Text Classification With Neural Networks

Guolun Li*
Department of Computer Science
University of Toronto
Toronto, ON M5S 1A1
polo.li@mail.utoronto.ca

Fengkai Ye*
Department of Computer Science
University of Toronto
Toronto, ON M5S 1A1
fengkai.ye@mail.utoronto.ca

Huifeng Wu*
Department of Computer Science
University of Toronto
Toronto, ON M5S 1A1
kentwhf.wu@mail.utoronto.ca

Abstract

We evaluate two text classification methodologies, Simple Graph Convolution (SGC) and Neural Attentive Bag-of-Entities(NABoE), by retraining the models and obtaining competitive state-of-the-art accuracy. Furthermore, we perform sensitivity analysis on the hyperparameters.We also extend the methods to a standard benchmark AG News to compare their strengths and weaknesses; NABoE takes more time to train while SGC requires large amount of memory for graph building.

1 Introduction

Text classification has been a fundamental task in Natural Language Processing (NLP). In recent years, the rise of deep learning has brought the performance of text classification models to a new level. With the aid of deep learning, applications of text classification grows rapidly and are impacting on many fields such as language detection, document organization, news filtering, fraud and spam detection. In our project, we focus on analysing and comparing two innovative models, Simple Graph Convolution (SGC) and Neural Attentive Bag-of-Entities (NABoE) and eventually evaluate their performance on AG News dataset for news categorization.

2 Related Work

2.1 Graph Convolutional Network

Graph convolutional networks (GCN) are a variant of convolutional neural networks. Kipf et al. proposed this novel network that operates convolution directly on a graph formed by its nodes and adjacency matrix along subsets of labelled nodes [1]. The network tends to classify all nodes in a graph. It was initially experimented in the citation dataset Cora; however, it has been broadly applied to text classification [2], recommender systems [3; 4; 5], and knowledge graphs [6; 7]. Multiple layers of first-order approximated spectral filters are stacked to learn the graph. GCN is expected to solve the overfitting problem regarding local neighbourhood structures where node degree distribution is complicated [1].

*All authors made equal to the project. The naming order is random.

Preprint. Under review.

2.2 Entity Linking and Attention Mechanism

To address text classification problems, past studies attempted to use entities in a knowledge base to capture the semantics in documents. However, linking each word to the correct entity behind it is a difficult task. Nedelchev et al. proposed an end-to-end entity linking method with knowledge graph embeddings [10]. Peng et al. proposed a linking system to create a set of relevant entities to the context [15]. Still, these approaches produce disambiguation errors and sometimes include entities irrelevant to the context. In recent years, attention mechanism has led tremendous improvements in different NLP areas. Gang et al. proposed a bidirectional LSTM with attention mechanism and convolutional layers for text classification [8].

3 Model Architecture

3.1 Simple Graph Convolution

SGC [13] can be seen as a simplified linear version of Graph Convolutional Network. GCN is a multi-layer neural network similar multi-layer perceptron (MLP), with major difference in feature propagation between each layer.

We here denote each of the n node in graph as v_i and each node has features of dimension d, we denote the feature as x_i. We construct the matrix X with $X = [x_1, ..., x_n]^T$. For the adjacency matrix A, each element A_{ij} denotes the weigh of the edge between node v_i and v_j. The degree matrix D is defined as a diagonal matrix with $d_{ii} = \sum_j a_{ij}$. For the loss calculation, GCN uses a graph Laplacian regularization term in the loss function with the following formula:

$$L = L_0 + \lambda L_{reg} \text{ where } L_{reg} = f(X)^T \vec{\Delta} f(X) \text{ and } \vec{\Delta} = D - A \tag{1}$$

Each of the layer of feature are form by applying a spectral filter on previous layer's features. And each layer follows the propagation rule of

$$H^{l+1} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^l W^l) \tag{2}$$

where $\tilde{A} = A + I_N$ and $\tilde{D}$ is a diagonal matrix with $\tilde{D_i} i = \sum_j \tilde{A_{ij}}$ and W^l is a trainable weight matrix. The last layer of the GCNs model predicts the label of each node using a softmax classifier. In comparison to GCN, SGC abandons the activation function in each layer, making each layer linear with respect to the previous layer. And thus the resulting classifier for a K layer SGC becomes

$$Y = \text{softmax}((\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}})^K X W) \tag{3}$$

The computation of $(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}})^K$ requires no weight and hence can be pre-processed, which reduces redundant computation. Denote it as $\bar{X}$, and this leads to a simple and compact classifier of

$$Y = \text{softmax}(\bar{X} W) \tag{4}$$

which is essentially a multi-class logistic regression on pre-processed features $\bar{X}$.

