Graph Convolutional Networks with Dependency Parser towards Multiview Representation Learning for Sentiment Analysis

1st Minqiang Yang
School of Information Science and Engineering
Lanzhou University
Lanzhou, China
yangmq@lzu.edu.cn

2nd Xinqi Liu
School of Information Science and Engineering
Lanzhou University
Lanzhou, China
xqliu2019@lzu.edu.cn

3rd Chengsheng Mao
Department of Preventive Medicine, Feinberg School of Medicine
Northwestern University
Chicago, USA
chengsheng.mao@northwestern.edu

4th Bin Hu*
School of Information Science and Engineering
Lanzhou University
Lanzhou, China
bh@lzu.edu.cn

2022 IEEE International Conference on Data Mining Workshops (ICDMW) | 979-8-3503-4609-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDMW58026.2022.00070

***Abstract*—Sentiment analysis has become increasingly important in natural language processing (NLP). Recent efforts have been devoted to the graph convolutional network (GCN) due to its advantages in handling the complex information. However, the improvement of GCN in NLP is hindered because the pretrained word vectors do not fit well in various contexts and the traditional edge building methods are not suited well for the long and complex context. To address these problems, we propose the LSTM-GCN model to contextualize the pretrained word vectors and extract the sentiment representations from the complex texts. Particularly, LSTM-GCN captures the sentiment feature representations from multiple different perspectives including context and syntax. In addition to extracting contextual representation from pretrained word vectors, we utilize the dependency parser to analyse the dependency correlation between each word to extract the syntax representation. For each text, we build a graph with each word in the text as a node. Besides the edges between the neighboring words, we also connect the nodes with dependency correlation to capture syntax representations. Moreover, we introduce the message passing mechanism (MPM) which allows the nodes to update their representation by extract information from its neighbors. Also, to improve the message passing performance, we set the edges to be trainable and initialize the edge weights with the pointwise mutual information (PMI) method. The results of the experiments show that our LSTM-GCN model outperforms several state-of-the-art models. And extensive experiments validate the rationality and effectiveness of our model.**

***Index Terms*—Sentiment Analysis, Multiview Representation Learning, GCN, LSTM, Dependency Parser**

I. Introduction

Sentiment analysis aims at learning and analyzing text representations to classify the sentiment polarity of the given text. It has become an important topic in natural language processing [1], [2] for its wide range of academic and commercial applications including business, marketing and advertising. Therefore, different methods and algorithms have been proposed in recent years for the analysis of the text emotional polarity. Sentiment analysis is a branch of the text classification and due to its vast range of academic and industrial applications, it obtains huge attention.

The traditional methods [3], [4], [5] to solve the sentiment analysis task usually focus on the handcraft features but ignore the importance of modeling contextual semantics. Recently, the machine learning methods have been introduced to implicitly model the semantic relationship of the context to extract the information for the prediction of the text sentimental tendency. These methods can be divided into three main categories including the statistics-based methods such as the Naïve Bayesian classifier [6], [7], the rule-based methods [8], [9] and the deep learning methods [10], [11].

The essential part in solving the sentiment analysis task is to learn the text representations which are influenced by the context. Previous studies have proposed different neural network models such as the back propagation neural network (BPNN) [12], convolutional neural network (CNN) [13] and long short term memory (LSTM) [10] model to learn useful information from the text. However, an inherent defect makes these model hard to extract the text representations effectively because these models cannot obtain the structure of the sentence to update the information. Later, the tree-structure was introduced for the description of the texts to obtain the hierarchical information and the Tree-LSTM [14], tree-CNN [15] and SDT-CNN [16] came into being. The introduction of the tree structure optimizes the previous models. However, there are still some problems [17] to be dealt with: (1). The training speed of tree-LSTM is too slow. (2). The tree-CNN cannot obtain the semantic information. (3). The SDT-CNN

* Corresponding author: Bin Hu

2375-9259/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDMW58026.2022.00070

only get the dependency relationship within the sentences instead of the whole text.

