entropy

MDPI

Article

Bert-Enhanced Text Graph Neural Network for Classification

Yiping Yang and Xiaohui Cui *

Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, Wuhan University, Wuhan 430000, China; yangyiping@whu.edu.cn

* Correspondence: xcui@whu.edu.cn

Abstract: Text classification is a fundamental research direction, aims to assign tags to text units. Recently, graph neural networks (GNN) have exhibited some excellent properties in textual information processing. Furthermore, the pre-trained language model also realized promising effects in many tasks. However, many text processing methods cannot model a single text unit's structure or ignore the semantic features. To solve these problems and comprehensively utilize the text's structure information and semantic information, we propose a Bert-Enhanced text Graph Neural Network model (BEGNN). For each text, we construct a text graph separately according to the co-occurrence relationship of words and use GNN to extract text features. Moreover, we employ Bert to extract semantic features. The former part can take into account the structural information, and the latter can focus on modeling the semantic information. Finally, we interact and aggregate these two features of different granularity to get a more effective representation. Experiments on standard datasets demonstrate the effectiveness of BEGNN.

Keywords: text classification; Bert; graph neural networks

Citation: Yang, Y.; Cui, X. Bert-Enhanced Text Graph Neural Network for Classification. *Entropy* **2021**, *23*, 1536. https://doi.org/10.3390/e23111536

Academic Editor: Fernando Morgado-Dias

Received: 20 October 2021
Accepted: 17 November 2021
Published: 18 November 2021

1. Introduction

Text classification is a fundamental task in natural language processing. It aims to assign labels to natural language text. Text classification has wide range of application scenarios, such as in sentiment classification and question answering [1,2]. Among these related tasks, text classification is the core of their application. It is used to deal with complex text information, which provides great help for fast and accurate text mining. For example, in a sentiment classification task, we focus on sentiment related words and classify texts by establishing a special emotion related dictionary. In the selective question answering, we extract the features of questions and alternative answers and classify them to select the most appropriate answer. Since the text is unstructured data written in natural language, which brings certain difficulties to its classification.

Early classification methods used bag-of-words features [3], that is, to calculate which word appeared in the text, and take it as the representation of the text, but this method did not consider the context information. The last decade has witnessed significant advances in text feature extraction using deep neural networks. Recently, with the progress in artificial intelligence research, a large number of neural network-based models are widely used in the task of text classification. Pre-trained word vectors such as Word2Vec [4] provide better initial embeddings for the tokens in the sentences. Other models such as RNN [5] and TextCNN [6] have also been proven effective in processing text data. In recent years, the pre-trained language model Bert [7] has gained increased attention and has refreshed the records in multiple natural language processing tasks. Attention mechanism [8] has also been integrated into various deep learning methods, which greatly improves the classification accuracy.

However, these methods cannot model the local and global structure features in text. While GNN has natural advantages in modeling structural information. There have been studies using graph neural networks to model text data. Some works build

homogeneous graphs or heterogeneous graphs from text data and perform graph neural network propagation such as convolution operations on the graphs [9,10]. In this way, the model can take into account the structural information, which is of great significance for understanding the meaning of the text. However, some methods build text graphs on the entire dataset, weakening the individual features of each document.

Based on the above analysis, the existing text classification methods have some limitations in text feature extraction. First, most models use RNN, LSTM [5] and other methods to process serialized data, which cannot take into account the text structure information. Secondly, some methods based on graph neural networks extract the representation of text by building a heterogeneous graph structure for the entire dataset, but it's hard to consider a single text's semantic features. In addition, some methods have combined structural features and semantic features of sequences for extraction, but they can not consider single text features alone or do not consider the interaction between features, which limits their representation ability.

To solve the problems of these algorithms, we construct the BEGNN model. Specifically, we first construct a graph structure for each document separately. Moreover, we propose to aggregate the features extracted from Bert and the features extracted by graph structures. The former represents the semantic information of the documents, and the latter is a representation that considers the structural feature of the text. Compared with other work, we also add a co-attention module to solve the problem of interaction between features, and performed a variety of experiments to integrate the features, which can maximize the representation ability of the extracted features.

Our contribution is as follows:

(1) Our model can extract features of different granularities, from a pre-trained language model and graph neural networks for text representation. It not only takes into account the semantic information, and also the structural information, which improves the effect of the learned text representation.

(2) In order to prevent the two features from being separated during the prediction process, we have designed and performed experiments on co-attention modules as well as different aggregation methods, which can consider the interaction of the two representations and make full use of them to achieve better classification capabilities.

(3) The experiment results and analysis on four datasets demonstrate the effectiveness of BEGNN.

In the following paragraphs: Section 2 introduces researches about text classification methods related to our work, Section 3 illustrates the overall model we proposed, Section 4 shows the experimental results, and finally, the conclusion.

2. Related Work

2.1. Traditional Feature Engineering Method

Traditional text classification methods need to extract manually defined features, and they are often combined with machine learning algorithms for training and prediction. For a specific task, some early studies classify sentences or documents by analyzing text data and extracting statistical features of the text, then use pre-specified training set as training data. Bag-Of-Words (BOW) [11] and n-grams [12] are commonly used word-based representation method, which represents a sentence based on a collection of words or n-gram sequences which occur in it. These features are usually combined with models such as SVM [11] and have achieved good results. However, machine learning requires extensive feature engineering and relies on domain knowledge, which makes it difficult for the features on a single task to be generalized to other aspects.