3.2 Neural Attentive Bag-of-Entities

Compared to past work, NABoE focuses on a small subset of entities more relevant to the context and less ambiguous in meaning by incorporating attention mechanism into the entity linking process [11]. The attention mechanism in NABoE emphasizes on two tasks: disambiguation of entity names (for the same word) and detection of entities relevant to the context. We use the open-source tool Wikipedia2Vec[12] to obtain word and entity embeddings pre-trained based on Wikipedia. Optionally, the embedding weights can be trained along with the model. Wikipedia also serves as a word-to-entity dictionary. Given a target context C with words $w = (w_1, ..., w_N)$, all possible referent entities $e = (e_1, ..., e_K)$ are generated from these N words by extracting corresponding entities in Wikipedia. We denote their vector embeddings as d-dimensional vectors $v_{w_1}, .., v_{w_N}, v_{e_1}, ...v_{e_K}$. Then, the word-based representation of C is firstly computed as the average of the N word-embeddings:

$$z_{word} = \frac{1}{N} \sum_{i=1}^{N} v_{w_i} \tag{5}$$

Then, the attention paid to each entity e_i is determined as

$$a_{e_i} = \text{softmax}_i(W\Phi(e) + b) \tag{6}$$

Here $\Phi(e_i) : R^d \mapsto R$ is the cosine similarity function between v_{e_i} and z_{word}.Note that Φ is applied element-wise to e. $W, b \in R$ are learnable weights. The softmax : $R^K \mapsto R^K$ transforms the outputs into normalized attention values. Then, entity-based representation of the context is computed as the attention-weighted average of all entity-vectors:

$$z_{entity} = \frac{1}{K} \sum_{i=1}^{K} a_{e_i} v_{e_i} \tag{7}$$

We then add z_{word} to z_{entity} to form a final representation vector of C. It is then fed into logistic regression to predict text label of C.

4 Experiments and Discussion

4.1 Sensitivity Analysis

We conduct experiments to evaluate the dependency on hyperparameters of each model. We tested potential hyperparameters including but not limited to the number of hidden units and layers, learning rate, weight decay and batch size. Different values are assigned to each hyperparameter, while setting the other ones default. See appendix A and B for the complete tables and figures.

4.1.1 Simple Graph Convolution

For sensitivity analysis on SGC, we will evaluate the performance of node classification on pre-processed Cora citation network datasets. Thus, we then select the valuables ones as regards to learning rate, weight decay, and epochs run for discussion. The magnitude of learning rate does not affect results tremendously. Train accuracy is invariant across all the experiments associated with learning rate, while validation accuracy slightly fluctuates. As for the total epochs to saturate model performance, it requires at least 80 epochs. Additionally, it is empirically found that epochs larger than 80 would not significantly influence on accuracy. Weight decay represents enforced regularization towards the model. The larger value of weight decay, the heavier the model is penalized. Our model is severely compromised along the size of weight decay, since we may have over-penalized it. Degree, which is defined as the dimension of propagation matrix used, poses a negative effect on model performance. It is considered overfitting once the degree of propagation matrix is excessively large.

4.1.2 Neural Attentative Bag-of-Entities

For NABoE, we conduct sensitivity analysis on AG News dataset. We use the hyperparameter patience instead of number of epochs. Patience is defined as the maximum acceptable number of epochs without improvement. No significant variation in accuracy is evident with respect to different values of patience.We observe that accuracy increases with embedding dimension until 5 and stays constant afterwards, because a simple model is sufficient to capture the patterns in AG News dataset. Also, dropout is used in the model, so it does not overfit even when embedding dimension is as large as 300. Accuracy stays constant with learning rate before 0.01 and gets worse afterwards. It is reasonable as a smaller learning rate allows the model to explore the solution space in more details. Accuracy generally decreases with weight decay. For batch size, patience and warm up epochs, we find accuracy independent of these parameters. Overall, performance of NABoE is not locally sensitive to the default hyper-parameters. Less than 2% change in accuracy is observed when a hyperparameter is multiplied or divided by a factor of 4.

