VGCN-BERT: Augmenting BERT with Graph Embedding for Text Classification

Zhibin Lu(✉), Pan Du, and Jian-Yun Nie

University of Montreal, Montreal, Canada
{zhibin.lu,pan.du}@umontreal.ca, nie@iro.umontreal.ca

Abstract. Much progress has been made recently on text classification with methods based on neural networks. In particular, models using attention mechanism such as BERT have shown to have the capability of capturing the contextual information within a sentence or document. However, their ability of capturing the global information about the vocabulary of a language is more limited. This latter is the strength of Graph Convolutional Networks (GCN). In this paper, we propose VGCN-BERT model which combines the capability of BERT with a Vocabulary Graph Convolutional Network (VGCN). Local information and global information interact through different layers of BERT, allowing them to influence mutually and to build together a final representation for classification. In our experiments on several text classification datasets, our approach outperforms BERT and GCN alone, and achieve higher effectiveness than that reported in previous studies.

Keywords: Text classification · BERT · Graph Convolutional Networks

1 Introduction

Text classification is a fundamental problem in natural language processing (NLP) and has been extensively studied in many real applications. In recent years, we witnessed the emergence of text classification models based on neural networks such as convolutional neural networks (CNN) [15], recurrent neural networks (RNN) [13], and various models based on attention [27]. BERT [8] is one of the self-attention models that uses multi-task pre-training technique based on large corpora. It often achieves excellent performance, compared to CNN/RNN models and traditional models, in many tasks [8] such as Named-entity Recognition (NER), text classification and reading comprehension.

The deep learning models excel by embedding both semantic and syntactic information in a learned representation. However, most of them are known to be limited in encoding long-range dependency information of the text [2]. The utilization of self-attention helps alleviate this problem, but the problem still remains. The problem stems from the fact that the representation is generated

J. M. Jose et al. (Eds.): ECIR 2020, LNCS 12035, pp. 369–382, 2020.
https://doi.org/10.1007/978-3-030-45439-5_25

from a sentence or a document only, without taking into account explicitly the knowledge about the language (vocabulary). For example, in the movie review below:

> *"Although it's a bit smug and repetitive, this documentary engages your brain in a way few current films do."*

both negative and positive opinions appear in the sentence. Yet the positive attitude *"a way few current films do"* expresses a very strong opinion of the innovative nature of the movie in an implicit way. Without connecting this expression more explicitly to the meaning of *"innovation"* in the context of movie review comments, the classifier may underweight this strong opinion and the sentence may be wrongly classified to be negative. On this example, self-attention that connects the expression to other tokens in the sentence may not help.

In recent studies, approaches have also been developed to take into account the global information between words and concepts. The most representative work is Graph Convolutional Networks (GCN) [16] and its variant Text GCN [32], in which words in a language are connected in a graph. By performing convolution operations on neighbor nodes in the graph, the representation of a word will incorporate those of the neighbors, allowing to integrate the global context of a domain-specific language to some extent. For example, the meaning of *"new"* can be related to that of *"innovation"* and *"surprised"* through the connections between them. However, GCNs that only take into account the global vocabulary information may fail to capture local information (such as word order), which is very important in understanding the meaning of a sentence. This is shown in the following examples, where the position of *"work"* in the sentence will change the meaning depending on its context:

- *"skip work to see it at the first opportunity."*
- *"skip to see it, work at the first opportunity."*

In this paper, inspired by GCN [16,32] and self-attention mechanism in BERT, we propose to combine the strengths of both mechanisms in the same model. We first construct a graph convolutional network on the vocabulary graph based on the word co-occurrence information, which aims at encoding the global information of the language, then feed the graph embedding and word embedding together to a self-attention encoder in BERT. The word embedding and graph embedding then interact with each other through the self-attention mechanism while learning the classifier. This way, the classifier can not only make use of both local information and global information, but also allow them to guide each other via the attention mechanism so that the final representation built up for classification will integrate gradually both local and global information. We also expect that the connections between words in the initial vocabulary graph can be spread to more complex expressions in the sentence through the layers of self-attention.