2.2. Deep Learning-Based Method

Methods based on deep learning have been investigated to resolve the limitations of manual feature engineering [13,14]. The text is automatically mapped to a low-dimensional vector through the model to extract text features. Word embedding has brought new

solutions to natural language tasks. Word2Vec [4] and GloVe [15] have been drawing great attention in NLP tasks. Mikolov et al. have shown these pre-trained embeddings can capture meaningful semantic features [16]. In addition, RNN models [17] have shown advantages in processing sequence data. TextCNN [6] performs convolution operations on text features and has achieved good results. Tan et al. [18] use a structure based on a dynamic convolutional gated neural network, making it possible to selectively control how much context information is contained in each specific location.

Recently, pre-trained language models have caused a great upsurge in research. Models such as Bert [7] are pre-trained on a large corpus, and can be simply transferred to downstream NLP tasks with fine-tuning, which have refreshed records on multiple NLP tasks. Bert [7] takes advantage of the self-attention mechanism, and builds a multi-layer self-attention network, which can also realize parallel computing. The attention mechanism is applied to various models, greatly improving the performance of various NLP tasks [19]. There have also been some studies exploring how to efficiently use Bert for natural language processing tasks [20–22]. These models have been proved effective in extracting features, but they cannot fully utilize the text's structural features. While graph structure has natural advantages in modeling structural information.

There have been some researches that model text as graph structure for feature extraction. GNN [23] can capture the features of the nodes and the structural features in the graph, which can learn more effective representations for the nodes or the whole graph. GatedGNN [10] and GCN [9] have been applied to the task of text classification. Textgcn [9] constructed a heterogeneous graph network of words and documents, and uses co-occurrence features and TFIDF to measure the relationship between words and documents. For a new document, it needs to update the whole graph structure to perform prediction. Additionally, it cannot take into account the structural characteristics of a single document well. TextING [10] builds graph structure on each single text, which can learn the fine-grained word representation of the local structure. Lei et al. [24] designed a structure that can integrate the graph convolutional features of multi-layer neighbors, alleviating the problem of over-fitting to a certain extent. However, semantic features used in the models rely on pre-trained word embeddings, which limits the effect of the model. Parcheta et al. [25] studied the influence of embeddings extracted by combining different methods on text classification models.

There are also some methods that combine the pre-trained language model with graph neural networks to extract features. VGCN-Bert [26] builds a graph of the whole dataset, and uses the features extracted by GCN [27] to enhance the effect of Bert [7]. However, as in GCN, the unique structural characteristics of each text cannot be fully taken into account. Jeong et al. [28] simply concatenate the features of Bert and GCN for the recommendation task, but this method cannot consider the features' interactive relationship, which reduces the representation ability. We show the methods of some related works in Table 1.

Considering the above problems, we propose to combine the features extracted by Bert [7] and graph neural networks, which can take into account the semantic and structural information of a single text. Different from the previous work, first of all, we build a graph structure for each text separately, and combine the graph neural network and Bert to extract different granular features. While most of the studies built a graph on the entire dataset or did not combine the different characteristics of different granularity. In addition, we employ the co-attention module to integrate features. As far as we know, we are the first to employ a co-attention module to combine the features of graph networks and Bert for text classification. So that we can take the advantages of the feature representation with different granularity.

Table 1. Comparison of some related work.

Related Researches	Method
[7]	Self-encoding language model, constructed using the encoder module of transformer.
[20]	Explore how to fine-tune Bert to improve the effect of text classification, including hierarchical learning rate adjustment, multi-task pre-training and other methods.
[21]	Prove the superiority of bert over traditional machine learning algorithms.
[9]	Build a heterogeneous graph for the entire dataset, and use the representation of the document node as the classification basis.
[10]	Build a separate graph for each text, use graph neural network and design the output layer to get the representation and classification of the text.
[26]	Build a graph for the entire dataset, word embedding and graph feature are feeded into attention layer.
[24]	Build a graph for the entire dataset, integrate the graph convolution features of multi-hop neighbors.
[28]	The output of GNN and Bert are concatenated for the recommendation task.
[18]	Dynamically Gated Convolutional Neural Network.
[25]	Research on effect of different embedding technologies when they are used together.

3. Method

In this part, we describe the structure of BEGNN in detail.

3.1. Architecture Overview

The model structure is illustrated in Figure 1. BEGNN is composed of five modules: graph construction, Bert-based feature extraction, GNN based feature extraction, feature interaction and aggregation. Given a document represented as $W_i = \{w_1, w_2, \ldots, w_n\}$, according to co-occurrence relationship of the words, we construct each text as a graph. By initializing the representation of graph nodes using word vectors and employing a graph neural network, we get the structure feature of each word. Moreover, We input the text into Bert [7] for semantic feature extraction. Finally, the two feature representations interact and aggregate through the co-attention layer and the aggregation layer to obtain the aggregated representation $\mathbf{H}_i = \{h_1, h_2, \ldots, h_n\}$. Taking the final representation $\mathbf{H}_i$, we finally use the fully connected layer to predict the category. The details are presented in Sections 3.2–3.6, respectively.

