BertGCN: Transductive Text Classification by Combining GCN and BERT

Yuxiao Lin[♠], Yuxian Meng[♣], Xiaofei Sun[♣]
Qinghong Han[♣], Kun Kuang[♠], Jiwei Li[♠♣] and Fei Wu[♠]
[♠]Computer Science Department, Zhejiang University
[♣] ShannonAI
{yuxiaolinling, kunkuang, jiwei_li, wufei}@zju.edu.cn
{yuxian_meng, xiaofei_sun, qinghong_han}@shannonai.com

Abstract

In this work, we propose BertGCN, a model that combines large scale pretraining and transductive learning for text classification. BertGCN constructs a heterogeneous graph over the dataset and represents documents as nodes using BERT representations. By jointly training the BERT and GCN modules within BertGCN, the proposed model is able to leverage the advantages of both worlds: large-scale pretraining which takes the advantage of the massive amount of raw data and transductive learning which jointly learns representations for both training data and unlabeled test data by propagating label influence through graph convolution. Experiments show that BertGCN achieves SOTA performances on a wide range of text classification datasets.[1]

1 Introduction

Text classification is a core task in natural language processing (NLP) and has been used in many real-world applications such as spam detection (Wang, 2010) and opinion mining (Bakshi et al., 2016). Transductive learning (Vapnik, 1998) is a particular method for text classification which makes use of both labeled and unlabeled examples in the training process. Graph neural networks (GNNs) serve as an effective approach for transductive learning (Yao et al., 2019; Liu et al., 2020). In these works, a graph is constructed to model the relationship between documents. Nodes in the graph represent text units such as words and documents, while edges are constructed based on the semantic similarity between nodes. GNNs are then applied to the graph to perform node classification. The merits of GNNs and transductive learning are as follows: (1) the decision for an instance (both training and test) does not depend merely on itself, but also its neighbors. This makes the model more immune to data outliers; (2) at the training time, since the model propagates influence from supervised labels across both training and test instances through graph edges, unlabeled data also contributes to the process of representation learning, and consequently higher performances.

Large-scale pretraining has recently demonstrated their effectiveness on a variety of NLP tasks (Devlin et al., 2018; Liu et al., 2019). Trained on large-scale unlabeled corpora in an unsupervised manner, large-scale pretrained models are able to learn implicit but rich text semantics in language at scale. Intuitively, large-scale pretrained models have potentials to benefit transductive learning. However, existing models for transductive text classification (Yao et al., 2019; Liu et al., 2020) did not take large-scale pretraining into consideration, and its effectiveness still remains unclear.

In this work, we propose BertGCN, a model that combines the advantages of both large-scale pretraining and transductive learning for text classification. BertGCN constructs a heterogeneous graph for the corpus with node being word or document, and node embeddings are initialized with pretrained BERT representations, and uses graph convolutional networks (GCN) for classification. By jointly training the BERT and GCN modules, the proposed model is able to leverage the advantages of both worlds: large-scale pretraining which takes the advantage of the massive amount of raw data and transductive learning which jointly learns representations for both training data and unlabeled test data by propagating label influence through graph edges. The proposed BertGCN model successfully combines the powers of large-scale pretraining and graph networks, and achieves new state-of-the-art performances on a wide range of text classification datasets.

[1] Code available at `https://github.com/ZeroRin/BertGCN`.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1456–1462
August 1–6, 2021. ©2021 Association for Computational Linguistics

2 Related Work

Graph neural networks (GNNs) are connectionist models that capture dependencies and relations between graph nodes via message passing through edges that connect nodes (Scarselli et al., 2008; Hamilton et al., 2017; Xu et al., 2018). GNNs are practically categorized into (Wu et al., 2020): graph convolutional networks (Kipf and Welling, 2016a; Wu et al., 2019), graph attention networks (Veličković et al., 2017; Zhang et al., 2018a), graph auto-encoder (Cao et al., 2016; Kipf and Welling, 2016b), graph generative networks (De Cao and Kipf, 2018; Li et al., 2018b) and graph spatial-temporal networks (Li et al., 2017; Yu et al., 2017). GNNs serve as powerful tools to utilize the relationship between different objects, and have been applied to various domains such as traffic prediction (Yu et al., 2018; Zhang et al., 2018a) and recommendation (Zhang et al., 2020; Monti et al., 2017). In the context of NLP, GNNs have achieved remarkable successes across a wide range of end tasks such as relation extraction (Zhang et al., 2018b), semantic role labeling (Marcheggiani and Titov, 2017), data-to-text generation (Marcheggiani and Perez-Beltrachini, 2018), machine translation (Bastings et al., 2017) and question answering (Song et al., 2018; De Cao et al., 2018).

