A SEMANTIC HIERARCHICAL GRAPH NEURAL NETWORK FOR TEXT CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The key to the text classification task is language representation and important information extraction, and there are many related studies. In recent years, the research on graph neural network (GNN) in text classification has gradually emerged and shown its advantages, but the existing models mainly focus on directly inputting words as graph nodes into the GNN models ignoring the different levels of semantic structure information in the samples. To address the issue, we propose a new hierarchical graph neural network (HieGNN) which extracts corresponding information from word-level, sentence-level (sen-level) and document-level (doc-level) respectively. The doc-level focuses on processing samples from a global perspective, while sen-level and word-level focus on processing samples from the sentences and words themselves. The model is tested on five datasets, and compared with the pure GNN-based model and the hybrid GNN and BERT model, it achieves better classification results on two datasets and similar results on three datasets, which demonstrate that our model is able to obtain more useful information for classification from samples.

1 INTRODUCTION

Text classification, which aims to classify unknown texts into predefined categories, has always been a relatively popular research direction in natural language processing (NLP). Two very critical points in text classification are language representation and extraction of important information related to the task. Many classification methods such as CNN-based models (Kim, 2014; Zhang et al., 2015), attention-based models (Yang et al., 2016; Peters et al., 2018), transformer-based models (Devlin et al., 2019; Yang et al., 2019) have been proposed and achieved the state of the art (SOTA) results because their models represent the text input to the model well and extract enough semantic information to some extent from different semantic perspectives.

We know that from the macroscopic things as large as the cosmic celestial bodies to the microscopic things as small as molecules and atoms, we can use graphs to describe their internal relationships, and, text data is no exception. Recently, graph neural networks have shown advantages in text classification, especially news classification tasks (Peng et al., 2018; Wu et al., 2019; Pal et al., 2020) due to their structural flexibility and convenience for abstract relationships between entities (Kipf & Welling, 2017). Yao et al. (2019) modeled a dataset directly into a graph structure to obtain word-word and word-document relationships and got the best classification results at the time. However, when new samples need to be predicted, the model needs to be retrained. To solve this problem, Huang et al. (2019) built each sample in the dataset into a graph and used two global matrices to store node features and edge weight, respectively. However, these methods have two main problems: firstly, these methods ignore the hierarchical structure information of the text, that is, within the sample, the relationship between sentences or phrases is ignored; secondly, the vector representation of a word in the global matrix will not change with different context, which means that a word has only one meaning. Obviously, this is not in line with our understanding. Whether the entire dataset is directly constructed as a graph or a sample is constructed as a graph, only using nodes to represent words will lose some useful information.

According to the habit of human beings to obtain text data, for example, when people read a piece of news, they tend to first understand the meaning of each paragraph or sentence, and then combine them to get the full meaning. Therefore, we should let the deep learning model learn the hierar-

chical representation of the data. The research on the application of graph neural networks in text classification tasks, such as the work of Huang et al. (2019), has achieved good results, but it ignores the semantic structure information of text to a certain extent. Therefore, to address these issues, we propose a semantic hierarchical graph neural network (HieGNN) to construct graphs from word-level, sentence-level and document-level, respectively. Because of the existence of word-level, it is possible to make a word have different semantics in different sentences. As shown in Fig.1, perform word segmentation on the input text to obtain word-level graph $\mathcal{G}_1$ and sentence-level graph $\mathcal{G}_2$ and directly construct the input text into document-level $\mathcal{G}_3$ without sentence segmentation. Then use graph neural networks (GNNs) to act on the three levels mentioned above to obtain the output of the three-level vector representations, finally, merge them and input them to the softmax and linear layer for classification.

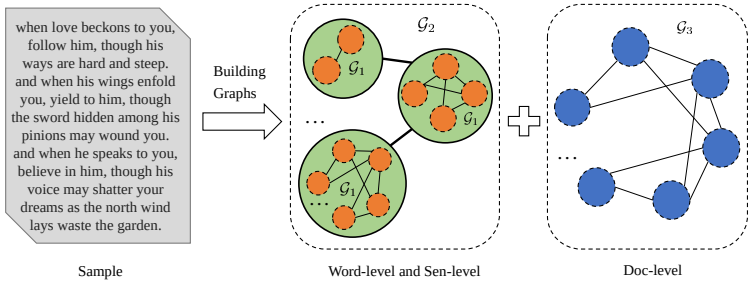

Figure 1: Illustration of HieGNN processing a sample: A sample will use GNN for semantic extraction from three levels: word-level, sen-level and doc-level, and finally the outputs of the three levels will be merged for further classification.