We call the proposed model VGCN-BERT. Our source code is available at https://github.com/Louis-udm/VGCN-BERT.

We carry out experiments on 5 datasets of different text classification tasks (sentiment analysis, grammaticality detection, and hate speech detection). On all these datasets, our approach is shown to outperform BERT and GCN alone.

The contribution of this work is twofold:

- *Combining global and local information*: There has not been much work trying to combine local information captured by BERT and global information of a language. We demonstrate that their combination is beneficial.
- *Interaction between local and global information through attention mechanism*: We propose a tight integration of local information and global information, allowing them to interact through different layers of networks.

2 Related Work

2.1 Self-attention and BERT

As aforementioned, attention mechanisms [28,31] based on various deep neural networks, in particular the self-attention mechanism proposed by Vaswan et al. [27], have greatly improved the performance in text classification tasks. The representation of a word acquired through self-attention can incorporate the relationship between the word and all other words in a sentence by fusing the representations of the latter.

BERT (Bidirectional Encoder Representations from Transformers) [8], which leverages a multi-layer multi-head self-attention (called transformer) together with a positional word embedding, is one of the most successful deep neural network model for text classification in the past years. The attention mechanism in each layer of the encoder enhances the new representation of the input data with contextual information by paying multi-head attentions to different parts of the text. A pre-trained BERT model based on 800M words from BooksCorpus and 2,500M words from English Wikipedia is made available. It has also been widely used in many NLP tasks, and has proven effective. However, as most of other attention-based deep neural networks, BERT mainly focuses on local consecutive word sequences, which provides local context information. That is, a word is placed in its context, and this generates a contextualized representation. However, it may be difficult for BERT to account for the global information of a language.

2.2 Graph Convolutional Networks (GCN)

Global relations between words in a language can be represented as a graph, in which words are nodes and edges are relations. Graph Neural Network (GNN) [2,5] based on such a graph is good at capturing the general knowledge about the words in a language. A number of variants of GNN have been proposed and applied to text classification tasks [7,12,16,21,33], of which Kipf et al. [16] creatively presented Graph Convolutional networks (GCN) based on spectral graph theory. GCN first builds a symmetric adjacency matrix based on

a given relationship graph (such as a paper citation relationship), and then the representation of each node is fused according to the neighbors and corresponding relationships in the graph during the convolution operation.

Text GCN is a special case of GCN for text classification proposed by Yao et al. [32] recently. Different from general GCN, it is based on a heterogeneous graph where both words and documents are nodes. The relationships among nodes, however, are measured in three different ways, which are co-occurrence relations among words, tf-idf measure between documents and words, and self similarity among documents. In terms of convolution, Text GCN uses the same algorithm as GCN. GCN and its variants are good at convolving the global information in the graph into a sentence, but they do not take into account local information such as the order between words. When word order and other local information are important, GCN may be insufficient. Therefore, it is natural to combine GCN with a model capturing local information such as BERT.

2.3 Existing Combinations of GCN and BERT

Some recent studies have combined GCN with BERT. Shang et al. [23] applied a combination to the medication recommendation task, which predict a medical code given the electronic health records (EHR), i.e., a sequence of historical medical codes, of a patient. They first embed the medical codes from a medical ontology using Graph Attention Networks (GAT), then feed the embedding sequence of the medical code in an EHR into BERT for code prediction. Nevertheless, the order in the code sequence is discarded in the transformer since it is not applicable in their scenario, making it incapable of capturing all the local information as in our text classification tasks.

Jong et al. [14] proposed another combination to the citation recommendation task using paper citation graphs. This model simply concatenates the output of GCN and the output of BERT for downstream predictive tasks. We believe that interactions between the local and global information are important and can benefit the downstream prediction tasks. In fact, through layers of interactions, one could allow the information captured in GCN be applied to the input text, and the representation of the input text be spread over GCN. This will produce the effect we illustrated in the earlier example of movie review (*a way few current films do* vs. *innovation*). This is the approach we propose in this paper.