The prevalence of neural networks has motivated a diverse array of works on developing neural models for text classification. Different neural model architectures (Kim, 2014; Zhou et al., 2015; Radford et al., 2018; Chai et al., 2020) have demonstrated their effectiveness against traditional statistical feature based methods (Wallach, 2006). Other works leverage label embeddings and jointly train them along with input texts (Wang et al., 2018; Pappas and Henderson, 2019). More recently, the success achieved by large-scale pretraining models has spurred great interests in adapting the large-scale pretraining framework (Devlin et al., 2018) into text classification (Reimers and Gurevych, 2019), leading to remarkable progressive on few-shot (Mukherjee and Awadallah, 2020) and zero-shot (Ye et al., 2020) learning.

Our work is inspired by the work of using graph neural networks for text classification (Yao et al., 2019; Huang et al., 2019; Zhang and Zhang, 2020). But different from these works, we focus on combining large-scale pretrained models and GNNs, and show that GNNs can significantly benefit from large-scale pretraining. Existing works that combine BERT and GNNs uses graph to model relationships between tokens within a single document sample (Lu et al., 2020; He et al., 2020b), which fall into the category of inductive learning. Different from these works, we use graph to model relationships between different samples from the whole corpus to utilize the similarity between labeled and unlabeled documents, and uses GNNs to learn their relationships.

3 Method

3.1 BertGCN

In the proposed BertGCN model, we initialize representations for document nodes in a text graph using a BERT-style model (e.g., BERT, RoBERTa). These representations are used as inputs to GCN. Document representations will then be iteratively updated based on the graph structures using GCN, the outputs of which are treated as final representations for document nodes, and are sent to the softmax classifier for predictions. In this way, we are able to leverage the complementary strengths of pretrained models and graph models.

Specifically, we construct a heterogeneous graph containing both word nodes and document nodes following TextGCN (Yao et al., 2019). We define word-document edges and word-word edges based on the term frequency-inverse document frequency (TF-IDF) and positive point-wise mutual information (PPMI), respectively. The weight of an edge between two nodes i and j is defined as:

$$A_{i,j} = \begin{cases} \text{PPMI}(i,j), & i,j \text{ are words and } i \neq j \\ \text{TF-IDF}(i,j), & i \text{ is document, } j \text{ is word} \\ 1, & i = j \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

In TextGCN, an identity matrix $X = I_{n_{\text{doc}}+n_{\text{word}}}$ is used as initial node features, where n_{doc} is the number of document nodes, n_{word} is the number of word nodes (including both training and test). In BertGCN, we use a BERT-style model to obtain the document embeddings, and treat them as input representations for document nodes. Document node embeddings are denoted by $X_{\text{doc}} \in \mathbb{R}^{n_{\text{doc}} \times d}$, where d is the embedding dimensionality. Overall, the initial node feature matrix is given by:

$$X = \begin{pmatrix} X_{\text{doc}} \\ 0 \end{pmatrix}_{(n_{\text{doc}}+n_{\text{word}})\times d} \tag{2}$$

We feed X into a GCN model (Kipf and Welling, 2016a) which iteratively propagates messages across training and test examples. Specifically, the output feature matrix of the i-th GCN layer $L^{(i)}$ is computed as

$$L^{(i)} = \rho(\tilde{A}L^{(i-1)}W^{(i)}) \tag{3}$$

where ρ is an activation function, $\tilde{A}$ is the normalized adjacency matrix and $W^{(i)} \in \mathbb{R}^{d_{i-1} \times d_i}$ is a weight matrix of the layer. $L^{(0)} = X$ is the input feature matrix of the model. Outputs of GCN are treated as final representations for documents, which is then fed to the softmax layer for classification:

$$Z_{\text{GCN}} = \text{softmax}(g(X, A)) \tag{4}$$

where g represents the GCN model. We use the cross entropy loss over labeled document nodes to jointly optimize parameters for BERT and GCN.