In this paper, we conduct experiments applying graph attention networks (GATs) (Velickovic et al., 2017) at three levels mentioned above. GATs and graph convolutional networks (GCNs) are two different graph neural network models. The main difference is that GCNs use message passing mechanism (MPM) (Gilmer et al., 2017) when aggregating k-hop neighbor nodes, while GAT first uses the attention mechanism (Vaswani et al., 2017) to calculate the weight of the edge and then aggregates. We conducted experiments on 5 datasets, and achieved better or similar results compared with multiple baseline methods, indicating that the HieGNN model proposed in this paper can effectively extract more text information to a certain extent under the condition of using only graph neural networks. The source code will be uploaded to GitHub for reproduction and comparison. In summary, the key contributions of this paper are as follows:

- In this paper, we propose a three-level hierarchical graph neural network model (HieGNN).
- We demonstrate the effectiveness of the proposed model HieGNN to extract textual semantic information from different semantic levels.
- This model is simple and not limited by specific neural networks, and can be generalized to other models and tasks.

2 Related Work

Text classification is one of the most important and fundamental tasks in NLP, it aims to classify new texts into predefined categories based on training samples. The study of text classification has been around for more than 60 years Li et al. (2022b). The related research of graph neural networks (GNNs) also has a history of several decades (Zhou et al., 2020; Wu et al., 2021), but the research on the combination of GNNs and text classification has only emerged in recent years (Malekzadeh et al., 2021). The early GNNs was difficult to popularize and use due to the complexity of the algorithm. Since Kipf & Welling (2017) simplified the complex GNNs, the research and application of GNNs have been in full swing. And due to the flexibility of the graph, the recent performance of graph neural networks on text classification tasks is very attractive (Lin et al., 2021).

Earlier, Yao et al. (2019) have successfully applied the GCNs model to the text classification task, which enables nodes to obtain k-hops neighbors' information by using a nonlinearly connected k-layer GCNs. On the basis, Wu et al. (2019) further demonstrated that the information acquisition of nodes in the GCNs model mainly comes from averaging the information of neighbor nodes, and removes the nonlinear transformation between layers. It significantly improves the training speed of the model and the classification effect does not drop. Huang et al. (2019) proposed a text-level GCN to solve the disadvantage that the general GNN-based models need to construct the entire corpus into a graph, which improves the classification results of GCN and reduces memory consumption to a certain extent by constructing a sample into a graph directly. However, whether it is directly constructing a sample into a graph alone, the structural information of the sample is not fully utilized. That is, words forms phrases, phrases forms sentences, sentences forms samples and even more levels can be divided. Therefore, unlike these methods, our model adopts a three-level semantic hierarchy model, which focuses on the role of the words, sentences and documents, respectively, so that the model can fully utilize the semantic structure information of the samples.

Our work is also inspired by the hierarchical attention network (HAN) proposed by Yang et al. (2016), which uses word and sentence attention mechanisms to find the most important words and sentences for the document classification tasks. The method in this paper also has certain similarities with DIFFPOOL proposed by Ying et al. (2018), the key difference is that DIFFPOOL is a hierarchical pooling of a graph, while this paper decomposes the original large graph from a semantic point of view, and attempts to use three different semantic levels extraction of information. The hierarchical graph mutual information (HGMI) (Li et al., 2022a) and semi-supervised graph classification via cautious/active iteration (SEAL-C/AI) (Li et al., 2019) analyze the advantages of hierarchical graph models from the perspective of social networks, and this paper also has similarities with them, but the details of our model are completely different. At the same time, we noticed that the GAT model proposed by Velickovic et al. (2017) shows the effectiveness of the attention mechanism on graphs, but there is a lack of related research on applying GAT to the text classification task. Therefore, in this paper, GAT is used as a specific GNN model to introduce our model.

3 HIEGNN

In this section, we will introduce our model hierarchical graph neural network (HieGNN) in detail, which aims to utilize the semantic information of text more reasonably from different levels. In general, our model consists of three steps: constructing graphs from samples, applying GNN models, and using a linear classifier to output results.

3.1 GAT

In this subsection, we will use graph attention network (GAT) (Velickovic et al., 2017) as an example to gradually build HieGNN without modifying the original GAT formula. To recap, the words in a sentence in a sample are constructed as a graph at the word-level. For the sen-level, it is to construct a graph of multiple sentences in the sample. For the doc-level, each word in the sample is directly constructed into a graph. Firstly, according to the formula of GAT(Velickovic et al., 2017), we know that on a graph, the update formula of node feature is:

$$\boldsymbol{h}_i^{(l+1)} = \sigma(\sum_{j \in N(i)} \alpha_{ij}^{(l)} \boldsymbol{z}_j^{(l)}) \tag{1}$$

where, $\boldsymbol{h}_i^{(l+1)}$ is the updated vector representation (the $l+1$ layer of GAT) of the i-th node in the graph, σ is an activation function, $N(i)$ is the neighbor set of the i-th node, $\alpha_{ij}^{(l)}$ (attention score) is the weight of the edge between the i-th node and the neighbor node j, and $\boldsymbol{z}_j^{(l)}$ is the vector representation of the j-th node in l layer computed by formula (2). And the calculation formulas for