One may question about the necessity to explicitly use graph embedding to cope with global dependency information, as some studies [18,25] have shown that word embedding trained on a corpus, such as Word2Vec [19], GloVe [22], FastText [10], can capture some global connections between words in a language. We believe that a vocabulary graph can still provide additional information given the fact that the connections between words observed in word embeddings are limited within a small text window (usually 5 words). Long-range connections are missing. In addition, by building a vocabulary graph on an application-specific document collection, one can capture application-dependent dependencies, in addition to the general dependencies in the pre-trained models.

3 Proposed Method

The global language information can take multiple forms. In this paper, we consider lexical relations in a language, i.e. a vocabulary graph. Any vocabulary graph can be used to complement BERT (e.g. Wordnet). In this paper, we consider a graph constructed using word co-occurrences with documents. Local information from a text is captured by BERT. The interaction between them is achieved by first selecting the relevant part of the global vocabulary graph according to the input sentence and transforming it into an embedding representation. We use multiple layers of attention mechanism on concatenated representation of input text and the graph. These processes are illustrated in Fig. 1. We will provide more details in the following subsections.

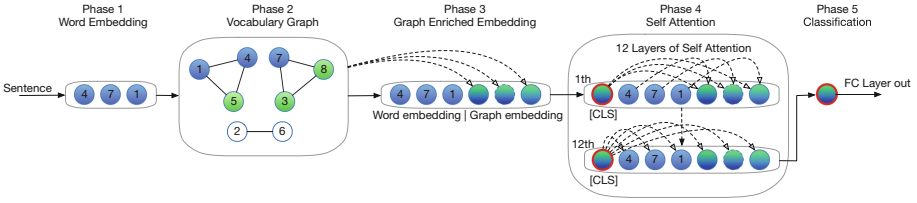

Fig. 1. Illustration of VGCN-BERT. The embeddings of input sentence (Phase 1) are combined with the vocabulary graph (Phase 2) to produce a graph embedding, which is concatenated to the input sentence (Phase 3). Note that from the vocabulary graph, only the part relevant to the input is extracted and embedded. In Phase 4, several layers of self-attention are applied to the concatenated representation, allowing interactions between word embeddings and graph embedding. The final embedding at the last layer is fed in a fully connected layer (Phase 5) for classification.

3.1 Vocabulary Graph

Our vocabulary graph is constructed using normalized point-wise mutual information (NPMI) [3], as shown in Eq. 1:

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

where i and j are words, $p(i,j) = \frac{\#W(i,j)}{\#W}$, $p(i) = \frac{\#W(i)}{\#W}$, $\#W(*)$ is the number of sliding windows containing a word or a pair of words, and $\#W$ is the total number of sliding windows. To obtain long-range dependency, we set the window to the whole sentence. The range of value of NPMI is [-1,1]. A positive NPMI value implies a high semantic correlation between words, while a negative NPMI value indicates little or no semantic correlation. In our approach, we create an edge between two words if their NPMI is larger than a threshold. Our experiments show that the performance is better when the threshold is between 0.0 and 0.3.

3.2 Vocabulary GCN

A general GCN [16] is a multi-layer (usually 2 layers) neural network that convolves directly on a graph and induces embedding vectors of nodes based on properties of their neighborhoods. Formally, consider a graph $G = (P, E)$[1], where P (with $|P| = n$) and E are sets of nodes and edges, respectively. For a single convolutional layer of GCN, the new representation is calculated as follows:

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

where $X \in \mathbb{R}^{n \times m}$ is the input matrix with n nodes and m dimensions of the feature, $W \in \mathbb{R}^{m \times h}$ is a weight matrix, $\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the normalized symmetric adjacency matrix, where $D_{ii} = \sum_j A_{ij}$. The normalization operation for A is to avoid numerical instabilities and exploding/vanishing gradients when used in a deep neural network model [16].

The graph nodes of GCN are "task entities" such as documents that need to be classified. It requires all entities, including those from training set, validation set, and test set, to be presented in the graph, so that no node representation is missing in downstream tasks. This limits the application of GCN in many predictive tasks, where the test data is unseen during the training process.