More recent efforts [18], [19] have been devoted into the Graph neural networks (GCNs) which extract the text representations with the exploitation of the sentence structure. A widely used method for building edges is n-gram [20], [21] which connects the words with their nearest *n* words. Besides, the GCNs can not only build edges between the words in the sentence, but also establish the connection between the sentences within the text which address the disadvantage of SDT-CNN. However, there are still two challenges when applying the traditional GCNs to the sentiment analysis task: (1). The pre-trained word vectors do not fit well in various context. (2). The graph constructed with the conventional methods cannot transmit information effectively when encountering complex texts such as the sentences which are longer than 200 words and consist of multiple clauses.

To address the aforementioned challenges, the BiLSTM was incorporated into our model and we introduce the dependency parser to improve the edge construction method. For the first challenge, we utilize the BiLSTM as sentence encoder to further analyze the hidden contextual information. The processed word vectors will be more in line with the context semantics in a specific context. For the second challenge, the dependency parser was introduced to analyze the dependency correlation between words to extract syntax representation which will be used for edge construction and these edges represent the semantic correlation. The purpose of this method is to directly connect the related words for the delivery of the syntax information even if there are many intermediate words between these words. The code of our model is available in Github[1].

Moreover, we utilize the Message Passing Mechanism (MPM) [22] for convolution to extract information from local features. With MPM, the node can obtain information from the adjacent nodes and update its own representation. Even if it is polysemous, we can update its meaning based on contextual information. Besides, for the edge weights of the graph, we introduce the pointwise mutual information (PMI) method to measure the correlation between two words to initialize the weights of the edges. Additionally, because in the process of MPM, the update of node information also depends on the weight of the edge, we set the edge weights to be trainable for better information transmission.

The summary of our contribution is as follows:

- We propose a LSTM-GCN model to analyse the sentiment polarity of the texts based on multiple different perspectives. We contextualize word vectors with the BiLSTM to capture the context representation. And the dependency parser is introduced to analyse the dependency correlation of the text to extract syntax representation. Based on the context and syntax representation, we build text level graph for GCN.
- We utilize the Message Passing Mechanism (MPM) for convolution to extract information from local features. Especially, to fit the graph structure of the sentence, we introduce the graph convolutional method for the aggregation of the message. With MPM, the node can update its representation under the influence of neighbor nodes.

[1]The code repository can be found here: https://github.com/Goat-L/LSTM-GCN

II. Related Work

A. Sentiment Analysis

According to the granularity of text processing, sentiment analysis can be roughly divided into three research levels: text level, sentence level and aspect level. The goal of the text level sentiment analysis is to automate the task of classifying the overall sentiment bias or polarity of the text (e.g., a complete online review) into positive or negative opinion [23], [19]. Sentence level sentiment analysis aims at dividing sentences into different types, and then performs sentiment analysis separately on different kinds of sentences [24], [25]. The aspect level sentiment analysis is also named aspect-based sentiment analysis. This method performs fine-grained analysis to determine the sentiment strength towards various aspects in the sentence [26], [27]. It will analyze the sentiment towards certain words in the sentence. Moreover, recent researches [28] also focus on the analysis based on facial information.

In recent years, machine learning has played an important role in the field of sentiment analysis. To deal with this task, two of the most frequently used algorithms are recurrent neural network (RNN) [29] and CNN [13]. Based on the RNN method, the text-LSTM [10] was proposed to solve the text classification task. Besides, the tree structure was introduced to encode the sentences to extract information. Tree-LSTM [14] was introduced to obtain the hierarchical information for sentiment analysis. However, this structure will slow down the speed of the model training process. After that, reference [15] introduced the tree structure into the CNN and proposed a tree-based convolutional neural network (TB-CNN). Later, the semantic dependency tree-based CNN (SDT-CNN) [16] was proposed to solve this problem, because the tree structure constructed in this method contains the semantic dependency of the sentence. However, the SDT-CNN model can only extract the dependency within the sentences. In text level sentiment analysis, a disadvantage is that this model does not take the sentence relationship in the text into account. Different from existing methods, our proposed graph construction method takes into account not only the word dependencies within the sentences but also the sentence correlation in text.