α_{ij} are as follows:

$$\boldsymbol{z}_i^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{h}_i^{(l)} \tag{2}$$

$$e_{ij}^{(l)} = LeakyReLU(\boldsymbol{a}^{(l)^T}(\boldsymbol{z}_i^{(l)}||\boldsymbol{z}_j^{(l)})) \tag{3}$$

$$\alpha_{ij}^{(l)} = \frac{exp(e_{ij}^{(l)})}{\sum_{k\in N(i)} exp(e_{ik}^{(l)})} \tag{4}$$

Equation (2) is a linear transformation of the eigenvector of the l layer node. "||" is concatenation, and $\boldsymbol{a}^{(l)}$ is a learnable weight vector. With the above knowledge about GAT, here we apply it to the HieGNN model.

3.2 Three Levels on GNN

As shown in Fig.2, firstly, we split and construct the sample into word-level graph $\mathcal{G}_1$, document-level graph $\mathcal{G}_3$ (abbreviated as doc-level), and use the word-level GNN to obtain the vector representation of a single sentence to construct sentence-level graph $\mathcal{G}_2$ (abbreviated as sen-level), and then use the sen-level GNN to obtain the semantics of sample, finally, merge the vector representations of these three levels GNN output into a final vector to represent the sample.

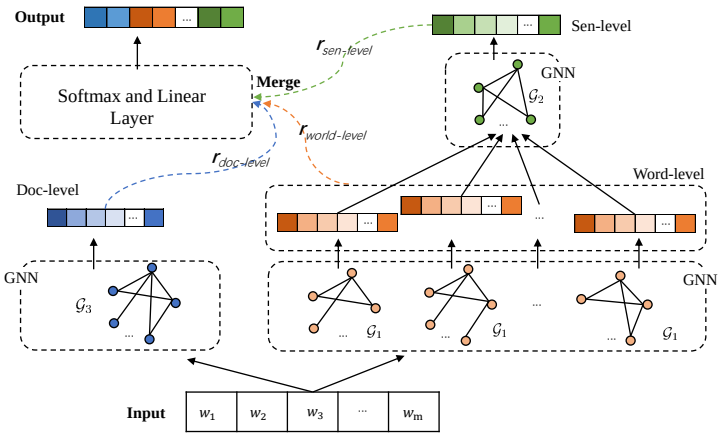

Figure 2: An illustration of the semantics of text extracted hierarchically using GNN: In the word-level $\mathcal{G}_1$, a sentence is built into a graph, and the nodes are word tokens, and the feature of the node is the embedding vector of the word. In the sen-level $\mathcal{G}_2$, the node features are sentence vectors formed by the aggregation of $\mathcal{G}_1$ node features. In the doc-level $\mathcal{G}_3$, an example is treated as a sentence, the nodes also are word tokens, and the node features are word embedding vectors. The existence of the edge of $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$ are determined according to n-grams, and the weight of the edge is obtained according to the adaptive attention mechanism in GAT.

Suppose a sample to be classified is denoted as $X = \{w_1, w_2, \ldots, w_i, \ldots, w_m\}$, w_i and m represent the i-th word and the number of words in the sample respectively. And the sample X will be divided into $S = \{s_1, s_2, \ldots, s_j, \ldots, s_k\}$ according to the punctuation in samples, s_j is the j-th sentence and k is the number of sentences in the sample. Then convert w_i to a vector representation using word embeddings (Mikolov et al., 2013). As shown in Fig.2: For the word-level, construct a graph $\mathcal{G}_1(V_1, E_1)$ on s_j sentence, where V_1 represents the set of nodes, E_1 represents the set of edges. For the sen-level, the graph $\mathcal{G}_2(V_2, E_2)$ is constructed on S and a sentence vector generated by the word-level graph $\mathcal{G}_1$ is used as a node feature. For the doc-level $\mathcal{G}_3(V_3, E_3)$, the construction process is same as Huang et al. (2019), that is, the sample X is directly regarded as a sentence and constructed in a word-level manner. Below we will introduce the above processes one by one.

