role in the text and the keywords should contribute more explicitly.

Finally, the label is predicted by feeding the graph-level vector into a softmax layer. We minimise the loss through the cross-entropy function:

$$\hat{y}_{\mathcal{G}} = \text{softmax}\left(\mathbf{W}\mathbf{h}_{\mathcal{G}} + \mathbf{b}\right), \quad (8)$$

$$\mathcal{L} = -\sum_i y_{\mathcal{G}i} \log\left(\hat{y}_{\mathcal{G}i}\right), \quad (9)$$

where $\mathbf{W}$ and $\mathbf{b}$ are weights and bias, and $y_{\mathcal{G}i}$ is the i-th element of the one-hot label.

Model Variant

We also extend our model with a multichannel branch TextING-M, where graphs with local structure (original TextING) and graphs with global structure (subgraphs from TextGCN) work in parallel. The nodes remain the same whereas the edges of latter are extracted from the large graph (built on the whole corpus) for each document. We train them separately and make them vote 1:1 for the final prediction. Although it is not the inductive case, our point is to investigate whether and how the two could complement each other from micro and macro perspectives.

3 Experiments

In this section, we aim at testing and evaluating the overall performance of TextING. During the experimental tests, we principally concentrate on three concerns: (i) the performance and advantages of our approach against other comparable models, (ii) the adaptability of our approach for words that are never seen in training, and (iii) the interpretability of our approach on how words impact a document.

Datasets. For the sake of consistency, we adopt four benchmark tasks the same as in (Yao et al., 2019): (i) classifying movie reviews into positive or negative sentiment polarities (MR)[2], (ii) & (iii) classifying documents that appear on Reuters newswire into 8 and 52 categories (R8 and R52 respectively)[3], (iv) classifying medical abstracts into 23 cardiovascular diseases categories (Ohsumed)[4]. Table 1 demonstrates the statistics of the datasets as well as their supplemental information.

Baselines. We consider three types of models as baselines: (i) traditional deep learning methods including TextCNN (Kim, 2014) and TextRNN (Liu et al., 2016), (ii) simple but efficient strategies upon word features including fastText (Joulin et al., 2017) and SWEM (Shen et al., 2018), and (iii) graph-based methods for text classification including TextGCN (Yao et al., 2019) and Huang et al. (2019).

Experimental Set-up. For all the datasets, the training set and the test set are given, and we randomly split the training set into the ratio 9:1 for actual training and validation respectively. The hyperparameters were tuned according to the performance on the validation set. Empirically, we set the learning rate as 0.01 with Adam (Kingma and Ba, 2015) optimiser and the dropout rate as 0.5. Some depended on the intrinsic attributes of the dataset, for example, the word interaction step and the sliding window size. We refer to them in the parameter sensitivity subsection.

Regarding the word embeddings, we used the pre-trained GloVe (Pennington et al., 2014)[5] with

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

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

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

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

Table 2: Test accuracy (%) of various models on four datasets. The mean ± standard deviation of our model is reported according to 10 times run. Note that some baseline results are from (Yao et al., 2019).

Model	**MR**	**R8**	**R52**	**Ohsumed**
CNN (Non-static)	77.75 ± 0.72	95.71 ± 0.52	87.59 ± 0.48	58.44 ± 1.06
RNN (Bi-LSTM)	77.68 ± 0.86	96.31 ± 0.33	90.54 ± 0.91	49.27 ± 1.07
fastText	75.14 ± 0.20	96.13 ± 0.21	92.81 ± 0.09	57.70 ± 0.49
SWEM	76.65 ± 0.63	95.32 ± 0.26	92.94 ± 0.24	63.12 ± 0.55
TextGCN	76.74 ± 0.20	97.07 ± 0.10	93.56 ± 0.18	68.36 ± 0.56
Huang et al. (2019)	-	97.80 ± 0.20	94.60 ± 0.30	69.40 ± 0.60
TextING	79.82 ± 0.20	98.04 ± 0.25	95.48 ± 0.19	70.42 ± 0.39
TextING-M	80.19 ± 0.31	98.13 ± 0.12	95.68 ± 0.35	70.84 ± 0.52

$d = 300$ as the input features while the out-of-vocabulary (OOV) words' were randomly sampled from a uniform distribution [-0.01, 0.01]. For a fair comparison, the other baseline models shared the same embeddings.

Results. Table 2 presents the performance of our model as well as the baselines. We observe that graph-based methods generally outperform other types of models, suggesting that the graph model benefits to the text processing. Further, TextING ranks top on all tasks, suggesting that the individual graph exceeds the global one. Particularly, the result of TextING on MR is remarkably higher. Because the short documents in MR lead to a low-density graph in TextGCN, it restrains the label message passing among document nodes, whereas our individual graphs (documents) do not rely on such label message passing mechanism. Another reason is that there are approximately one third new words in test as shown in Table 1, which implies TextING is more friendly to unseen words. The improvement on R8 is relatively subtle since R8 is simple to fit and the baselines are rather satisfying. The proportion of new words is also low on R8.

The multichannel variant also performs well on all datasets. It implies the model can learn different patterns through different channels.

Under Inductive Condition. To examine the adaptability of TextING under inductive condition, we reduce the amount of training data to 20 labelled documents per class and compare it with TextGCN. Word nodes absent in the training set are masked for TextGCN to simulate the inductive condition. In this scenario, most of the words in the test set are unseen during training, which behaves like a rigorous cold-start problem. The result of both models on MR and Ohsumed are listed in Table 3. An average gain of 21.06% shows that TextING is much less impacted by the reduction of exposed words. In addition, a tendency of test performance and gain with different percentages of training data on MR is illustrated as Figure 2. TextING shows a consistent improvement when increasing number of words become unseen.