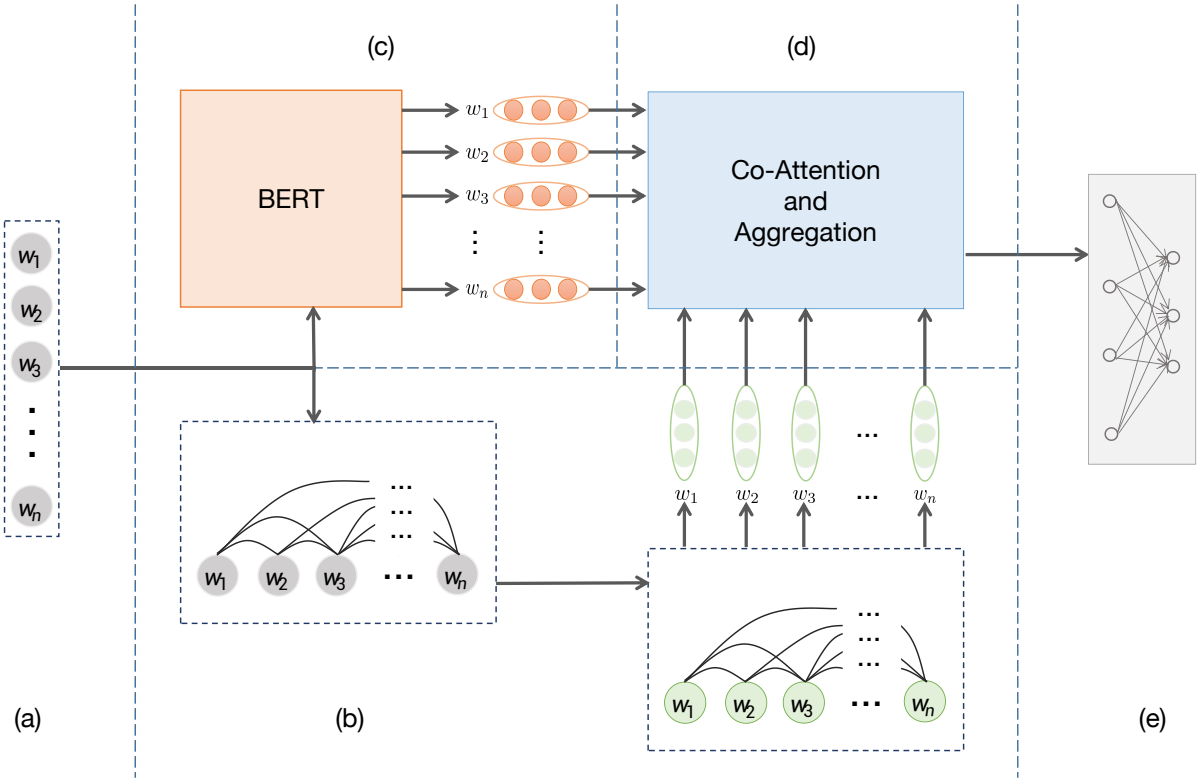

Figure 1. The architecture of BEGNN. (**a**) The input document. (**b**) Graph construction and graph neural network based feature extraction. (**c**) Bert based feature extraction. (**d**) Interactive feature aggregation. (**e**) Fully-connected layer.

3.2. Graph Construction

We create separate graphs for the documents, expressed as $G = (V, E)$. V is the set of nodes in the graph, including all the words in the text. While E includes edges between nodes. We use standard methods to pre-process the text, including word segmentation and cleaning. Afterwards, co-occurrence information is extracted to model the relationship between words in a document. We build an undirected text graph by setting a fixed-size sliding window, connecting the words appearing in the same window with undirected edges. Figure 2 is an instance.

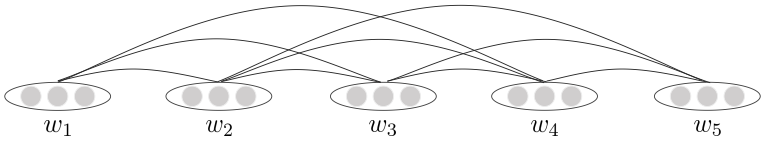

Figure 2. The graph constructed for a document with five words.

The feature vector of the nodes are initialized with the GloVe word vector [15], document i is represented by $\mathbf{H}_i^0 = \left\{h_1^0, h_2^0, \ldots, h_{|V|}^0\right\}$. $\mathbf{H}_i^0 \in \mathbb{R}^{|V|*d}$, d is the word embedding size. For the graph structure established for each document, we use graph neural network for message passing.