In our case, we aim to convolve the related words instead of the documents in the corpus for classification. Therefore, the graph of our proposed GCN is constructed on the vocabulary instead of the documents. Thus, for a single document, assuming the document is a row vector $\boldsymbol{x}$ consisting of words in the vocabulary, a layer of convolution is defined in Eq. 3:

$$\boldsymbol{h} = (\tilde{A}\boldsymbol{x}^T)^T W = \boldsymbol{x}\tilde{A}W, \tag{3}$$

where $\tilde{A}^T = \tilde{A}$ represent the vocabulary graph. $\boldsymbol{x}\tilde{A}$ extracts the part of vocabulary graph relevant to the input sentence $\boldsymbol{x}$. W holds the weights of the hidden state vector for the single document, with dimension $|V| \times h$. Then for m documents in a mini-batch, the one-layer graph convolution in Eq. 3 becomes:

$$H = X\tilde{A}W, \tag{4}$$

and the corresponding 2-layer Vocabulary GCN with ReLU function is as follows:

$$\mathbf{VGCN} = \mathrm{ReLU}(X_{mv}\tilde{A}_{vv}W_{vh})W_{hc}, \tag{5}$$

where m is the mini-batch size, v is the vocabulary size, h is the hidden layer size, c the class size or sentence embedding size. Every row of X_{mv} is a vector containing document features, which can be a bag-of-words vector, or word embedding of BERT. The above equation aims to produce a layer of convolution of the graph, which captures the part of the graph relevant to the input (through $X_{mv}\tilde{A}_{vv}$), then performs 2 layers of convolution, combining words from input sentence with their related words in vocabulary graph.

[1] 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.

3.3 Integrating VGCN into BERT

When BERT is applied to text classification, a typical solution contains three parts. The first part is the word embedding module with the position information of the word; the second part is the transformer module using multi-layer multi-head self-attention stacking; and the third part is the fully connected layer using the output sentence embedding for classification.

Self-attention operates with a query Q against a key K and value V pair. The attention score is calculated as follows:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{6}$$

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 encoding the contextual information.

Instead of using only word embeddings of the input sentence in BERT, we feed both the vocabulary graph embedding obtained by Eq. 5 and the sequence of word embeddings to BERT transformer. This way, not only the order of the words in the sentence is retained, but also the background information obtained by VGCN is utilized. The overall VGCN-BERT model is schematically illustrated in Fig. 1. Through the attention score calculated by Eq. 6, local embedding and global embedding are fully integrated after layer-by-layer interaction in 12-layer and 12-heads self-attention encoder. The corresponding VGCN can then be formulated as:

$$\mathbf{G_{embedding}} = \text{ReLU}(X_{mev}\tilde{A}_{vv}W_{vh})W_{hg}, \tag{7}$$

where W_{hg}, which was originally used for classification, becomes the output of size g of graph embedding (hyperparameter) whose dimension is the same as every word embedding; m is the size of the mini-batch; e is the dimension of word embedding, and v is the vocabulary size.

4 Experiment

We evaluate VGCN-BERT and compare it with baseline models on 5 datasets to verify whether our model can leverage both local and global information.

4.1 Baselines

In addition to the original BERT model, we also use several other neural network models as baselines.

– **MLP:** Multilayer perceptron with 2 hidden layers (512 and 100 nodes), and bag-of-words model with TF weighting.

- **Bi-LSTM** [11]**:** The BERT's pre-trained word embeddings are used as input to the Bi-LSTM model.
- **Text GCN:** The original Text GCN model uses the same input feature as MLP model, and we use the same training parameters as in [32].
- **VGCN:** This model only uses VGCN, corresponding to Eq. 7, but the output dimension becomes the class size. BERT's pre-trained word embeddings are used as input. The output of VGCN is relayed to a fully connected layer with Softmax function to produce the classification score. This model only uses the global information from vocabulary graph.
- **BERT:** We use the small version (Bert-base-uncased) pre-trained BERT [8].
- **Vanilla-VGCN-BERT:** Vanilla combination of BERT and VGCN is similar to [14], which produces two separate representations through BERT and GCN, and then concatenates them. ReLU and a fully connected layer are applied to the combined representation for classification. The main difference of this model with ours is that it does not allow interactions between the input text and the graph.