Word-Level: Observing samples from a word-level perspective, first, we build a global word embedding matrix $\boldsymbol{M}_1 \in \mathbb{R}^{N\times n}$ to store the graph node features based on the current dataset of this sample, where N means the number of words in the vocabulary of this dataset, n is the dimension

of node eigenvector. We do not save edge weights because, in GAT, edge weights are adaptively determined according to the node-to-node attention scores, see equation (2)-(4). As mentioned above, then the sample X is divided into multiple sentences $S = \{s_1, s_2, \ldots, s_j, \ldots, s_k\}$ based on punctuation, $s_i = \{w_1, w_2, \ldots, w_j, \ldots, w_{m'}\}$, k and m' represent the number of sentences in a sample and the number of words in a sentence, respectively. For every sentence, according to the $\boldsymbol{M}_1$ matrix, the word w_j is converted into the initial feature vector $\boldsymbol{h}_j^{(0)}$, and the edge between nodes is determined by the n-gram (refer to Eq. (7) below). After the graph $\mathcal{G}_1(V_1, E_1)$ is constructed for every sentence, the graph node is updated according to equation (1). The eigenvector of s_i sentence is obtained according to the following formula:

$$\boldsymbol{r}_i^{(l+1)} = R(\boldsymbol{h}_j^{(l)} | j \in \mathcal{G}_1) \tag{5}$$

In Equation (5), R is a readout function (Gilmer et al., 2017), the common ones are mean, max and sum, for example, if it is mean, then $\boldsymbol{r}_i^{(l+1)} = \dfrac{1}{m'} \sum_{j=1}^{m'} \boldsymbol{h}_j^{(l)}$. Finally, each r_i is merged by mean operation, that is, the first output about the sample at the word-level is obtained:

$$\boldsymbol{r}_{word-level}^{(l+1)} = \frac{1}{k} \sum_{i=1}^{k} \boldsymbol{r}_i^{(l)} \tag{6}$$

where, $\boldsymbol{r}_{word-level}^{(l+1)}$ is the word-level output (See the orange-yellow vector in Fig. 2).

The role of word-level: As mentioned in section 1, people's habit of reading text and extracting information should first analyze each sentence and of the text in turn, then find out the key information in each sentence, and then synthesize the information found. Few people will directly check word by word, find out important word information directly from the text, and synthesize it. Therefore, it is necessary to add word-level graphs to comprehensively consider the above factors.

Sen-Level: Analyzing the text from sentence-level, if we treat the sample directly as a sentence like Huang et al. (2019), then sentences with important information will be regarded as equally important. Therefore, we partition the samples into sentences S (as mentioned above) and build the $\mathcal{G}_2(V_2, E_2)$ as shown in figure 2. It is no longer necessary to use a global matrix to store information on the sen-level because the initial feature vector of the $\mathcal{G}_2$ node is obtained first, and then GAT is used again on $\mathcal{G}_2$ to update the node and adjust the value of the word embedding vector in $\boldsymbol{M}_1$ (See the word-level paragraph above). Adjacency matrix $A \in \mathbb{R}^{k \times k}$ (k is the number of sentences in a sample) of $\mathcal{G}_2$ is used to store the connection relationship of the sentence node s_j. It is determined by n-grams. For example, for 2 grams:

$$\boldsymbol{A}_{i,j} = \begin{cases} 1, & |i-j| \leq 2 \\ 0, & otherwise \end{cases} \tag{7}$$

which indicates that if the distance between two nodes is no greater than 2, they are connected. The number '1' only means that there is an edge between node i and j, and '0' means there is no edge. The edge weight is obtained by Equation (4).

The eigenvalue of $\mathcal{G}_2$ node is the features aggregation of the nodes at the word-level is $\boldsymbol{r}_i$ (See Eq. (5)), and the node update formula of sen-level is:

$$\boldsymbol{s}_i^{(l+1)} = \sigma\Big(\sum_{k \in N(i)} \alpha_{ik}^{(l)} \boldsymbol{r}_k^{(l)}\Big) \tag{8}$$

where, $\boldsymbol{s}_i^{(l+1)}$ is the updated node feature of node i, and $N(i)$ is the neighbor node of node i. For the transformation of the initial node $\boldsymbol{r}_i$ from $\mathcal{G}_1$ we use two new parameters $\boldsymbol{W}_s$ and $\boldsymbol{b}$, because the model learns different information at sen-level and word-level. Here is the transform:

$$\boldsymbol{r}_i^{(l)} = \boldsymbol{W}_s^{(l)} \boldsymbol{r}_i^{(l)} \tag{9}$$

$$e_{ij}^{(l)} = LeakyReLU(\boldsymbol{b}^{(l)^T} (\boldsymbol{z}_i^{(l)} || \boldsymbol{z}_j^{(l)})) \tag{10}$$

Finally, merge the graph nodes feature to obtain the second output of the sample at sen-level:

$$\boldsymbol{r}_{sen-level}^{(l+1)} = \frac{1}{k} \sum_{i=1}^{k} \boldsymbol{s}_i^{(l)} \tag{11}$$

The role of the sen-level solves the disadvantage that a word has only one vector representation from beginning to end in a classification task (Yao et al., 2019; Huang et al., 2019), and at the same time obtains different semantic level information of the text.

