RALIGRAPH at HASOC 2019: VGCN-BERT: Augmenting BERT with Graph Embedding for Offensive Language Detection

Zhibin Lu[1] and Jian-Yun Nie[2]

[1] University of Montreal zhibin.lu@umontreal.ca
[2] University of Montreal nie@iro.umontreal.ca
http://rali.iro.umontreal.ca/rali/?q=en

Abstract. Hate speech and offensive language detection are receiving more and more attention in recent years. The RALIGRAPH team participated in the Shared Task on the Identification of Offensive content for Indo-European languages within the FIRE conference. This paper describes our approach VGCN-BERT model for all three sub-tasks of hate language and offensive language detection in English. VGCN-BERT takes into account both local and global information, by combining the Graph Convolutional Networks (GCN) and the Self-Attention Encoder (BERT). Our approach produced good results in the experiments.

Keywords: Offensive Language Detection · Graph Convolutional Networks · Graph Embedding · Self-Attention Encoder.

1 Introduction

In recent years, an increasing number of users are subjected to offensive languages or have witnessed abusive and hateful texts online, which is related to sexism, racism or other types of aggressive behaviors and cyberbullying. Governments have started to enact laws, and major social platforms such as Facebook and Twitter are also censoring offensive posts with the assistance of artificial intelligence technologies, human reviewing processes, user reporting processes, and so on. However, it seems the problem is still far from being successfully resolved.

A lot of research methods for offensive language detection have been proposed in the past few years [10]. Among which, deep learning (DL) based methods [2] are attracting more and more research interests [6], [1]. Most of the existing DL methods are based on convolutional neural networks (CNN) [8] and/or recurrent neural networks (RNN) such as long short-term memory (LSTM) [7]. Self-attention [12] technology has been widely used in many NLP tasks in recent

years. BERT [4] is currently one of the most powerful self-attention models that uses multi-task pre-training technique based on a large number of corpora. It often achieves the best performance in many tasks such as Named entity Recognition (NER), text classification and reading comprehension.

However, the existing deep learning models may capture well semantic and syntactic information in local contexts, but may ignore global information such as word co-occurrences in a corpus which carries non-consecutive and long-distance dependencies [11].

In our research on the task of classifying offensive language, we intend to leverage both the local information captured in BERT and the global information on the whole vocabulary in a language. To obtain global information, we use Graph Convolutional Networks (GCN) [9] which can perform convolution calculations on neighbor nodes in the graph. In this paper, we take inspiration from Text GCN [13] to build a vocabulary graph, and construct the graph convolutional networks as an embedding layer which brings global information to the entire sentence, then feed the vocabulary graph embedding and word embedding together to the Self-attention encoder in the BERT. We call it VGCN-BERT model. We participated in the HASOC 2019 [10] competition with this model and got third place on Task B and C.

In section two, we describe some related work, such as self-attention mechanisms, BERT model and graph convolutional networks. In section three, we describe our approach, including the structure and training settings of VGCN-BERT model, and the method of datset preprocessing. In section four, we list the results and make some comments.

2 Related Work

2.1 Self-Attention and BERT

Self-Attention [12] uses three weights of Q(Query), K(Key), and V(Value), and calculates the relation between each word and all other words, and obtains different attention scores as follows

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V, \tag{1}$$

where the denominator is a scaling factor used to control the scale of the attention score, d_k is the dimension of the query and key vectors. Using these attention scores, every word can get a weighted vector representation which captures contextual information.

The structure of BERT [4] for text classification is divided into two major parts: the positional word embedding part, and the multi-layer multi-head self-attention encoder [12] part. For every layer of encoder, it contains a multi-head attention (a layer-superimposed self-attention mechanism), which reads the input data and uses the multi-head attention to get a new representation of the context information for each word.

BERT is a pre-trained model. The goal of pre-training is to provide a good initialization for model training, which has been widely used in image classification and NLP. BERT is trained on 800M words from BooksCorpus and 2,500M words from English Wikipedia, and uses two unsupervised task to improves the pre-training:

- Masked Language Model. Instead of the traditional n-gram language model, BERT randomly selects words to mask them out and then tries to predict the masked words from their context.
- Next Sentence Prediction. BERT uses sentence classification as a pre-training task to determine if a sentence is the real next sentence or another randomly picked sentence.