4.2 Datasets

We ran our experiments on the following five datasets:

- **SST-2.** The Stanford Sentiment Treebank is a binary single-sentence classification task consisting of sentences extracted from movie reviews with human annotations of their sentiment [24]. We use the public version[2] which contains 6,920 examples in training set, 872 in validation set and 1,821 in test set, for a total of 4,963 positive reviews and 4,650 negative reviews. The average length of reviews is 19.3 words.
- **MR** is also a movie review dataset for binary sentiment classification, in which each review only contains one sentence [20][3]. We used the public version in [26][4]. It contains 5,331 positive and 5,331 negative reviews. The average length is 21.0 words.
- **CoLA.** The Corpus of Linguistic Acceptability is a binary single-sentence classification task. CoLA is manually annotated for acceptability (grammaticality) [29]. We use the public version which contains 8,551 training data and 1,043 ation data[5], for a total of 6,744 positive and 2,850 negative cases. The average length is 7.7 words. Since we do not have the label for the test set, we split 5% of the training set as validation set and use the original validation set as the test set.
- **ArangoHate** [1] is a resampled dataset merging the datasets from [30] and [6]. It contains 2,920 hateful documents and 4,086 normal documents. The average length is 13.3 words. Since the dataset is not pre-divided into training, validation and test sets, we randomly split it into three sets at the ratio of 85:5:10.

[2] https://github.com/kodenii/BERT-SST2.
[3] http://www.cs.cornell.edu/people/pabo/movie-review-data/.
[4] https://github.com/mnqu/PTE/tree/master/data/mr.
[5] https://github.com/nyu-mll/GLUE-baselines.

– **FountaHate** is a large four-label dataset for hate speech and offensive language detection [9]. It contains 99,996[6] tweets with cross-validated labels and is classified into 4 labels: normal (53,851), spam (14,030), hateful (27,150) and abusive (4,965). The average length is 15.7 words. Since the dataset is not pre-divided into training, validation and test sets, we split it into three sets at the ratio of 85:5:10 after shuffle.

4.3 Preprocessing and Setting

We removed URL strings and @-mentions to retain the text content, then the text was lower-cased and tokenized using NLTK's *TweetTokenizer*[7]. We use BERTTokenizer function to split text, so that the vocabulary for GCN is always a subset of pre-trained BERT's vocabulary. When computing NPMI on a dataset, the whole sentence is used as the text window to build the vocabulary graph. The threshold of NPMI is set as 0.2 for all datasets 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 9 epochs with a dropout rate of 0.2. The following are other parameter settings for different datasets.

– SST-2: mini-batch = 16, learning rate = 1e−5, L_2 loss weight decay = 0.01.
– CoLA and MR: mini-batch = 16, learn. rate = 8e−6, L_2 loss decay = 0.01.
– ArangoHate: mini-batch = 16, learn. rate = 1e−5, and L_2 loss decay = 1e−3.
– FountaHate: mini-batch = 12, learn. rate = 4e−6, and L_2 loss decay = 2e−4.

These parameters are set based on our preliminary tests. We also use the default fine-tuning learning rate and L_2 loss weight decay as in [8]. The baseline methods are set with the same parameters as in the original papers.

4.4 Loss Function

We use the cross-entropy as the loss function for all models, except for FountaHate dataset where we use the mean squared error as the loss function in order to leverage the annotators' voting information.

We use Adam as training optimizer for all models. For cases where the label distributions are uneven (CoLA (2.4:1), ArangoHate (1.4:1) and FountaHate (10.9:5.5:2.8:1)), *comput_class_weight* function[8] from scikit-learn [4] is used as the weighted loss function. The weight of each the classes (W_c) is calculated by

$$W_{classes} = \frac{\#dataset}{\#classes \cdot \#every\_class}, \tag{8}$$

[6] The final version provided by the author is more than the one described in the paper.
[7] http://www.nltk.org/api/nltk.tokenize.html.
[8] https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html.