B. Graph Convolutional Network

GCN is a variant of conventional convolutional neural networks (CNNs) and it performs directly on graphs [30]. GCN is based on CNN and graph structure. So, when processing the graph structured data [20], [21], a GCN directly utilize the convolutional operations on directly connected neighbor nodes to update the information. In the text-GNN [18], each node can update its representation from its neighbor nodes

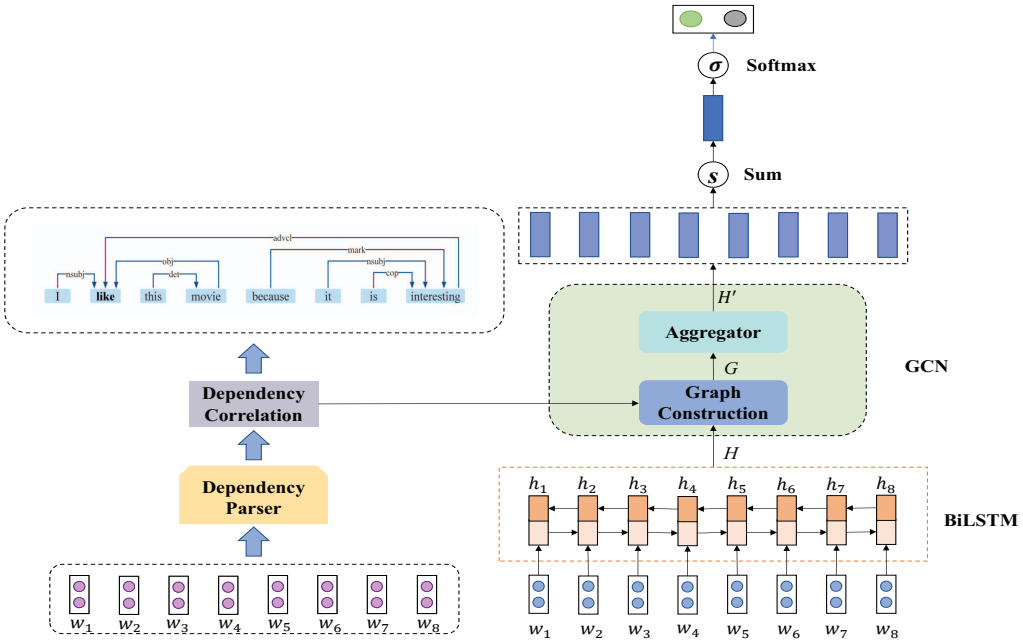

Fig. 1. This is the overview of our LSTM-GCN model. In the process of graph construction, the GCN takes the dependency correlations of the words produced by dependency parser and the hidden states produced by BiLSTM as inputs. The aggregator will encourage the nodes in the text level graph to update their representation based on their neighbor nodes. Details of the architecture and operations are in the method section.

with the message passing. However, there is a disadvantage of these graph construction methods. They connect the node with its neighbor nodes to build the edges. But if there are multiple intermediate nodes between two important related nodes, then the message passing between these two words will definitely be attrition, resulting in inaccurate update of the node representations. Therefore, we introduce the dependency parser [31] to select the word pairs with dependency relationship and build edges between them. Particularly, different with the previous methods [32] [33] which use the whole dependency tree, our model removes meaningless relationships including punctuation, determiner and so on. And for the implementation of GCN layers, we introduce the GraphSAGE layers [34].

III. Method

We propose the LSTM-GCN architecture to analyse the sentiment feature representations based on the contextual and syntax representations. In this section, we will show our model in detail.