3.3. Graph Neural Network Based Feature Extraction

For each text graph, we use Gated Graph Neural Networks [29] for feature propagation and extraction. It is a classical spatial domain message passing model based on GRU. The proposal of Gated Graph Neural Networks enables GNN to be better used to deal with sequence problems. In the process of message passing in the whole graph structure, the principle of GRU is adopted. The embedding of a node at time $t + 1$ is determined by the embedding of itself and its neighbor nodes, and the edge information of the interaction

between the nodes. By stacking such layers for T times, the nodes are able to receive the information of their T-hop neighbors. The formulas of the propagation recurrence in the k-th layer are:

$$\mathbf{a}^{t+1} = \mathbf{A}\mathbf{H}^{t}\mathbf{W}_{a} + \mathbf{b} \tag{1}$$

$$\mathbf{z}^{t+1} = \sigma\Big(\mathbf{W}_{z}\mathbf{a}^{t+1} + \mathbf{U}_{z}\mathbf{H}^{t} + \mathbf{b}_{z}\Big) \tag{2}$$

$$\mathbf{r}^{t+1} = \sigma\Big(\mathbf{W}_{r}\mathbf{a}^{t+1} + \mathbf{U}_{r}\mathbf{H}^{t} + \mathbf{b}_{r}\Big) \tag{3}$$

$$\tilde{\mathbf{H}}^{t+1} = \tanh\Big(\mathbf{W}_{H}\mathbf{a}^{t+1} + \mathbf{U}_{H}\Big(\mathbf{r}^{t+1} \odot \mathbf{H}^{t}\Big) + \mathbf{b}_{H}\Big) \tag{4}$$

$$\mathbf{H}^{t+1} = \tilde{\mathbf{H}}^{t+1} \odot \mathbf{z}^{t+1} + \mathbf{H}^{t} \odot \Big(1 - \mathbf{z}^{t+1}\Big) \tag{5}$$

where $A \in \mathbb{R}^{|V|*|V|}$ is the adjacency matrix, $\mathbf{a}^{t+1}$ represents the result of the interaction between the nodes and their adjacent nodes through the edges. Formulas (2)–(4) is similar to the calculation process of GRU. Among them, $\mathbf{z}^{t+1}$ controls the forgotten information, and $\mathbf{r}^{t+1}$ controls the newly generated information. $\mathbf{H}^{t+1}$ is the final updated node status of $t+1$-th layer. σ is sigmoid function. $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{b}$ are trainable weight matrices.

To simplify, we can write such a message passing process as:

$$\mathbf{H}^{t+1} = GGNN(\mathbf{H}^{t}, \mathbf{A}; \Theta_{t}) \tag{6}$$

where Θ_t is the parameter set of the gated graph neural network of the t-th layer. After message passing of T layers, we get the final representation $\mathbf{H}_0^T$.

3.4. Bert Based Feature Extraction

In addition to using GNN to obtain the features of the word nodes, we also fine-tune Bert to obtain the words' semantic features. Pre-trained on large-scale corpus in an unsupervised way, the parameters of Bert are then fine-tuned according to downstream tasks. Bert is composed of the encoder of transformer module, which includes the self-attention layer and feed-forward layer. Self-attention is calculated by:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{T}}{\sqrt{d_{k}}}\right)\mathbf{V} \tag{7}$$

$\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ are the matrix of queries, keys and values, respectively. d_k is the dimension of the matrices. Furthermore, multi-head attention can be defined as:

$$MultiHead(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = CONCAT(\mathbf{head_1}, \ldots, \mathbf{head_n})\mathbf{W} \tag{8}$$

$$\mathbf{head_i} = \text{ Attention } \left(\mathbf{Q}\mathbf{W}_{i}^{Q}, \mathbf{K}\mathbf{W}_{i}^{K}, \mathbf{V}\mathbf{W}_{i}^{V}\right) \tag{9}$$

After the multi-layer transformer module, we eventually get the final word feature representation $\mathbf{H}_0^{bert}$.

3.5. Co-Attention Layer

We introduce the co-attention layer as shown in Figure 3. Given the text representation extracted by GNN and Bert, the query, key and value matrices are calculated, just as they are calculated in the standard self-attention mechanism. However, the keys and values of both text features are passed as input to each other's multi-headed attention block.

According to $\mathbf{H}_0^T$ and $\mathbf{H}_0^{bert}$, we calculate the query, key and value matrix, respectively. Different from the self-attention mechanism, we take $\mathbf{H}_0^T\mathbf{W}_T^Q$, $\mathbf{H}_0^{bert}\mathbf{W}_{bert}^K$ and $\mathbf{H}_0^{bert}\mathbf{W}_{bert}^V$ as the input of the formula (7) to obtain $\mathbf{H}^T$, and take $\mathbf{H}_0^{bert}\mathbf{W}_{bert}^Q$, $\mathbf{H}_0^T\mathbf{W}_T^K$ and $\mathbf{H}_0^T\mathbf{W}_T^V$ as the input to obtain $\mathbf{H}^{bert}$. Where $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V$ are parameter matrices.

Then we get the attention representation of GNN conditioned on Bert output and the attention representation of Bert conditioned on GNN output. Therefore, we obtain the mutually conditional attention convergence feature between the two representations.

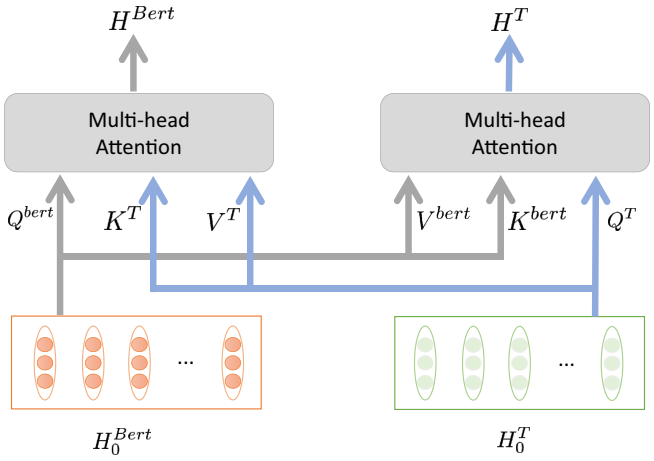

Figure 3. Co-attention layer.