Doc-Level: Observing the sample from the doc-level, no matter how many sentences there are in the sample, it is treated as a long sentence, that is, $X = \{w_1, w_2, \ldots, w_i, \ldots, w_m\}$, which is also the practice of Huang et al. (2019). In the HieGNN model, we keep this part because this layer is very useful in extracting textual information, this can be verified in the experimental section below.

For the construction of graph $\mathcal{G}_3(V_3, G_3)$, it is roughly the same as the construction of $\mathcal{G}_1$ in word-level, and also constructs a global matrix $\boldsymbol{M}_2 \in \mathbb{R}^{N \times n}$, and according to the n-gram to determine the node connection relationship. The application of GAT is the same as the word-level.

Finally, the doc-level output extracted from the sample is obtained by:

$$\boldsymbol{r}_{doc-level}^{(l+1)} = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{h}_i^{(l)} \tag{12}$$

3.3 Model Outputs

As shown in Fig. 2, first convert the n-dimension of $\boldsymbol{r}_t$ to C-dimension using a linear layer to get the score for each category x_i ($t \in \{d, s, w\}$, d, s and w represent the output of doc-level, sen-level and word-level respectively. C is the number of categories in the dataset). Then the $\boldsymbol{r}_t \in \mathbb{R}^{1 \times C}$ is further transformed by the softmax and log function $\ln(\frac{exp(x_i)}{\sum_j^C exp(x_j)})$ to obtain $\boldsymbol{r}_t^{'} \in \mathbb{R}^{1 \times C}$. Then the outputs on the three levels are combined:

$$\hat{\boldsymbol{y}} = \lambda_d \boldsymbol{r}_d^{'} + \lambda_s \boldsymbol{r}_s^{'} + \lambda_w \boldsymbol{r}_w^{'} \tag{13}$$

where, $\hat{\boldsymbol{y}} \in \mathbb{R}^{1 \times C}$, λ_t represents the weight of the output at the corresponding level, and $\lambda_d + \lambda_s + \lambda_w = 1$ ($\lambda_t \in (0, 1), t \in \{d, s, w\}$).

In order not to increase the complexity of the model, we only associate λ with the number of sentences (denoted as x_s) in the sample, $\lambda_s = \frac{1}{2}(1 - \lambda_d)$ and $\lambda_w = \frac{1}{2}(1 - \lambda_d)$ decrease as the number of sentences increases, λ_d is:

$$\lambda_d = \frac{1}{\ln(x_s) + 1} \tag{14}$$

The design motivation of Eq. (14): give less weight to word-level and sen-level with large number of sentences, this is because we prefer the model to extract the structural information in each sentence, and pay less attention to the words themselves, on the contrary, giving more weight to doc-level makes it possible to directly extract important word information from the samples.

3.4 Optimization Goal

After obtaining the output $\hat{\boldsymbol{y}}$, we use the cross-entropy loss function to optimize the model:

$$\mathcal{L} = -\frac{1}{N} \sum_{i}^{N} \sum_{k}^{C} \boldsymbol{y}_{ik} \ln(\hat{\boldsymbol{y}}_{ik}) \tag{15}$$

In Eq. (15), N is the number of samples in the mini-batch, C is the number of categories, the elements of $\boldsymbol{y}$ are the true labels, and the elements of $\hat{\boldsymbol{y}}$ are the labels predicted by the model. Adjust the matrix $\boldsymbol{M}_1, \boldsymbol{M}_2$ (as mentioned above) and the parameters of the GAT model at each level according to the minimum loss $\mathcal{L}$, see Eq. (1)-(4).

3.5 Time Complexity

Since the HieGAT proposed in this paper reuses GAT model many times, it is necessary to analyze their time complexity. The core of the HieGAT model is GAT, which can be viewed as a multiple reuse of GAT. The main operation of GAT is Eq. (2)-(4). For one sample, assuming that H is used to represent the dimension of the graph node vector, and the dimension of the node vector does not

change after linear transformation. So the complexity of Eq. (2) is $\mathcal{O}(H^2)$, since each node of the graph needs to be calculated, it is $\mathcal{O}(|V| \times H^2)$ ($|V|$ is the number of graph nodes). Eq. (3) is a mapping function that maps a $2H$-dimensional vector to a real number, so its complexity is $\mathcal{O}(H)$. Eq. (4) needs to calculate the attention score for each edge of the graph, so the complexity is $\mathcal{O}(|E| \times H)$ ($|E|$ is the number of edges of the graph). To sum up, the complexity of the GAT model is $T(|V|, |E|, H) = \mathcal{O}(|V| \times H^2 + |E| \times H)$.