4.2 Position Encoding

For NABoE, we test whether word positioning can help exploit more useful information during text classification. We add sinusoidal position encoding as described in [9] to the entity embeddings: For each context, the k^{th} entity embedding is transformed by $v'_{e_k} = v_{e_k} + PE(k)$ where $PE(k)_{2i} = sin(k/10000^{2i/d}), PE(k)_{2i+1} = cos(k/10000^{2i/d})$ and d is the embedding dimension. We do not add position encoding to word embeddings because weight of each word embedding is always equal.

Then, we compare accuracy of the new and original models, as shown in appendix C. Given Wikipedia pre-trained embeddings, position encoding model generally performs worse; for 20 newsgroup dataset we even observe an absolute decrement of more than 20% in accuracy. However, when embedding weights are learnable, the position encoding version slightly outperforms the original model. On 20 NG and R8 dataset, they respectively exhibit 0.63% and 0.48% absolute increment in accuracy.

4.3 Extension to New Dataset

Along the optimal set of hyperparameter values found in sensitivity analysis, we perform our empirical evaluation on AG News text documents dataset (120,000 training samples and 7,600 test samples). For SGC model, due to the limitation of memory, we were not able to train the model on the full AG News dataset, since for large dataset like AG News, we need an excessive memory to process the data and build a corresponding large graph. We instead trained our models on 20% (randomly sampled) out of the original dataset for both SGC and NABoE. Data fed into SGC are pre-processed in a way that both documents and words are treated as nodes in the graph. We also compare our training result with baseline models including GCN and XLNet [14] by using the publicly released implementations.

4.3.1 Performance and Efficiency

Comparing the test accuracy across all models, it is clear that SGC and NABoE have competitive performance, although the current state-of-the-art model XLNet still outperforms both of them.

From the results in Table 1, GCN performs slightly better than SGC in terms of test accuracy by 2%. The higher accuracy is atrributed to the fact that GCN contains more trainable parameters and can potentially capture more complex structure in the graph, but the effect of extra weights and non-linearity in layers is not significant in our experiment. As for training time, however, we find SGC outperforms GCN with training time 13 times faster, due to removed non-linearity and reduced extra weights in layers.

For NABoE model, learning the embedding weights from the new dataset performs much better than pretrained Wikipedia embeddings. This is contradictory to the original paper [11] where the pretrained embeddings were alleged to help achieve better performance on both R8 and 20Ng datasets.

Architecture	XLNet	GCN	SGC	NABoE-1*	NABoE-2
Accuracy(%)	95.55	89.76	87.37	79.89	87.09
Training Time	N/A	43.2s	3.4s	84s	15m12s
Trainable Weights	N/A	56238	10038	1204	15403207

Table 1. Model comparison in terms of accuracy, training time, and number of trainable parameters

5 Conclusion

In this project, we explore and investigate the underlying structure and mechanism of two models, SGC and NABoE, after which we applied them to text classification. SGC enables to capture the graphical (non-Euclidean) relationship within data, whereas NABoE uses attention mechanism to identify entities precisely and focuses on important components of the context. SGC is limited by the required amount of space for graph building and that is difficult to train with large datasets, yet its training takes much shorter time than NABoE. When extended to the unseen AG News dataset, SCC and NABoE both show competitive performance, with an accuracy around 87%. Sensitivity analysis on hyperparameters show similar characteristics for both models with respect to the change in weight decay, learning rate and number of epochs. However, SGC tends to overfit when degree is high, but NABoE seems tolerable to this issue with embedding dimension.

*NABoE-1 model uses pretrained word and entity embeddings from Wikipedia2Vec. NABoE-2 model has learnable embedding weights; embedding dimension is set to 300 to match the pretrained case.

References

[1] Kipf, T., Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. ArXiv, abs/1609.02907.

[2] Yao, L., Mao, C., Luo, Y. (2019). Graph convolutional networks for text classification. In 33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019 (pp. 7370-7377). (33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019). AAAI Press.

[3] Wang, H., Zhang, F., Zhang, M., Leskovec, J., Zhao, M., Li, W., Wang, Z. (2019). Knowledge Graph Convolutional Networks for Recommender Systems with Label Smoothness Regularization. ArXiv, abs/1905.04413.