3.6. Feature Aggregation

We designed three ways to aggregate the extracted features interactively, namely max-pooling, concatenation and addition. For a word w_i in the sequence, the features extracted by GNN and Bert are denoted as $\mathbf{h}_i^{gnn}$ and $\mathbf{h}_i^{bert}$, respectively. The three aggregation methods are as follows.

max-pooling. This function takes the larger value of the two features in each dimension to form the final representation:

$$\mathbf{h}_i = MAX\left(\mathbf{h}_i^{gnn}, \mathbf{h}_i^{bert}\right) \tag{10}$$

which chooses the most informative feature in each dimension.

Concatenation. It takes the concatenation of the representation directly in the node feature dimension:

$$\mathbf{h}_i = CONCAT\left(\mathbf{h}_i^{gnn}, \mathbf{h}_i^{bert}\right) \tag{11}$$

which can keep the output of each module intact.

Addition.

$$\mathbf{h}_i = \mathbf{h}_i^{gnn} \oplus \mathbf{h}_i^{bert} \tag{12}$$

where $\oplus$ operation means element-wise addition.

We denote the final representation of the whole document i as $\mathbf{H}_i$.

3.7. Final Prediction

After feature aggregation, we employed a fully connected layer for classification. We minimize the cross-entropy loss to train our model.

$$\hat{y} = \text{softmax}(\mathbf{W}\mathbf{H}_i + \mathbf{b}) \tag{13}$$

$$L = -\sum_i y_i \cdot \log \hat{y}_i \tag{14}$$

$\mathbf{W}$, $\mathbf{b}$ are trainable parameters. $\hat{y}_i$ and y_i are the predicted and true label for the document, respectively.

4. Experiments

Here, we evaluated the effect of BEGNN and compared it with baseline models on four publicly available datasets.

4.1. Datasets

We adopted four widely used datasets for text classification:

MR [30]. It is a sentiment classification dataset, each review is classified as positive or negative.

SST-2 [31]. It is the Stanford Sentiment Treebank dataset, which includes sentences from movie reviews. Each sample is labeled as negative or positive.

R8 [32]. It is a subset of the Reuters-21578 dataset and had been manually classified into eight categories.

Ohsumed [33]. It is from the MEDLINE database, which is a bibliographic database. Each document had been classified into 23 cardiovascular diseases categories.

The statistics are in Table 2.

For each dataset, we use 10% of the training data for validation to assist in model training. For each piece of data in the dataset, we proceed with it as follows. First, a BertTokenizer is used to segment the document. Second, in the Bert-based feature extraction module, we directly use the segmentation as the input. Third, in the graph neural network-based module, to ensure that the two modules can be aligned, we use the result of Bert word segmentation, and then use the Glove word vector as the words' initial representation.

Table 2. Statistics of datasets.

Dataset	Documents	Training	Test	Class	Average Length
MR	10,662	7108	3554	2	18
SST-2	9613	7792	1821	2	19
R8	7674	5485	2189	8	41
Ohsumed	7400	3357	4043	23	79

4.2. Compared Methods

We make a comparison with some state-of-the-art models, including deep models for processing serialized data and models based on GNNs.

Fasttext [34]. A lightweight neural network model. The input is multiple words represented by vectors. In the hidden layer, the average of word vectors is calculated. The last hidden layer's output is the basis for classification.

Bi-LSTM [35]. It is a kind of RNN. It is specially designed to solve the long-term dependency problem of general RNN. The final hidden state is used for classification.

TextGCN [9]. A GNN based text classification model. The whole corpus is used to construct a large heterogeneous graph. Furthermore, GCN is designed to jointly learn the embedding of words and documents. We build the text graph in the same way as the original paper and use the final representation of the document node as the basis for classification.

TextING [10]. It is another graph based model. Different from TextGCN, it constructs a graph for each text. The final representation of the text is obtained through the output layer and classified.

VGCN-Bert [26]. The word embedding and graph features are fed to the attention layer. Then the attention module's output is used as the basis for classification.

BEGNN (our proposed method). It is a text classification model combining graph neural networks and Bert, which can extract the semantic and structural information of the text.

Fasttext [34] is a non sequential model while LSTM [35] is a model for sequential data. TextGCN [9] and TextING [10] are graph based models. TextGCN builds a large graph of

thesaurus and documents together. The difference is that TextING builds a text graph of words in each document. By comparing these methods, we can analyze which feature is more important to the model.

4.3. Hyper-Parameter Settings

Regarding the setting of hyperparameters, based on previous research and experimental experience, we refer to some optimization algorithms based on the Bayesian method [36,37], and use the python open-source toolkit 'advisor' for parameter optimization.