A. Model Architecture

Figure 1 shows the main structure of the LSTM-GCN model. To process the word vector to make it more suitable for its current context, we introduce the BiLSTM. In this task, we notate a sentence with m words as $S = \{w_1, ...w_i, ...w_m\}$ in which w_i represents the i_{th} word. We initialize the words as vectors with d dimension word embedding which can be updated by training. In this step, we use 300-dimentional GloVe [35] word embeddings. Then, we utilize BiLSTM model as a sentence encoder to extract hidden contextual representations to contextualize the initial representations. Firstly, we build a vocabulary of V words whose frequency of occurrence in the dataset is greater than 10, and then, we map the pretrained word vectors in GloVe to this vocabulary to get the word embedding table $T \in \mathbb{R}^{|V| \times d_e}$, where $|V|$ represents the size of the vocabulary and d_e represents the dimensionality of the word embeddings. For each sentence S, we can obtain the word embeddings from this table. Secondly, the BiLSTM model is utilized as a sentence encoder to extract hidden contextual representations. Then, we feed the word embeddings into the BiLSTM model to produce the hidden state vectors $H = \{h_1, ...h_i, ...h_m\}$, where the hidden state vector at time t from the BiLSTM is $h_i \in \mathbb{R}^{2d_e}$ and d_e is the dimensionality of the vector which is the output of the unidirectional LSTM. The BiLSTM model will combines forward and reverse processed vectors together.

To capture more sentiment feature representations, we utilize the GCN networks because it can associate edges not only between the words, but also between the sentences. It takes the graph as input which consists of the nodes and edges $G = \langle N, E \rangle$, the features for all nodes $x_n, \forall n \in N$ and the edge weights for all edges $w_e, \forall w \in E$. The nodes represent the words in the sentence. Obtaining the output of word vectors from the BiLSTM, we utilize them as the features for all nodes. Other details of construction of text level graph are in III-B.

However, the simple GCN networks cannot effectively extract semantic information from complex texts. To address this challenge of obtaining information from the graphs that

contains rich node attribute information, we introduce the concept of aggregator in GraphSAGE [34] that aggregates feature information from a node's local neighborhood. Instead of mean, LSTM and pooling aggregator, we choose to use GCN aggregator which is inspired by the semi-supervised GCN [30] proposed by Thomas N. Kipf and Max Welling. The GCN aggregator will perform spectral convolutions on graphs to aggregate the messages. Compared to the traditional mean aggregation method, the GCN aggregator scales linearly in the number of graph edges and learns hidden layer representations that encode both local graph structure and features of nodes, which means that it is more suitable for graph-structured data and it has a larger expressive capability.

We use K to represent the number of the search depth of the graph. In the process of aggregating the feature information, with K aggregators $(aggregator_k, \forall k \in \{1, ..., K\})$, the GCN model aggregate information from node neighbors with a set of weight matrices $(w_k, \forall k \in \{1, ..., K\})$ which is used to propagate information between different layers. We use k to notate the current search depth, h^k to notate the node's hidden states (representations) in this depth and $P(n)$ to notate the neighbor nodes of node *n*. Firstly, each node $n \in N$ aggregates the states from its neighbor nodes, $\{h_u^{k-1}, u \in P(n)\}$, into a single vector $h_{P(n)}^{k-1}$. Then, the GCN networks concatenates the current state of the node h_n^{k-1}, with the aggregated neighborhood vector $h_{P(n)}^{k-1}$. The propagation rules will be discussed in III-C. And this combined vector is fed through a fully connected layer with nonlinear activation function σ. Through all the search depth, we get the final representations of all nodes.

B. Construction of Text Level Graph

Text level graph construction is an essential part since the structure of the graph will determine the aggregation and extraction of information. Besides the traditional method of building edges with the neighbors of the node, we propose the method of building edges between the nodes which have dependency correlation.

1) Nodes of Graph: To build the graph for a given sentence $S = \{w_1, ...w_i, ...w_m\}$, we regard the words that appear in the sentence as the nodes of the graph. Then, we will connect related words as edges. The rule for connecting edges will be introduced in III-B2. As mentioned in III-A, the BiLSTM extract more contextual hidden states $H = \{h_1, ...h_i, ...h_m\}$ from the initial word vectors and we will get the final output. We consider the output of BiLSTM as the contextual representations of the words in the sentence. The sentence can be defined as $S = \{r_1, ...r_i, ...r_m\}$ So, for the graph of sentence S, $G = \langle N, E \rangle$, its node set N can be defined as:

$$N = \{r_i | i = 1, ..., m\} \tag{1}$$

Because we build graph for each sentence, compared with the previous methods of building a whole graph on the entire corpus, this could be adapted for inductive learning with batch propagation more intuitively. Because when there are new data, we just need to build a new graph for it without changing the overall arrangement.