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

4.5 Evaluation Metrics

We adopt the two most widely used metrics to evaluate the performance of the classifiers - the weighted average F1-score, and the macro F1-score [17].

$$\textbf{Weighted avg F1} = \sum_{i=1}^{C} F1_{c_i} * W_{c_i}, \qquad \textbf{Macro F1} = \frac{1}{C} \sum_{i=1}^{C} F1_{c_i} \tag{9}$$

4.6 Experimental Result

The main results on weighted average F1-Score and macro F1-Score on test sets are presented in Table 1. The main observation is that VGCN-BERT outperforms all the baseline models (except against Vanilla-VGCN-BERT on MR dataset). In particular, it outperforms both VGCN and BERT alone, confirming the advantage to combine them.

Among the models that only use local information, we see that BERT outperforms MLP, Bi-LSTM. Between the models that exploit a vocabulary graph - VGCN and Text-GCN, the performance is similar.

Vanilla-VGCN-BERT and VGCN-BERT are two models that combine local and global information. In general, these models perform better than the other baseline models. This result confirms the benefit of combining local information and global information.

Comparing VGCN-BERT with Vanilla-VGCN-BERT, we see that the former generally performs better. The difference is due to the interactions between local and global information. The superior performance of VGCN-BERT clearly shows the benefit of allowing interactions between the two types of information.

4.7 Visualization

To better understand the behaviors of BERT, and its combination with VGCN, we visualize the attention distribution of the [CLS] token in the self-attention module of BERT, VGCN-BERT and Vanilla-VGCN-BERT models. As the vocabulary graph is embedded into vectors of 16 dimensions, it is not obvious to show what meaning corresponds to each dimension. To facilitate our understanding, we show the top two words from the sub-graph related to the input sentence, which are strongly connected to each of the 16 dimensions of graph embedding. More specifically, w each word embedding of a document is input to Eq. 7, we only need to broadcast the result of XA and element-multiply it by W to obtain the representation value of the words involved. The equation for obtain the involved words' id is as follow:

$$Z = (\boldsymbol{x}A)^T \odot W, \tag{10}$$

$$\text{IDs involved} = \arg\text{sort}(Z[:, g]), \tag{11}$$

Table 1. Weighted average F1-Score and (Macro F1-score) on the test sets. We run 5 times under the same preprocessing and random seed. Macro F1-score and Weighted F1-Score are the same on SST-2 and MR. Bold indicates the highest score and underline indicates the second highest score.

Model	SST-2	MR	CoLA	ArangoHate	FountaHate
MLP	80.78	75.55	61.39 (53.20)	84.71 (84.42)	79.22 (65.33)
Text-GCN	80.45	75.67	56.18 (52.30)	84.77 (84.43)	78.74 (64.54)
Bi-LSTM	81.32	76.39	62.88 (55.25)	84.92 (84.58)	79.04 (65.13)
VGCN	81.64	76.42	63.59 (54.82)	85.97 (85.69)	79.00 (64.04)
BERT	91.49	86.24	81.22 (77.02)	87.99 (87.75)	80.59 (66.61)
Vanilla-VGCN-BERT	91.38	**86.49**	80.70 (76.30)	88.01 (87.79)	81.11 (67.86)
VGCN-BERT	**91.93**	86.35	**83.68** (**80.46**)	**88.43** (**88.22**)	**81.26** (**68.45**)