For the three-level graph on HieGAT proposed in this paper, assuming that a sample is divided into $|S|$ sentences, each sentence has $|W|$ words, then the complexity of the doc-level GAT is $T_d(|S|, |W|, H) = \mathcal{O}((|S| \times |W| \times H^2 + |E_3| \times H))$. Similarly, we get sen-level complexity $T_s(|S|, |W|, H) = \mathcal{O}(|S| \times H^2 + |E_2| \times H)$ and word-level complexity $T_w(|S|, |W|, H) = \mathcal{O}(|S| \times |W| \times H^2 + |E_1| \times H)$. Since the average number of sentences per sample $|S|$ is very small (see Table 1) and n-gram are used to build the graph (See Eq. (7)), the number of edges in the graph is much smaller than $|V| \times (|V| - 1)$ (according to the relationship between the number of vertices and edges of the graph). Assuming that $|E| \propto |V|$, then $|E_3| > |E_1|$, so $T_d > T_w \gg T_s$. And because the doc-level and word-level are two computations that can be performed at the same time, the total time complexity is a little bit larger than T_d.

4 EXPERIMENTS

In the previous sections, we introduced the implementation details of HieGNN using GAT as a specific model. In this section, we will use HieGAT, BERTHieGAT and RoBERTaHieGAT to conduct experiments on 5 common datasets and compare them with pure GNN-based models and hybrid models of BERT and GNN to test the effectiveness of the three-level semantic extraction method (HieGNN) proposed in this paper.

4.1 BASELINE METHODS

These baseline models we selected are the current models that apply GNN in the field of text classification, mainly TextGCN, SGC, Text-Level-GCN, BERTGAT and RoBERTaGAT, etc. It should be pointed out that, except the Text-Level-GCN and HieGAT in this paper, which are graph-level classification models, the others are node-level classification models.

- **TextGCN** (Yao et al., 2019): The model mainly completes two tasks: the first is graph node classification and the second is graph representation learning. By constructing a dataset into a graph, the vector representation of each document node is learned. It became one of the best graph neural network models for classification at that time.
- **SGC** (Wu et al., 2019): This model mainly simplifies the current GCN model (such as FastGCN Chen et al. (2018)), which accelerates the model inference speed without reducing the model classification effect. Even better results are achieved on some datasets compared with TextGCN.
- **Text-Level-GCN** (Huang et al., 2019): This model mainly models a single sample into a graph, saves global information through word embedding matrix and edge weight matrix, and finally mergers graph nodes into a vector to represent the sample, achieves a good result.
- **BERTGAT and RoBERTaGAT**(Lin et al., 2021): It combines BERT (Devlin et al., 2019) (RoBERTa (Liu et al., 2019)) with GAT by $\mathbf{Z} = \lambda \mathbf{Z}_{GAT} + (1 - \lambda)\mathbf{Z}_{BERT}$, where $\mathbf{Z}$ is the prediction score for each class, which is obtained by passing the vector representation of the sample through softmax function. That is, the output of BERT (or RoBERTa) and GAT are synthesized according to a certain ratio λ, and the final comprehensive prediction result is obtained. It is noted that hybrid model of BERT and GCN in the work of the paper gives the best results, but our work only implementations the BERTHieGAT and RoBERTaHieGAT model due to limited time, so only BERTGAT and RoBERTaGAT are compared here.

4.2 DATASETS

In order to compare the performance of GNN and HieGNN on text classification tasks, we choose the same datasets as Yao et al. (2019), Huang et al. (2019), Lin et al. (2021) and Wu et al. (2019).

The datasets can be found here[1] and the overall datasets statistics information as shown in Table 1, it should be pointed out that for the R8 and R52 datasets, the original data with punctuation has been lost, and only the cleaned data with punctuation removed is retrained in the dataset used now. This paper uses the NNSplit [2] library for sentence segmentation of unpunctuated text. The 20NG, Ohsumed, and MR datasets use NLTK [3] for sentence segmentation, and all datasets use NLTK for word segmentation. For more detailed information about these datasets, refer to Yao et al. (2019).

Table 1: Datasets statistics. Except the last column is newly added, other columns are taken from Yao et al. (2019). '#' means quantity, "Avg. L" and "Avg. S" denote the average length and the average number of sentences of each sample, respectively.

Dataset	#Docs	#Training	#Test	#Words	#C	Avg. L	Avg. S
20NG	18,846	11,314	7,532	42,757	20	221.26	4.89
R8	7,674	5,485	2,189	7,688	8	65.72	6.24
R52	9,100	6,532	2,568	8,892	52	69.82	6.29
Ohsumed	7,400	3,357	4,043	14,157	23	135.82	9.02
MR	10,662	7,108	3,554	18,764	2	20.39	1.19

4.3 Experimental Setup

The experiments are mainly done using the Deep Graph Library (DGL) [4] in PyTorch, and all experiments are done on the Google Colab platform using a Tesla P100 GPU with 16GB of RAM. Use a 3-layer and 3-head GAT on the doc-level, and a 1-layer and 1-head GAT on the sen-level and word-level. The batch size is 64 and should not be set too large. (Large batches will affect the calculation of average sentence number, see Eq. (14).). The activation function between GAT layers uses ELU, the negative slope of LeakyReLU angle is 0.2, and dropout rate is 0.5. The learning rate on MR dataset is 0.0001, the others are 0.001. All parameters are not guaranteed to be optimal and can be adjusted according to specific circumstances. In order to compare with related work, the accuracy metric is used, which means the number of correct classifications divided by the total number of samples.