3.2 Interpolating BERT and GCN Predictions

Practically, we find that optimizing BertGCN with a auxiliary classifier that directly operates on BERT embeddings leads to faster convergence and better performances. Specifically, we construct an auxiliary classifier by directly feeding document embeddings (denoted by X) to a dense layer with softmax activation:

$$Z_{\text{BERT}} = \text{softmax}(WX) \tag{5}$$

The final training objective is the linear interpolation of the prediction from BertGCN and the prediction from BERT, which is given by:

$$Z = \lambda Z_{\text{GCN}} + (1 - \lambda) Z_{\text{BERT}} \tag{6}$$

where λ controls the tradeoff between the two objectives. $\lambda = 1$ means we use the full BertGCN model, and $\lambda = 0$ means we only use the BERT module. When $\lambda \in (0, 1)$, we are able to balance the predictions from both models, and the BertGCN model can be better optimized.

The explanation for better performances achieved by the interpolation is as follows: The Z_{BERT} directly operates on the input of GCN, making sure that inputs to GCN are regulated and optimized towards the objective. This helps the multi-layer GCN model to overcome intrinsic drawbacks such as gradient vanishing or over-smoothing (Li et al., 2018a), and thus leads to better performances.

3.3 Optimization using Memory Bank

The original GCN model uses the full-batch gradient descent method for training, which is intractable for the proposed BertGCN model, since the full-batch method can not be applied to BERT due to the memory limitation. Inspired by techniques in contrastive learning which decouples the dictionary size from the mini-batch size (Wu et al., 2018; He et al., 2020a), we introduce a memory bank that stores all document embeddings to decouple the training batch size from the total number of nodes in the graph.

Specifically, during training, we maintain a memory bank M that tracks input features for all document nodes. At the beginning of each epoch, we first compute all document embeddings using the *current* BERT module and store them in M. During each iteration, we sample a mini batch from both labeled and unlabeled document nodes with the index set $B = \{b_0, b_1 ... b_n\}$, where n is the mini-batch size. We then compute their document embeddings M_B also using the *current* BERT module and update the corresponding memories in M.[2] Next, we use the updated M as input to derive the GCN output and compute the loss for the current mini batch. For back-propagation, M is considered as constant except the records in B.

With the memory bank, we are able to efficiently train the BertGCN model including the BERT module. However, during training, the embeddings in the memory bank are computed using the BERT module at different steps in an epoch and are thus inconsistent. To overcome this issue, we set a small learning rate for the BERT module to improve consistency of the stored embeddings. With low learning rate the training takes more time. In order to speed up training, we fine-tune a BERT model on the target dataset before training begins, and use it to **initialize** the BERT parameters in BertGCN.

4 Experiments

4.1 Experiment Setups

We run experiments on five widely-used text classification benchmarks: 20 Newsgroups (20NG)[3], R8

[2] Note that the BERT module used to compute M_B is the one finished training in the last iteration, which is different from the the BERT module used to compute the initial M.

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

Model	20NG	R8	R52	Ohsumed	MR
TextGCN	86.3	97.1	93.6	68.4	76.7
SGC	88.5	97.2	94.0	68.5	75.9
BERT	85.3	97.8	96.4	70.5	85.7
RoBERTa	83.8	97.8	96.2	70.7	89.4
BertGCN	89.3	98.1	**96.6**	**72.8**	86.0
RoBERTaGCN	**89.5**	**98.2**	96.1	**72.8**	**89.7**
BertGAT	87.4	97.8	96.5	71.2	86.5
RoBERTaGAT	86.5	98.0	96.1	71.2	89.2

Table 1: Results for different models on transductive text classification datasets. We run all models 10 times and report the mean test accuracy.

and R52[4], Ohsumed[5] and Movie Review (MR)[6].

We compare BertGCN to current state-of-the-art pretrained and GCN models: TextGCN (Yao et al., 2019), SGC (Wu et al., 2019), BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019). Details for datasets and baseline are left in the supplementary material.

We follow protocols in TextGCN to preprocess data. For BERT and RoBERTa, we use the output feature of the [CLS] token as the document embedding, followed by a feedforward layer to derive the final prediction. We use BERT$_{\text{base}}$ and a two-layer GCN to implement BertGCN. We initialize the learning rate to 1e-3 for the GCN module and 1e-5 for the fine-tuned BERT module. We also implement our model with RoBERTa and GAT (Veličković et al., 2017). GAT variants are trained over the same graph as GCN variants, but learn edge weights through attention mechanism instead of using predefined weight matrix.