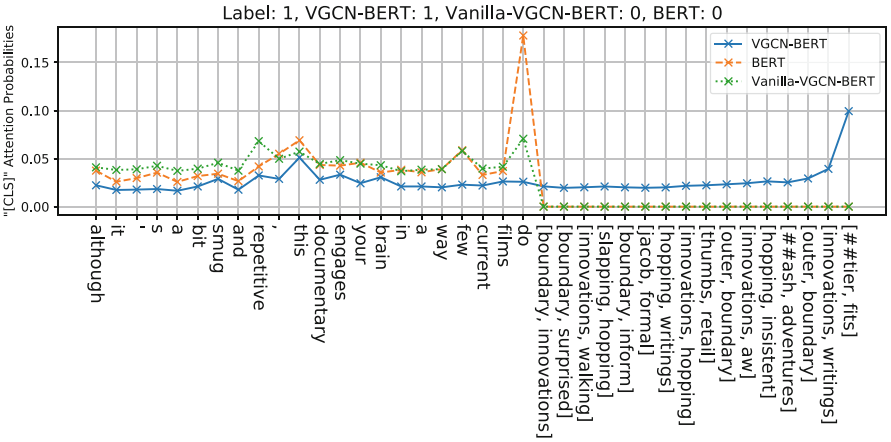

Fig. 2. Visualization of the attention that the token [CLS] (used as sentence embedding) pays to other tokens. The first part corresponds to word embeddings of the sentence. The second part is the graph embedding. [word1, word2] indicates the approximate meaning of a dimension in graph embedding.

where x is a document in row vector, $g \in [1, G]$, $G = 16$ is the size of graph embedding. For example, the first dimension of graph embedding shown in Fig. 2 corresponds roughly to the meaning of *"[boundary, innovations]"*.

In Fig. 2, we show the attention paid to each word (embedding) and each dimension of graph embedding (second part). As BERT does not use graph embedding, the attention paid to graph embedding is 0. In VGCN-BERT, we see that graph embedding draws an important part of attention.

For the movie review *"Although it's a bit smug and repetitive, this documentary engages your brain in a way few current films do."*, the first half of the sentence is explicitly negative, while the remaining part expresses a positive attitude in an implicit way, which makes the sentence difficult to judge. For this

example, BERT pays a very high attention to *"do"*, and a quite high attention to *"this"*. These words do not bear much meaning in sentiments. The final classification results by BERT is 0 (negative) while the true label is 1 (positive).

Vanilla-VGCN-BERT concatenates graph embedding with BERT without interaction between them. We can see that still no attention is paid to graph embedding, showing that such a simplistic combination cannot effectively leverage vocabulary information.

Finally, for VGCN-BERT, we see that a considerable part of attention is paid to graph embedding. The graph embedding is produced by integrating gradually the local information in the sentence with the global information in the graph. At the end, several dimensions of the graph embedding imply the meaning of *"innovation"*, to which quite high attentions are paid. This results in classifying the sentence to the correct class (positive).

The meaning of *"innovation"* is not produced immediately, but after a certain number of layers in BERT. In fact, through the layers of BERT, local information in the input sentence is combined to generate a higher level representation. In this example, at a certain layer, the expression *"a way few current films do"* is grouped and represented as an embedding similar to the meaning of *"innovation"*. From then, the meaning related to *"innovation"* in the graph embedding is capture through self-attention, and reinforced later on through interactions between the local and global information.

5 Conclusion and Future Work

In this study, we propose a new VGCN-BERT model to integrate a vocabulary graph embedding module with BERT. The goal is to complement the local information captured by BERT with the global information on the vocabulary, and allow both types of information to interact through the layers of attention mechanism. Our experiments on classification on 5 datasets show that the graph embedding does bring useful global information to BERT and this improves the performance. In comparison with BERT and VGCN alone, our model can clearly lead to better results, showing that VGCN-BERT can indeed take advantage of both mechanisms.

As future work, we will consider using other types of vocabulary graph such as Wordnet, in addition to a graph created by co-occurrences. We believe that Wordnet contains useful connections between words that NPMI cannot cover. It is thus possible to combine several lexical resources into the vocabulary graph.

References

1. Arango, A., Perez, J., Poblete, B.: Hate Speech Detection is Not as Easy as You May Think: A Closer Look at Model Validation. Paris (2019)
2. Battaglia, P.W., et al.: Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261 (2018)