Regarding the relevant models we compared, we continued the parameter settings of original papers for experiments. For fair comparison, we uniformly use GloVe embedding [15] as the initial word feature vector. For our proposed method BEGNN, we set the learning rate of 0.00005, the l2 regularization weight of 0.01, and the optimized function of Adam. For the text graph, the sliding window size is 3. The number of attention heads is set to 8. Early stopping is applied, the number of epochs is 100. The nonlinearity function is set to ReLU. We use the BertTokenizer to split text. For each dataset, we use three interactive aggregation methods we designed to aggregate the features and report the best results. While training, to ensure the convergence, we firstly pre-trained the GNN network, and then trained the entire model.

4.4. Experimental Results

We adopt the classification accuracy and the macro-F1 value as the evaluation metrics. From the experimental results, we can make the following observations. The main results are presented in Table 3.

Table 3. This is a table caption. Tables should be placed in the main text near to the first time they are cited.

Methods	MR	SST-2	R8	Ohsumed
Fasttext	75.04	80.25	96.11	57.70
Bi-LSTM	76.27	81.23	96.30	50.27
TextGCN	75.56	80.25	96.89	67.44
TextING	78.93	83.69	97.92	70.41
VGCN-BERT	86.21	91.02	97.98	70.53
BEGNN	86.42	91.43	98.41	71.19

(1) BEGNN outperforms all the baselines. We use Bert based feature extraction module and GNN based feature extraction module. At the same time, the co-attention module is employed to interactively combine the two features. Suggesting that the combination of GNN based method and pre-trained language method benefits text processing.

(2) The longer the text, the more obvious the improvement of our model to the experimental effect. According to the statistics of the datasets, the text length of R8 and Ohsumed is longer. Especially on the Ohsumed dataset, the average text length is 79. On the datasets where the average text length is less than 20, the performance improvement of our model is relatively lower than the other two datasets with longer text. This shows our model can better process longer texts. Our feature extraction module based on graph neural network passes through message in multiple layers, and can mine the information of multi-hop neighbors. Superior to RNN based model, the self-attention module in Bert can also pay attention to words that are farther away.

(3) RNN based model outperforms Fasttext and TextGCN in two datasets, and shows comparable capability in R8, which shows its advantages in processing sequential data. While in Ohsumed, it does not perform well. The text length of this dataset is long, causing difficulties in processing long-distance context. RNN-based models have no advantage when dealing with longer text data. After long-distance propagation, information will be

lost. LSTM adds the memory module to solve the problem of long-distance dependence of traditional RNN architecture, but when the average text length exceeds 70 in the Ohsumed dataset, there are still some problems.

(4) TextGCN and TextING are graph based models. When they are used in text classification tasks, TextING has achieved better results on each dataset. This is because, for the texts, TextGCN constructs a graph of the entire corpus, which is low-density. However, TextING constructs a graph structure for each document separately, which can take into account the different structural information of each text, which will not be so sparse as it in TextGCN.

(5) The performance of VGCN-Bert surpasses other models besides our proposed model. It takes the features extract from graph neural networks and word embedding features as the input of the attention module. However, it builds a graph structure on the entire dataset. Compared with our operation of building a graph structure from a single text, it cannot fully consider the unique structural characteristics of each text. Furthermore, it chooses to concatenate the two representations and send them to the attention module. Different from it, we interact and aggregate the features from GNN module and Bert based module, which can avoid the separation of the two representations and utilize their correlation.

Compared to other related models, first of all, the experimental results demonstrate the superiority of BEGNN. Secondly, our model shows a more obvious advantage in the processing of long texts and can extract features that span longer distances. In addition, our model can take the semantic and structure information of the given documents. The transformer module in Bert uses the attention mechanism to perform parallel calculations, also extracts semantic features. The module based on GNN can extract the structure information of the text well. While the interactive aggregation of these two features can combine the advantages of these two features to the greatest extent. This ensures that BEGNN attains a better effect over the baseline models.

4.5. Ablation Study

To analyze the usefulness of each component of BEGNN, we performed the following experiments.

4.5.1. Effectiveness of the Text Graph

In our base model, we build a text graph based on the word co-occurrence relation in the document and aggregate the features obtained from the text graph and the features obtained from Bert. Compared with the original Bert, our model can not only consider the semantic features, but also integrate the structural information. To validate the effectiveness of this module, we designed experiments to compare the effects of our basic model and the model without a graph neural network. We name the model with GNN module removed as BEGNN-GNN. That is, a separate Bert model. Figure 4 illustrates the experimental results. At the same time, we also experimented on the model BEGNN-CoAttention without a feature interaction module.

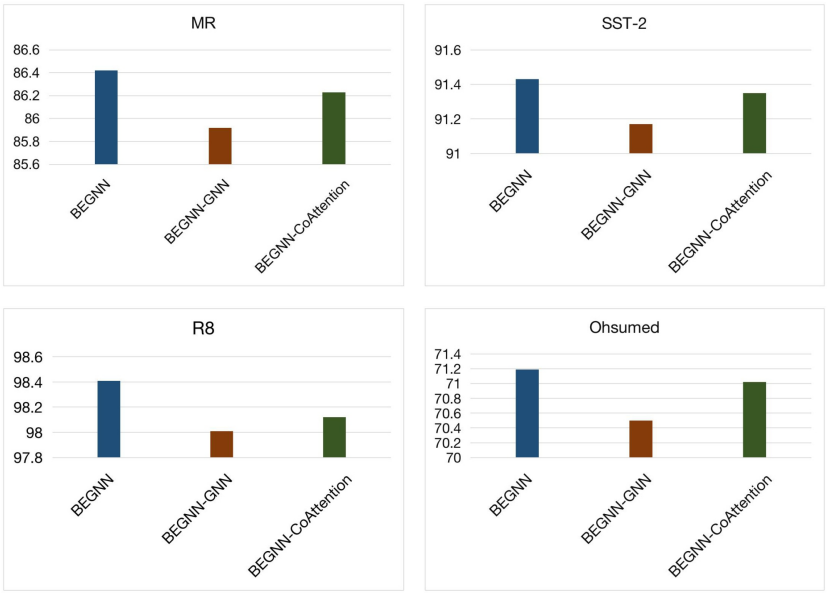

Figure 4. Ablation study of the text graph and the co-attention modules of the model.