4.2 Main Results

Table 1 presents the test accuracy of each model. We can see that BertGCN and RoBERTaGCN perform the best across all datasets. Only using BERT and RoBERTa generally performs better than GCN variants except 20NG, which is due to the great merits brought by large-scale pretraining. Compared with BERT and RoBERTa, the performance boost from BertGCN and RoBERTaGCN is significant on the 20NG and Ohsumed datasets. This is because the average length in 20NG and Ohsumed is much longer than that in other datasets: the graph is constructed using word-document statistics, which means that long texts may produce more document connections transited via an intermediate word node, and this potentially benefits message passing through the graph, leading to better performances when combined with GCN. This may also explain why GCN models perform better than BERT models on 20NG. For datasets with shorter documents such as R52 and MR, the power of graph structure is limited, and thus the performance boost is smaller relative to 20NG. BertGAT and RoBERTaGAT can also benefit from the graph structure, but their performance are not as good as GCN variants due to the lack of edge weight information.

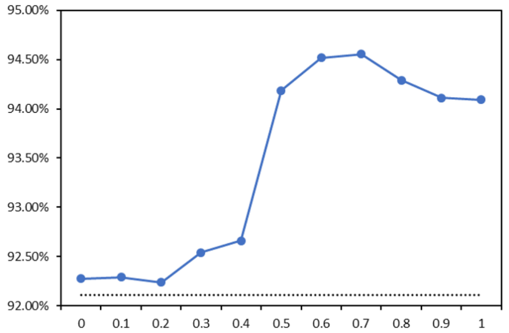

Figure 1: Accuracy of RoBERTaGCN when varying λ on 20NG development set. The dotted line indicates the corresponding RoBERTa baseline.[7]

Strategy	*w/ both*	*w/o finetune*	*w/o small lr.*	*w/o both*
Accuracy	94.7	93.8	10.3[8]	10.3[8]

Table 2: Accuracy on 20NG development set for different strategies. "finetune" means we use the finetuned RoBERTa as initialization, and "small lr." means we use a smaller learning rate for the RoBERTa module.

4.3 The Effect of λ

λ controls the trade-off between training BertGCN and BERT. The optimal value of λ can be different for different tasks. Fig.1 shows the accuracy of RoBERTaGCN with different λ. On 20NG, the accuracy is consistently higher with larger λ value. This can be explained by the high performance of graph-based methods on 20NG. The model reaches its best when $\lambda = 0.7$, performing slightly better than only using the GCN prediction ($\lambda = 1$).

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

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

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

4.4 The Effect of Strategies in Joint Training

To overcome inconsistency of embeddings in the memory bank, we set a smaller learning rate for the BERT module and use a finetuned BERT model for initialization. We evaluate the effect of the two strategies. Table 2 shows the results of RoBERTaGCN on 20NG with and without these strategies. With the same learning rate for RoBERTa and GCN, the model cannot be trained due to inconsistency in the memory bank, regardless of whether the fine-tuned RoBERTa is used. Models can be successfully trained when we set a smaller learning rate for the RoBERTa module, and additional using finetuned RoBERTa leads to the best performance.

5 Conclusion and Future Work

In this work, we propose BertGCN, which takes the best advantages from both large-scale pretraining models and transductive learning for text classification. We efficiently train BertGCN by using a memory bank that stores all document embeddings and updates part of them with respect to the sampled mini batch. The framework of BertGCN can be built on top of any document encoder and any graph model. Experiments demonstrate the power of the proposed BertGCN model. However, in this work, we only use document statistics to build the graph, which might be sub-optimal compared to models that are able to automatically construct edges between nodes. We leave this in future work.

Acknowledgement

This work is supported by National Key R&D Program of China (2020AAA0105200) and Beijing Academy of Artificial Intelligence (BAAI).

References

Rushlene Kaur Bakshi, Navneet Kaur, Ravneet Kaur, and Gurpreet Kaur. 2016. Opinion mining and sentiment analysis. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 452–455. IEEE.

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, Copenhagen, Denmark. Association for Computational Linguistics.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep neural networks for learning graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Duo Chai, Wei Wu, Qinghong Han, Fei Wu, and Jiwei Li. 2020. Description based text classification with reinforcement learning. In *International Conference on Machine Learning*, pages 1371–1382. PMLR.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2018. Question answering by reasoning across documents with graph convolutional networks. *arXiv preprint arXiv:1808.09920*.