3. Bouma, G.: Normalized (pointwise) mutual information in collocation extraction. In: Proceedings of the Biennial GSCL Conference 2009, University of Potsdam (2009). https://pdfs.semanticscholar.org/1521/8d9c029cbb903ae7c729b2c644c24994c201.pdf
4. Buitinck, L., et al.: API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pp. 108–122 (2013)
5. Cai, H., Zheng, V.W., Chang, K.: A comprehensive survey of graph embedding: problems, techniques and applications. IEEE Trans. Knowl. Data Eng. **30**(9), 1616–1637 (2018)
6. Davidson, T., Warmsley, D., Macy, M., Weber, I.: Automated hate speech detection and the problem of offensive language. In: Proceedings of the 11th International AAAI Conference on Web and Social Media, ICWSM 2017, pp. 512–515 (2017)
7. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: NIPS, pp. 3844–3852 (2016)
8. 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)
9. Founta, A.M., et al.: Large scale crowdsourcing and characterization of twitter abusive behavior. In: 11th International Conference on Web and Social Media, ICWSM 2018. AAAI Press (2018)
10. Grave, E., Mikolov, T., Joulin, A., Bojanowski, P.: Bag of tricks for efficient text classification. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, 3–7 April 2017, Volume 2: Short Papers, pp. 427–431 (2017). https://www.aclweb.org/anthology/E17-2068/
11. Graves, A., Mohamed, A., Hinton, G.: Speech recognition with deep recurrent neural networks. In: Acoustics, Speech and Signal Processing (ICASSP), pp. 6645–6649 (2013)
12. Henaff, M., Bruna, J., LeCun, Y.: Deep convolutional networks on graph-structured data. arXiv preprint arXiv:1506.05163 (2015)
13. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
14. Jeong, C., Jang, S., Shin, H., Park, E., Choi, S.: A context-aware citation recommendation model with BERT and graph convolutional networks. arXiv:1903.06464 (2019)
15. Kim, Y.: Convolutional neural networks for sentence classification. In: EMNLP, pp. 1746–1751 (2014)
16. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
17. Lever, J., Krzywinski, M., Altman, N.: Classification evaluation. Nat. Methods **13**(8), 603–604 (2016). https://doi.org/10.1038/nmeth.3945
18. Levy, O., Goldberg, Y.: Neural word embedding as implicit matrix factorization. In: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, Montreal, Quebec, Canada, 8–13 December 2014, pp. 2177–2185 (2014). http://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization
19. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, 2–4 May 2013, Workshop Track Proceedings (2013). http://arxiv.org/abs/1301.3781

20. Pang, B., Lee, L.: Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In: ACL, pp. 115–124 (2005)
21. Peng, H., et al.: Large-scale hierarchical text classification with recursively regularized deep graph-CNN. In: WWW, pp. 1063–1072 (2018)
22. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), vol. 14, pp. 1532–1543 (2014)
23. Shang, J., Ma, T., Xiao, C., Sun, J.: Pre-training of graph augmented transformers for medication recommendation. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, 10–16 August 2019, pp. 5953–5959 (2019). https://doi.org/10.24963/ijcai.2019/825
24. Socher, R., et al.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP), vol. 14, pp. 1631–1642 (2013)
25. Srinivasan, B., Ribeiro, B.: On the equivalence between node embeddings and structural graph representations. In: Proceedings of International Conference on Learning Representations 2020 (2020). https://openreview.net/forum?id=SJxzFySKwH
26. Tang, J., Qu, M., Mei, Q.: PTE: predictive text embedding through large-scale heterogeneous text networks. In: KDD, pp. 1165–1174. ACM (2015)
27. Vaswani, A., et al.: Attention is all you need. Long Beach (2017)
28. Wang, Y., Huang, M., Zhao, L., et al.: Attention-based LSTM for aspect-level sentiment classification. In: EMNLP, pp. 606–615 (2016)
29. Warstadt, A., Singh, A., Bowman, S.R.: Neural network acceptability judgments. arXiv preprint arXiv:1805.12471 (2018)
30. Waseem, Z.: Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter (2016)
31. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: NAACL, pp. 1480–1489 (2016)
32. Yao, L., Mao, C., Luo, Y.: Graph convolutional networks for text classification. In: AAAI (2019)
33. Zhang, Y., Liu, Q., Song, L.: Sentence-state LSTM for text representation. In: ACL, pp. 317–327 (2018). https://aclanthology.info/papers/P18-1030/p18-1030