A typical input to BERT is a pair of sentences as follows [4]:

Input = [CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]

Label = IsNext

where [CLS] is the BOS (begin of sentence) for a pair of sentences; [SEP] is used as a separator of two sentences or as EOS (end of sentence); [MASK] is used to mask out the predicted words in the masked language model. The final state of [CLS] embedding after encoding is used as the aggregated sequence representation for classification tasks. In our offensive language detection task, we only need to classify every individual document, which is usually as short as a tweet, with only one or two sentences. So we treat the document as a sentence. Below is an example of the sentence classification:

Input = [CLS] Stupid ass coward b*tch f*ggot racist piece of sh*t. [SEP]

Label = Offensive

2.2 Graph Convolutional Networks (GCN)

A GCN [9] is a multilayer neural network that calculates directly on a graph and induces embedding vectors of nodes based on properties of their neighborhoods. Formally, consider a graph $G = (P, E)$ [3], where P ($|P| = n$) and E are sets of nodes and edges, respectively. In GCN, the graph is a self-loop graph, where every node is assumed to be connected to itself, i.e., $(p, p) \in E$ for any p.

Usually, people use the adjacency matrix A and its degree matrix D to represent graph G, where $D_{ii} = \sum_j A_{ij}$. The diagonal elements of A are set to 1 because of self-loops. For one convolutional layer, the formula is

$$H = \tilde{A}XW, \tag{2}$$

where $X \in \mathbb{R}^{n \times m}$, n is the number of nodes, m is the dimension of the feature, $\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the normalized symmetric adjacency matrix and $W \in \mathbb{R}^{m \times h}$ is a weight matrix.

[3] In order to distinguish from notations $(v, V, |V|)$ of vocabulary, this paper uses notations $(p, P, |P|)$ to represent the point(vertex) of the graph.

Usually, we use two layers of GCN to capture information about direct and indirect neighbors [9]. Then a two-layers GCN model is as follows,

$$Z = \text{softmax}(\tilde{A}\ \text{ReLU}(\tilde{A}XW_h)W_c), \tag{3}$$

where ReLU is an activation function, W_h is the hidden layer weight and W_c is the output layer weight.

Text GCN [13] is a special case of GCN, in which the vocabulary and all documents correspond to nodes in a graph. The edges between words are determined by point-wise mutual information (PMI) and those between a word and a document by TF-IDF. After the phase of building the graph, Text GCN uses the same Formula 3 during the training phase, but the feature matrix X is an identity matrix, which means every word or document is represented as a one-hot vector as the input to Text GCN.

3 Methodology

3.1 VGCN-BERT model

The idea we explore in our work is to combine the graph's global information-awareness capabilities (global receptive fields) with the self-attention local information-awareness capabilities (local receptive fields). Specifically, we want each document to get some global information about the vocabulary through a vocabulary graph before doing self-attention training. On the other hand, BERT is able to take into account local dependencies between words in a sentence. Our approach combines GCN with BERT.

For vocabulary graph, we take inspiration from Text GCN [13] to build a vocabulary graph, and use NPMI [3] to calculate the vocabulary graph as follows:

$$\text{NPMI}(i,j) = -\frac{1}{\log p(i,j)} \log \frac{p(i,j)}{p(i)p(j)} \tag{4}$$

After building the vocabulary graph, we construct the vocabulary graph embedding module and insert it into BERT in the word embedding phase. To obtain the vocabulary graph embedding, we modify the formula 2 to the following form:

$$G_{embedding} = X\tilde{A}W, \tag{5}$$

where X is the word embedding matrix which comes from word embedding layer of BERT, and $\tilde{A}$ is the normalized symmetric adjacency matrix of vocabulary graph, and the output dimension of W controls the number of vocabulary graph embedding whose dimension is the same as every word embedding.

Therefore, the original sentence represented by word embeddings will be enriched by the vocabulary graph embedding. Then we feed all the embedding vectors to the self-attention encoder, and the encoder module will pay attention to all word embeddings as well as graph embeddings. Finally, we use the embedding produced for [CLS] for classification.