4.4 Experimental Results

We first compared with the pure GNN model, and the experimental results are shown in Table 2. It can be seen that the effect of the HieGAT model has a certain improvement on R8 and MR datasets. And outperforms TextGCN and SGC on four out of five datasets.

Table 2: Experimental results on 5 datasets, accuracy metric is adopted. We run 5 times and report mean results. Except for the bottom data, other data are taken from the results in the baseline papers. '-' indicates that the original paper did not provide the result.

Model	20NG	R8	R52	Ohsumed	MR
TextGCN	0.8634	0.9707	0.9356	0.6836	0.7674
SGC	**0.885**	0.972	0.94	0.685	0.759
TextLevelGCN	-	0.978	**0.946**	**0.6994**	-
HieGAT	0.8584	**0.9783**	0.9454	0.6984	**0.7804**

In addition, inspired by the work of Lin et al. (2021), we combined the proposed HieGAT with BERT and RoBERTa to obtain RoBERTaHieGAT and BERTHieGAT, and compared them to verify the effectiveness of the model. The experimental results are shown in Table 3. We can see that the model achieves better results on the R8 and Ohsumed datasets. It shows that the HieGAT model can effectively improve the classification effect when combined with other classification models.

[1] https://github.com/yao8839836/text_gcn/tree/master/data

[2] https://bminixhofer.github.io/nnsplit/

[3] https://www.nltk.org/

[4] https://www.dgl.ai/

Table 3: Comparing with mixed models based on BERT and GAT.

Model	20NG	R8	R52	Ohsumed	MR
BERT	0.853	0.978	0.964	0.705	0.857
RoBERTa	0.838	0.978	0.962	0.707	**0.894**
BertGAT	**0.874**	0.978	0.965	0.712	0.865
RoBERTaGAT	0.865	0.980	0.961	0.712	0.892
BERTHieGAT	0.8505	0.9783	**0.9657**	0.7071	0.8603
RoBERTaHieGAT	0.8538	**0.9805**	0.9613	**0.7249**	0.8921

On the proposed HieGAT model, in addition to the above comparison experiments, we also performed an ablation experiment to verify whether the hierarchical structure really works. Compared with HieGAT, only the value of λ_t ($t \in \{d, s, w\}$) is changed, and other parameters remain unchanged. The main comparative experiments on 5 datasets are: (1) Use only one of the three levels at a time (i.e. set one coefficient to 1 and the other two to 0). (2) Use only two of the three levels at a time (i.e. set one coefficient to 0 and the other two to non-zero).

The results of the ablation experiments are shown in Table 4, in which the top column is the use of only one level, the middle column is the use of only two levels, and the bottom column is the result of using three levels. We can see that it is difficult to achieve the effect of HieGAT whether only one level of GAT is used alone or two levels of GAT are used. Therefore, processing the input text from a three-level perspective proposed in this paper can extract more information, thereby improving the classification performance of the original GAT model.

Table 4: Ablation experiments on HieGAT.

λ	20NG	R8	R52	Ohsumed	MR
$\lambda_d = 1, \lambda_{s,w} = 0$	0.8420	0.974	0.9364	0.6634	0.7731
$\lambda_s = 1, \lambda_{d,w} = 0$	0.8064	0.9653	0.9124	0.6667	0.7597
$\lambda_w = 1, \lambda_{d,s} = 0$	0.8033	0.9691	0.9294	0.6696	0.7640
$\lambda_d = 0, \lambda_{s,w} \neq 0$	0.8062	0.9679	0.9204	0.6714	0.7586
$\lambda_s = 0, \lambda_{d,w} \neq 0$	0.8551	0.9751	0.9441	0.6882	0.7659
$\lambda_w = 0, \lambda_{d,s} \neq 0$	0.8553	0.9760	0.9392	0.6863	0.7690
HieGAT($\lambda_{d,s,w} \neq 0$)	**0.8584**	**0.9783**	**0.9454**	**0.6984**	**0.7804**

5 CONCLUSION

This paper proposes a hierarchical graph neural network (HieGNN) to solve the problem of insufficient information extraction (ie, whether the dataset corpus or the sample is directly constructed into a graph, it ignores the progressive relationship of words that form sentences and sentences that form samples, and we can also further divide them into more levels) of input text by current graph neural networks (GNNs) models. By constructing a sample into three levels of corresponding graphs, the progressive semantic information of the input sample (ie, the semantics of words, the semantics of sentences, and the semantics of documents) is effectively extracted. At the same time, since our model is not related to specific GNNs, it can be easily combined with other models.