Compared with using Bert only for training and testing, our original model with graph neural network achieves significant results on four datasets. This confirms the necessity of adding a text graph neural network in our proposed model. Among them, the model with graph structure features has achieved the most significant effect on the Ohsumed dataset. Showing advantages of BEGNN in processing longer text features. Compared with the model without the graph neural network feature extraction module, even without feature interaction, the model containing two granular features still achieves better results than the original Bert model. This also illustrates the importance of adding structural features. Other than semantic features, adding structural features can improve the representation ability of the extracted joint features.

4.5.2. Effectiveness of the Co-Attention Module

On the basis of using Bert to extract semantic features, and adding the structural features extracted by the graph neural network module, we also hope that the two features can interact, rather than being separated from each other. For the features extracted from the Bert model and the graph neural network, we add the co-attention mechanism in order to provide interaction between these two features. We name the model with co-attention module removed as BEGNN-CoAttention.

As shown in Figure 4, removing the co-attention module in the training procedure causes performance degradation on four datasets. In the four datasets, although there is a certain gap in text length, the degree of effect decline is basically the same. When dealing with the interaction of text data, the co-attention mechanism is important in both long and short texts. In the co-attention module, we get the attention representation of GNN conditioned on the features extracted from Bert, and the attention representation of Bert conditioned on the features extracted from GNN. In this way, the two representations interact with each other and improve the performance.

5. Conclusions

In this article, we conduct research on text classification algorithms. The application scenarios of text classification are very extensive, and it is important in public opinion analysis and news classification. We propose a Bert-enhanced graph neural network (BEGNN) to improve the representation ability of text. Although it is designed for text classification, its ideas can be applied to other research fields, such as information retrieval. We build a text graph structure for each document and extract the structural features of

the text. Furthermore, Bert is used to extract semantic features. In addition, we added an interaction module and aggregated the semantic and structural features of text. Different from other studies, we can take into account the two granular text features in an innovative way, and employ the co-attention module to interact and aggregate them. Experimental results prove the effectiveness of BEGNN.

In future research, we will further study what algorithms and features will have a positive impact on the deep learning model when using Bert and graph neural network for feature extraction. At the same time, we will study how to use this analysis result to further optimize the model, increase the interpretability of the model and produce more fine-grained and reasonable interpretation. We will also consider further research on the lightweight optimization to reduce the cost of calculation and reasoning while ensuring the effect of the model.

Author Contributions: Conceptualization, Y.Y. and X.C.; methodology, Y.Y.; software, Y.Y.; validation, Y.Y.; formal analysis, Y.Y. and X.C.; investigation, Y.Y.; resources, Y.Y.; data curation, Y.Y.; writing—original draft preparation, Y.Y.; writing—review and editing, Y.Y. and X.C.; visualization, Y.Y.; supervision, Y.Y.; project administration, Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The source code and the datasets used in the experiments is available at https://github.com/pingpingand/BEGNN, accessed on 24 July 2021.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Minaee, S.; Kalchbrenner, N.; Cambria, E.; Nikzad, N.; Chenaghlu, M.; Gao, J. Deep Learning–based Text Classification: A Comprehensive Review. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–40. [CrossRef]
2. Chen, W.; Yu, W.; He, G.; Jiang, N. Coarse-to-Fine Attention Network via Opinion Approximate Representation for Aspect-Level Sentiment Classification. In *International Conference on Neural Information Processing*; Springer: Cham, Switzerland, 2020; pp. 704–715.
3. Wang, P.; Hu, J.; Zeng, H.J.; Chen, Z. Using Wikipedia knowledge to improve text classification. *Knowl. Inf. Syst.* **2009**, *19*, 265–281. [CrossRef]
4. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv Prepr.* **2013**, arXiv:1301.3781.
5. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
6. Guo, B.; Zhang, C.; Liu, J.; Ma, X. Improving text classification with weighted word embeddings via a multi-channel TextCNN model. *Neurocomputing* **2019**, *363*, 366–374. [CrossRef]
7. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv Prepr.* **2018**, arXiv:1810.04805.
8. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv Prepr.* **2014**, arXiv:1409.0473.
9. Yao, L.; Mao, C.; Luo, Y. Graph convolutional networks for text classification. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; pp. 7370–7377.
10. Zhang, Y.; Yu, X.; Cui, Z.; Wu, S.; Wen, Z.; Wang, L. Every document owns its structure: Inductive text classification via graph neural networks. *arXiv Prepr.* **2020**, arXiv:2004.13826.
11. Joachims, T. Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 137–142.
12. Ayed, R.; Labidi, M.; Maraoui, M. Arabic text classification: New study. In Proceedings of the 2017 International Conference on Engineering & MIS (ICEMIS), Monastir, Tunisia, 8–10 May 2017; pp. 1–7.
13. Ma, C.; Shi, X.; Zhu, W.; Li, W.; Cui, X.; Gui, H. An Approach to Time Series Classification Using Binary Distribution Tree. In Proceedings of the 2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks, Shenzhen, China, 11–13 December 2019; pp. 399–404.
14. Li, W.; Liu, X.; Liu, J.; Chen, P.; Wan, S.; Cui, X. On improving the accuracy with auto-encoder on conjunctivitis. *Appl. Soft Comput.* **2019**, *81*, 105489. [CrossRef]