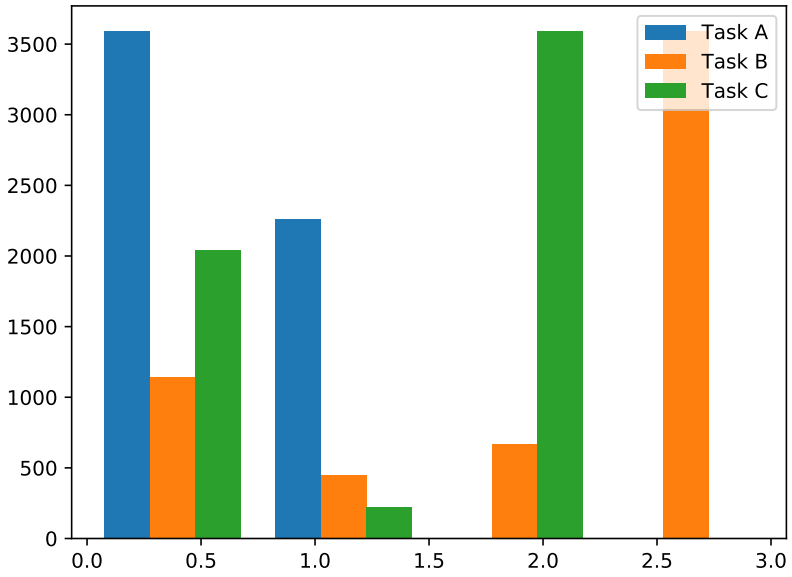

Fig. 1. Data distribution

3.2 Datasets

The training dataset of HASOC has a total of 5852 texts, and its labels are divided into three sub-tasks. Figure 1 shows the data distribution on the training set and find that the categories of the dataset are unbalanced, especially Task B and Task C.

Therefore we use the weighted cross entropy as loss function. The weight of each of the classes (W_c) is calculated by

$$W_c = \frac{\#dataset}{\#classes \cdot \#one\_class}, \tag{6}$$

where $\#dataset$ is the total size of dataset and $\#classes$ is the number of classes and $\#one\_class$ is the count of one class.

In addition to the dataset provided by HASOC, for sub-task A, we also used **Founta et al.'s dataset** [5] [4] to pre-train the model. Founta et al. constructed this large dataset in order to reflect a real-world setting where abuse is relatively rare. It contain 99,996 [5] tweets with cross-validated labels and is classified into 4 labels, normal (53,851), spam (14,030), hateful (27,150) and abusive (4,965). We

[4] https://github.com/ENCASEH2020/hatespeech-twitter
[5] The final version provided by the author is more than the one described in the paper.

merge the labels of normal and spam into label NOT of sub-task A and merge the labels of hateful and abusive into label HOF of sub-task A.

3.3 Preprocessing and setting

To clean the text, we removed URL string and @-mentions for every text, then the text was lower-cased and tokenized using NLTK's *TweetTokenizer* [6]. We use bertTokenizer function to split text, so that the vocabulary for GCN is always a subset of pre-trained BERT's vocabulary.

To build the vocabulary graph, we set the window size as 20 when calculating NPMI, and set the threshold of NPMI as 0.2 to filter out non meaningful relationships between words.

In the VGCN-BERT model, the graph embedding output size is set as 16, and the hidden dimension of graph embedding as 128. We use the *bert-base-uncased* version of pre-trained BERT, and set the max sequence length as 200. The model is then trained in 15 epoch-es with a dropout rate of 0.2.

The following are other parameter settings for the three tasks.

- **Sub-task A**: While using Founta et al.'s dataset to pre-train the model, we set learning rate as 4e-6 and mini batch size as 12 and L_2 loss weight decay as 1e-4. While fine-tuning the model using dataset of sub-task A, we set learning rate as 1.9e-6 and mini batch size as 16 and L_2 loss weight decay as 0.02.
- **Sub-task B**: we set learning rate as 4e-5, and L_2 loss weight decay as 0.07.
- **Sub-task C**: we set learning rate as 1.4e-5, and L_2 loss weight decay as 0.05.

We use the original BERT model and a 2-layer MLP model as the two baselines. The parameter settings of BERT are the same as the VGCN-BERT for all three tasks, except that there is no graph embedding output dimension. We also use Founta et al.'s dataset to pre-train the BERT model for sub-task A. For the 2 hidden layers MLP, we use the term-frequency (TF) as inputs feature, and we set the first hidden layer dimension as 512 and the second hidden layer dimension as 100, learning rate as 1.5e-3 and L_2 loss weight decay as 2e-5, batch size as 64, total train epoch as 100, early stopping as 10.