Nicola De Cao and Thomas Kipf. 2018. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*.

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*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020a. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738.

Qi He, Han Wang, and Yue Zhang. 2020b. Enhancing generalization in natural language inference by syntax. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4973–4978.

Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2019. Text level graph neural network for text classification. *arXiv preprint arXiv:1910.02356*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Thomas N Kipf and Max Welling. 2016a. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Thomas N Kipf and Max Welling. 2016b. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018a. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[7] The original training/test split of 20NG is based on post date, but the development set is randomly sampled from the original training set. The accuracy on test set is thus much lower than that on development set.

[8] Experiments without a small lr. failed to converge.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. 2018b. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*.

Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. 2020. Tensor graph convolutional networks for text classification.

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. *arXiv preprint arXiv:1907.11692*.

Zhibin Lu, Pan Du, and Jian-Yun Nie. 2020. Vgcn-bert: augmenting bert with graph embedding for text classification. In *European Conference on Information Retrieval*, pages 369–382. Springer.

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 1–9, Tilburg University, The Netherlands. Association for Computational Linguistics.

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, Copenhagen, Denmark. Association for Computational Linguistics.

Federico Monti, Michael M Bronstein, and Xavier Bresson. 2017. Geometric matrix completion with recurrent multi-graph neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3700–3710.

Subhabrata Mukherjee and Ahmed Hassan Awadallah. 2020. Uncertainty-aware self-training for text classification with few labels.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

Nikolaos Pappas and James Henderson. 2019. Gile: A generalized input-label embedding for text classification. *Transactions of the Association for Computational Linguistics*, 7:139–155.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2018. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *arXiv preprint arXiv:1809.02040*.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1165–1174.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Hanna M Wallach. 2006. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pages 977–984.

Alex Hai Wang. 2010. Don't follow me: Spam detection in twitter. In *2010 international conference on security and cryptography (SECRYPT)*, pages 1–10. IEEE.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*.

Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. 2019. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

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.

Zhiquan Ye, Yuxia Geng, Jiaoyan Chen, Jingmin Chen, Xiaoxiao Xu, SuHang Zheng, Feng Wang, Jun Zhang, and Huajun Chen. 2020. Zero-shot text classification via reinforced self-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3014–3024, Online. Association for Computational Linguistics.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3634–3640.

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.

Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit Yan Yeung. 2018a. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*.

Shengyu Zhang, Ziqi Tan, Zhou Zhao, Jin Yu, Kun Kuang, Tan Jiang, Jingren Zhou, Hongxia Yang, and Fei Wu. 2020. Comprehensive information integration modeling framework for video titling. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2744–2754.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018b. Graph convolution over pruned dependency trees improves relation extraction. *arXiv preprint arXiv:1809.10185*.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

A Dataset Details

- The 20NG dataset[9] contains 18,846 newsgroups posts from 20 different topics. We use the bydate version which splits the dataset to 11,314 train samples and 7,532 test samples based on the posting date.

- R8 and R52[10] are two subsets of the Reuters dataset with respectively 8 and 52 categories. R8 has 5,485 training and 2,189 test documents. R52 has 6,532 training and 2,568 test documents.

- The OHSUMED test collection[11] is a set of references from MEDLINE, the online medical information database. Following previous works, we use 7,400 documents belonging to one of the 23 disease categories to form a classification dataset, with 3,357 documents for training and 4,043 for test.

- MR (Pang and Lee, 2005)[12] is a movie review dataset for binary sentiment classification. The corpus has 10,662 reviews. We use the train/test split in Tang et al. (2015)

B Baselines

- TextGCN (Yao et al., 2019): TextGCN is a model that operates graph convolution over a word-document heterogeneous graph. Node features are initialized using an identity matrix.
- SGC (Wu et al., 2019): Simple Graph Convolution is a variant of GCN that reduces the complexity of GCN by removing nonlinearities and collapsing weight matrices between consecutive layers.
- BERT (Devlin et al., 2018): BERT is a large-scale pretrained NLP model.
- RoBERTa (Liu et al., 2019): a robustly optimized BERT model that improves upon BERT with different pretraining methods.

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

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

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

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