15. 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, Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
16. Mikolov, T.; Yih, W.T.; Zweig, G. Linguistic regularities in continuous space word representations. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Atlanta, GA, USA, 9–15 June 2013; pp. 746–751.
17. Zhou, P.; Shi, W.; Tian, J.; Qi, Z.; Li, B.; Hao, H.; Xu, B. Attention-based bidirectional long short-term memory networks for relation classification. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 7–12 August 2016; pp. 207–212.
18. Tan, Z.; Chen, J.; Kang, Q.; Zhou, M.; Abusorrah, A.; Sedraoui, K. Dynamic embedding projection-gated convolutional neural networks for text classification. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, 1–10. [CrossRef] [PubMed]
19. Wang, Y.; Huang, M.; Zhu, X.; Zhao, L. Attention-based LSTM for aspect-level sentiment classification. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, TX, USA, 1–5 November 2016; pp. 606–615.
20. Sun, C.; Qiu, X.; Xu, Y.; Huang, X. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*; Springer: Cham, Switzerland, 2019; pp. 194–206.
21. González-Carvajal, S.; Garrido-Merchán, E.C. Comparing BERT against traditional machine learning text classification. *arXiv Prepr.* **2020**, arXiv:2005.13012.
22. Jin, D.; Jin, Z.; Zhou, J.T.; Szolovits, P. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; pp. 8018–8025.
23. Cai, H.; Zheng, V.W.; Chang, K. A comprehensive survey of graph embedding: problems, techniques and applications. *IEEE Trans. Knowl. Data Eng.* **2018**, *30*, 1616–1637. [CrossRef]
24. Lei, F.; Liu, X.; Li, Z.; Dai, Q.; Wang, S. Multihop Neighbor Information Fusion Graph Convolutional Network for Text Classification. *Math. Probl. Eng.* **2021**. [CrossRef]
25. Parcheta, Z.; Sanchis-Trilles, G.; Casacuberta, F.; Rendahl, R. Combining Embeddings of Input Data for Text Classification. *Neural Process. Lett.* **2020**, *53*, 1–29. [CrossRef]
26. Lu, Z.; Du, P.; Nie, J.Y. VGCN-BERT: Augmenting BERT with graph embedding for text classification. In *European Conference on Information Retrieval*; Springer: Cham, Switzerland, 2020; pp. 369–382.
27. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv Prepr.* 2016, arXiv:1609.02907.
28. Jeong, C.; Jang, S.; Shin, H.; Park, E.; Choi, S. A context-aware citation recommendation model with BERT and graph convolutional networks. *arXiv* **2019**, arXiv:1903.06464.
29. Li, Y.; Tarlow, D.; Brockschmidt, M.; Zemel, R. Gated graph sequence neural networks. *arXiv Prepr.* **2015**, arXiv:511.05493.
30. Pang, B.; Lee, L. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *arXiv Prepr.* **2005**, arXiv:cs/0506075.
31. Socher, R.; Perelygin, A.; Wu, J.Y.; Chuang, J.; Manning, C.D.; Ng, A.Y.; Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, WA, USA, 18–21 October 2013.
32. Cardoso-Cachopo, A.; Oliveira, A.L. Semi-supervised single-label text categorization using centroid-based classifiers. In Proceedings of the 2007 ACM Symposium on Applied Computing, Seoul, Korea, 11–15 March 2007; pp. 844–851.
33. Hersh, W.; Buckley, C.; Leone, T.J.; Hickam, D. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *SIGIR'94*; Springer: London, UK, 1994; pp. 192–201.
34. Joulin, A.; Grave, E.; Bojanowski, P.; Mikolov, T. Bag of tricks for efficient text classification. *arXiv Prepr.* **2016**, arXiv:1607.01759.
35. Graves, A.; Mohamed, A.; Hinton, G. Speech recognition with deep recurrent neural networks. In Proceedings of the Acoustics, Speech and Signal Processing (ICASSP), Vancouver, BC, Canada, 26–31 May 2013; pp. 6645–6649.
36. Kaselimi, M.; Doulamis, N.; Voulodimos, A.; Protopapadakis, E.; Doulamis, A. Context aware energy disaggregation using adaptive bidirectional LSTM models. *IEEE Trans. Smart Grid* **2020**, *11*, 3054–3067. [CrossRef]
37. Golovin, D.; Solnik, B.; Moitra, S.; Kochanski, G.; Karro, J.; Sculley, D. Google vizier: A service for black-box optimization. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 1487–1495.