4 Result

Before getting the test set from HASOC, we take out small parts of the training set as validation set for model selection. The shuffled training set are divided into three sets with a ratio of 80:5:15, which represent the training set, the valid1 set, and the valid2 set, respectively. Table 1 lists the f1-score performance of the valid2 set when the performance of the valid1 set gets the best. It should be noted that we can't ensure that such parameter settings are optimal for all three

[6] http://www.nltk.org/api/nltk.tokenize.html

Table 1. Weighted average F1-Score on valid 1/2 set.

Model	**Task A**	**Task B**	**Task C**
	valid 1/valid 2	valid 1/valid 2	valid 1/valid 2
MLP	**68.21**/64.00	60.12/60.62	66.61/64.58
BERT	64.29/69.61	60.81/**61.92**	68.15/65.85
VGCN-BERT	64.82/**70.33**	**66.13**/61.26	**70.08/66.06**

Table 2. Results of Macro F1 and Weighted F1 on Test set.

Team	**Task A**	**Task B**	**Task C**
	macro/weighted	macro/weighted	macro/weighted
Top Team	**78.82/83.95**	**54.46/72.77**	**51.11**/75.63
RALIGRAPH	74.09/78.76	47.90/72.18	49.07/**77.19**

models, because we found that the parameters of learning rate and $L2$ decay have a great influence on the final performance of the models, and sometimes small changes of them will cause large fluctuations. We believe the small dataset and the uneven data distribution are the main causes. Despite this, we find that in most cases, the performance of VGCN-BERT is better than the other two models, especially on task C.

Table 2 lists the results on test set provided by the HASOC. Our team got third place on Task B and C and scored the second highest weighted F1-score on the Task C.

5 Conclusions

In this paper, we report the work we performed in the FIRE hate language detection tasks. Our main idea is to complement the local information captured by BERT with global information on the vocabulary. The combination of BERT with GCN seems to produce good results, and our submissions are ranked quite high.

As we do not have much training data specific for the tasks, the model can be further improved with more training data. In addition, as future work, we will also consider using different relationship measurements to construct various vocabulary graphs. Also, more layers of GCN networks needs to be explored to get a sense of the influences of the network structures when introducing BERT embeddings into VGCN.

References

1. Agrawal, S., Awekar, A.: Deep learning for detecting cyberbullying across multiple social media platforms. In: Advances in Information Retrieval - 40th European

Conference on IR Research, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings. pp. 141–153 (2018). https://doi.org/10.1007/978-3-319-76941-7_11, https://doi.org/10.1007/978-3-319-76941-7_11

2. Badjatiya, P., Gupta, S., Gupta, M., Varma, V.: Deep learning for hate speech detection in tweets. In: Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia, April 3-7, 2017. pp. 759–760 (2017). https://doi.org/10.1145/3041021.3054223, https://doi.org/10.1145/3041021.3054223
3. Bouma, G.: Normalized (pointwise) mutual information in collocation extraction. Proceedings of the Biennial GSCL Conference 2009, University of Potsdam (2009), https://pdfs.semanticscholar.org/1521/8d9c029cbb903ae7c729b2c644c24994c201.pdf
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
5. Founta, A.M., Djouvas, C., Chatzakou, D., Leontiadis, I., Blackburn, J., Stringhini, G., Vakali, A., Sirivianos, M., Kourtellis, N.: Large scale crowdsourcing and characterization of twitter abusive behavior. In: 11th International Conference on Web and Social Media, ICWSM 2018. AAAI Press (2018)
6. Gambäck, B., Sikdar, U.K.: Using convolutional neural networks to classify hate-speech. In: Proceedings of the First Workshop on Abusive Language Online, ALW@ACL 2017, Vancouver, BC, Canada, August 4, 2017. pp. 85–90 (2017), https://www.aclweb.org/anthology/W17-3013/
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
8. Kim, Y.: Convolutional neural networks for sentence classification. In: EMNLP. pp. 1746–1751 (2014)
9. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
10. Modha, S., Mandl, T., Majumder, P., Patel, D.: Overview of the HASOC track at FIRE 2019: Hate Speech and Offensive Content Identification in Indo-European Languages. In: Proceedings of the 11th annual meeting of the Forum for Information Retrieval Evaluation (December 2019)
11. Peng, H., Li, J., He, Y., Liu, Y., Bao, M., Wang, L., Song, Y., Yang, Q.: Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In: WWW. pp. 1063–1072 (2018)
12. Vaswani, A., etc, N.S.: Attention Is All You Need. Long Beach
13. Yao, L., Mao, C., Luo, Y.: Graph convolutional networks for text classification. In: AAAI (2019)