The experimental results show that our model can achieve good results on both the pure GNN model and the hybrid model, and the analysis of the computational complexity of the model shows that the complexity is slightly larger than the original model, but this can be solved using a parallel computing mechanism. However, although the hierarchical structure effectively extracts textual semantics, the effect of improvement is very limited, which shows that the simple segmentation sentence construction graph needs to be improved. In addition, the work of this paper only conducts experiments on the GAT-based model, not on the GCN-based model, we will complete this part in the follow-up work.

REFERENCES

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rytstxWAW.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272. PMLR, 2017. URL http://proceedings.mlr.press/v70/gilmer17a.html.

Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. Text level graph neural network for text classification. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 3442–3448. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1345. URL https://doi.org/10.18653/v1/D19-1345.

Yoon Kim. Convolutional neural networks for sentence classification. In Alessandro Moschitti, Bo Pang, and Walter Daelemans (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1746–1751. ACL, 2014. doi: 10.3115/v1/d14-1181. URL https://doi.org/10.3115/v1/d14-1181.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Jia Li, Yu Rong, Hong Cheng, Helen Meng, Wen-bing Huang, and Junzhou Huang. Semi-supervised graph classification: A hierarchical graph perspective. In Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (eds.), *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pp. 972–982. ACM, 2019. doi: 10.1145/3308558.3313461. URL https://doi.org/10.1145/3308558.3313461.

Jia Li, Yongfeng Huang, Heng Chang, and Yu Rong. Semi-supervised hierarchical graph classification. *CoRR*, abs/2206.05416, 2022a. doi: 10.48550/arXiv.2206.05416. URL https://doi.org/10.48550/arXiv.2206.05416.

Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. A survey on text classification: From traditional to deep learning. *ACM Trans. Intell. Syst. Technol.*, 13(2):31:1–31:41, 2022b. doi: 10.1145/3495162. URL https://doi.org/10.1145/3495162.

Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li, and Fei Wu. Bertgcn: Transductive text classification by combining GNN and BERT. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pp. 1456–1462. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-acl.126. URL https://doi.org/10.18653/v1/2021.findings-acl.126.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692.

Masoud Malekzadeh, Parisa Hajibabaee, Maryam Heidari, Samira Zad, Özlem Uzuner, and James H. Jones. Review of graph neural network in text classification. In *12th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference, UEMCON 2021, New York, NY, USA, December 1-4, 2021*, pp. 84–91. IEEE, 2021. doi: 10.1109/UEMCON53757.2021.9666633. URL https://doi.org/10.1109/UEMCON53757.2021.9666633.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun (eds.), *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. URL http://arxiv.org/abs/1301.3781.

Ankit Pal, Muru Selvakumar, and Malaikannan Sankarasubbu. MAGNET: multi-label text classification using attention-based graph neural network. In Ana Paula Rocha, Luc Steels, and H. Jaap van den Herik (eds.), *Proceedings of the 12th International Conference on Agents and Artificial Intelligence, ICAART 2020, Volume 2, Valletta, Malta, February 22-24, 2020*, pp. 494–505. SCITEPRESS, 2020. doi: 10.5220/0008940304940505. URL https://doi.org/10.5220/0008940304940505.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In Pierre-Antoine Champin, Fabien Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis (eds.), *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pp. 1063–1072. ACM, 2018. doi: 10.1145/3178876.3186005. URL https://doi.org/10.1145/3178876.3186005.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In Marilyn A. Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pp. 2227–2237. Association for Computational Linguistics, 2018. doi: 10.18653/v1/n18-1202. URL https://doi.org/10.18653/v1/n18-1202.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *CoRR*, abs/1710.10903, 2017. URL http://arxiv.org/abs/1710.10903.

Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6861–6871. PMLR, 2019. URL http://proceedings.mlr.press/v97/wu19e.html.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 32 (1):4–24, 2021. doi: 10.1109/TNNLS.2020.2978386. URL https://doi.org/10.1109/TNNLS.2020.2978386.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver,*

BC, Canada, pp. 5754–5764, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. Hierarchical attention networks for document classification. In Kevin Knight, Ani Nenkova, and Owen Rambow (eds.), *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pp. 1480–1489. The Association for Computational Linguistics, 2016. doi: 10.18653/v1/n16-1174. URL https://doi.org/10.18653/v1/n16-1174.

Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 7370–7377. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33017370. URL https://doi.org/10.1609/aaai.v33i01.33017370.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 4805–4815, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/e77dbaf6759253c7c6d0efc5690369c7-Abstract.html.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 649–657, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/250cf8b51c773f3f8dc8b4be867a9a02-Abstract.html.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020. doi: 10.1016/j.aiopen.2021.01.001. URL https://doi.org/10.1016/j.aiopen.2021.01.001.