[4] Wang, H., Lian, D., Ge, Y. (2019). Binarized Collaborative Filtering with Distilling Graph Convolutional Networks. IJCAI.

[5] Zhang, J., Shi, X., Zhao, S., King, I. (2019). STAR-GCN: Stacked and Reconstructed Graph Convolutional Networks for Recommender Systems. ArXiv, abs/1905.13129.

[6] Wang, Z., Lv, Q., Lan, X., Zhang, Y. (2018). Cross-lingual Knowledge Graph Alignment via Graph Convolutional Networks. EMNLP.

[7] Schlichtkrull, M., Kipf, T., Bloem, P., Berg, R.V., Titov, I., Welling, M. (2018). Modeling Relational Data with Graph Convolutional Networks. ESWC.

[8] Liu, Gang Guo, Jiabao. (2019). Bidirectional LSTM with attention mechanism and convolutional layer for text classification. Neurocomputing. 337.

[9] Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I. (2017). Attention is All you Need. ArXiv, abs/1706.03762.

[10] Nedelchev, R., Chaudhuri, D., Lehmann, J., Fischer, A. (2020). End-to-End Entity Linking and Disambiguation leveraging Word and Knowledge Graph Embeddings. ArXiv, abs/2002.11143.

[11] Yamada, I., Shindo, H. (2019). Neural Attentive Bag-of-Entities Model for Text Classification. CoNLL.

[12] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2018a. Wikipedia2Vec: An Optimized Tool for Learning Embeddings from Wikipedia. arXiv preprint arXiv:1812.06280v2.

[13] Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T. Weinberger, K.. (2019). Simplifying Graph Convolutional Networks. Proceedings of the 36th International Conference on Machine Learning, in Proceedings of Machine Learning Research 97:6861-6871 Available from http://proceedings.mlr.press/v97/wu19e.html.

[14] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., Le, Q.V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. NeurIPS.

[15] Peng Jin, Yue Zhang, Xingyuan Chen, and Yunqing Xia. 2016. Bag-of-embeddings for Text Classification. In Proceedings of the Twenty-Fifth Inter- national Joint Conference on Artificial Intelligence, pages 2824–2830.

Appendix

A. SGC Model Hyperparameter Tuning

learning rate	0.01	0.0575	0.105	0.1525	0.2
train acc	0.9071	0.9071	0.9071	0.9071	0.9071
validation acc	0.7580	0.7600	0.7640	**0.7660**	0.7620
weight decay	0	0.02	0.04	0.06	0.08
train acc	1.00	0.9143	0.9071	0.8929	0.8571
validation acc	**0.7620**	0.7540	0.7520	0.7300	0.6820
epochs	0	20	40	60	80
train acc	0.1786	0.7000	0.8929	0.9071	0.9000
validation acc	0.1020	0.4060	0.6740	**0.7540**	0.7540
degree	2	4	8	16	32
train acc	0.9071	0.9143	0.8714	0.8429	0.7500
validation acc	**0.7620**	0.7620	0.7560	0.7120	0.6520

B. NABoE Model Hyperparameter Tuning

embedding dim	2	5	20	100	300
validation acc	0.7797	0.8876	0.8848	**0.8902**	0.8891
weight decay	0	0.2	0.4	0.8	1.6
validation acc	**0.8885**	0.8857	0.8811	0.8848	0.8583
warm up epochs	0	2	4	8	16
validation acc	0.8871	0.8933	**0.8950**	0.8892	0.8892
learning rate	0.0001	0.0003	0.01	0.03	0.1
validation acc	0.8892	**0.8933**	0.8892	0.8775	0.8558
batch size	8	16	32	64	128
validation acc	0.8967	**0.9000**	0.8933	0.8933	0.8942
patience	3	7	10	15	30
validation acc	0.8885	**0.8887**	0.8848	0.8849	0.8848

C. Position Encoding Effect on the Accuracy of NABoE*

Dataset\Model	w/o PE, pre-trained*	w/o PE	PE, pre-trained	PE
20NG	0.868	0.844	0.6223	**0.8743**
R8	0.971	0.957	0.9324	**0.9758**
AG News	0.7797	**0.8876**	0.7783	0.8566

*The four versions of NABoE model, from left to right are: pretrained embeddings without position encoding, learned embeddings without position encoding, pretrained embeddings with position encoding, learned embeddings with position encoding.

D. More Sensitivity Test with SGC