Table 3: Accuracy (%) of TextGCN and TextING on MR and Ohsumed, where MR uses 40 labelled documents (0.5% of full training data) and Ohsumed uses 460 labelled documents (13.7% of full training data).

Model	**MR***	**Ohsumed***
TextGCN	53.15	47.24
TextING	64.43	57.11
# Words in Training	465	7,009
# New Words in Test	18,299	7,148

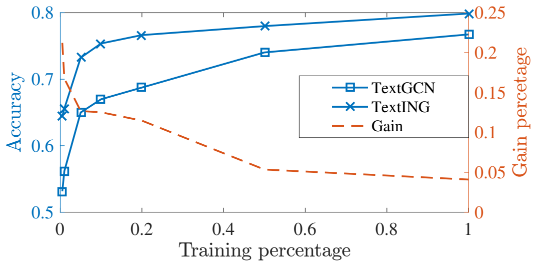
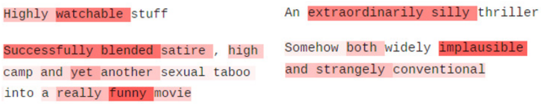


Figure 2: Test performance and gain with different percent of training data ranging from 0.005 to 1 on MR. The less data in training, the more new words in test.

Case Study. To understand what is of importance that TextING learns for a document, we further visualise the attention layer (i.e. the readout function), illustrated as Figure 3. The highlighted words are proportional to the attention weights, and they show a positive correlation to the label, which interprets how TextING works in sentiment analysis.

Parameter Sensitivity. Figure 4 exhibits the performance of TextING with a varying number of the graph layer on MR and Ohsumed. The result reveals that with the increment of the layer, a node could receive more information from high-order



Highly watchable stuff

Successfully blended satire , high camp and yet another sexual taboo into a really funny movie

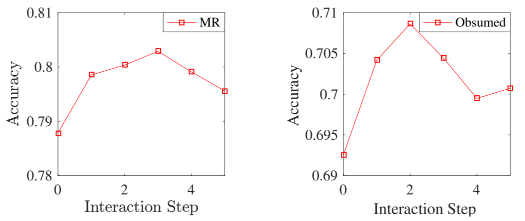
An extraordinarily silly thriller

Somehow both widely implausible and strangely conventional

(a) Positive reviews (b) Negative reviews

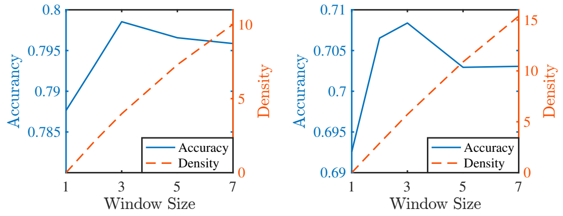
Figure 3: Attention visualisation of positive and negative movie reviews in MR.

neighbours and learn its representation more accurately. Nevertheless, the situation reverses with a continuous increment, where a node receives from every node in the graph and becomes over-smooth. Figure 5 illustrates the performance as well as the graph density of TextING with a varying window size on MR and Ohsumed. It presents a similar trend as the interaction step's when the number of neighbours of a node grows.



(a) MR (b) Ohsumed

Figure 4: Accuracy with varying interaction steps.



(a) MR (b) Ohsumed

Figure 5: Accuracy with varying graph density.

4 Conclusion

We proposed a novel graph-based method for inductive text classification, where each text owns its structural graph and text level word interactions can be learned. Experiments proved the effectiveness of our approach in modelling local word-word relations and word significances in the text.

Acknowledgement

This work is supported by National Natural Science Foundation of China (U19B2038, 61772528) and National Key Research and Development Program (2018YFB1402600, 2016YFB1001000).

References

Ion Androutsopoulos, John Koutsias, Konstantinos V Chandrinos, George Paliouras, and Constantine D Spyropoulos. 2000. An evaluation of naive bayesian anti-spam filtering. *arXiv preprint cs/0006013*.

Roi Blanco and Christina Lioma. 2012. Graph-based term weighting for information retrieval. *Information retrieval*.

George Forman. 2008. Bns feature scaling: an improved representation over tf-idf for svm text classification. In *CIKM*. ACM.

Fenyu Hu, Yanqiao Zhu, Shu Wu, Weiran Huang, Liang Wang, and Tieniu Tan. 2019a. Graphair: Graph representation learning with neighborhood aggregation and interaction. *arXiv preprint arXiv:1911.01731*.

Fenyu Hu, Yanqiao Zhu, Shu Wu, Liang Wang, and Tieniu Tan. 2019b. Hierarchical graph convolutional networks for semi-supervised node classification. *arXiv preprint arXiv:1902.06667*.

Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng WANG. 2019. Text level graph neural network for text classification. In *EMNLP*.

Armand Joulin, Edouard Grave, and Piotr Bojanowski Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *EACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

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

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. In *ICLR*.

Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *ACM*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *IJCAI*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *ISCA*.

Giannis Nikolentzos, Antoine Jean-Pierre Tixier, and Michalis Vazirgiannis. 2020. Message passing attention networks for document understanding. In *AAAI*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

François Rousseau, Emmanouil Kiagias, and Michalis Vazirgiannis. 2015. Text categorization as a graph classification problem. In *ACL*.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *ACL*.

Songbo Tan. 2006. An effective refinement strategy for KNN text classifier. *Expert Systems with Applications*.

Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *AAAI*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *AAAI*.