2) Edges of graph: In order to connect the related nodes, we define three types of edges which includes the word adjacency edges, the self-loop edges and the dependency correlated edges. The word adjacency edges connect each word with its adjacent words and the self-loop edges start from and end with the same node. The reason of setting these two types of edges is that the words are definitely correlated to themselves and their neighbors. Setting these edges can construct a fully connected graph to aid the transmission of semantic information. Then, to effectively obtain semantic information from complex texts, we utilize the dependency parser to calculate the dependency correlation between two words. It will select the related words in terms of the parts of speech and syntax. With the dependency correlation, the meaningful words can be connected directly. In this step, we introduce the dependency parser algorithm to extract the syntax representations. In this algorithm, the highway BiLSTM takes as input pretrained word embeddings, frequent word and lemma embeddings, character-level word embeddings and summed morphological features embeddings. Then, the output of BiLSTM will be fed into the fully-connected layer. Finally, a biaffine classifier will make the prediction. Figure 2 shows a example of dependency parser of the sentence 'I like this movie because it is interesting':

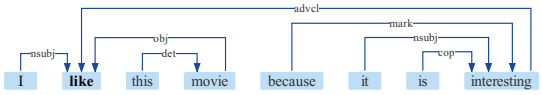

Fig. 2. This is the dependency tree of the example sentence of a movie review. The root of this tree is like.

Additionally, following the model proposed by Diego Marcheggiani and Ivan Titov [36], we add the self-loop for each node. We denote the case that there are dependency relationship between $word_i$ and $word_j$ as $D(i, j)$. Overall, for the graph $G = \langle N, E \rangle$ of sentence with *m* words, the edges set can be defined as:

$$E = \{e_{ij} | i \in [1, m]; j[i - p, i + p]\} \cup \{e_{ij} | D(i, j)\} \tag{2}$$

In Equation (2), e_{ij} denotes the edge between $word_i$ and *p* denotes the number of adjacent words connected to each word in the graph. When adding the edge between the words with dependency relationship, if the edge exists, we will skip it.

Figure 3 shows an example of graph for a single text "I like this movie because it is interesting". We can see that if we only utilize the traditional edge construction method, there will not be direct edges between the words ("like", "movie") and ("like", "interesting"). However, with the dependency parser, we can connect these words directly which will convey stronger semantic information. For the cases that there are more than one sentences, the subsentences will be connected

by the word adjacent edges and the dependency correlation edges.

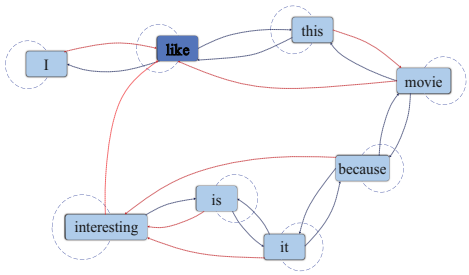

Fig. 3. This is the graph for the text "I like this movie because it is interesting". We set $p = 1$ for each nodes which means they establish edges with their nearest neighbor. These edges are denoted with dark blue solid lines. The highlighted node "like" is the root in the dependency tree. The red lines represents the dependency correlation between the nodes. The dash blue dotted lines notate the self-loops of nodes.

For the initialization of the edge weights, instead of setting the initial edge weights to be 1, we introduce the concept of pointwise mutual information (PMI) [37] to measure the correlation between two words to initialize the weights of the edges. PMI is the variant of mutual information (MI). PMI can be calculated by the Equation (3):

$$PMI(w_i, w_j) = \log \frac{P(w_i, w_j)}{P(w_i)P(w_j)} \tag{3}$$

w_i, wj denotes the word *i* and word *j*, $P(w_i), P(w_j)$ refers the probability of occurrence of word i and word j. And $P(w_i, w_j)$ is the probability of words i and j appearing in the same sentence. The PMI will be calculated in the training set. With the PMI to be the initial values, the edge weights will be updated during the process of the training.

C. Message Passing Mechanism

There are two ways for the implementation of convolution in graph: the spectral approaches [38] and the non-spectral approaches [39]. In order to enable each node to store global semantic information in its representation, we use a non-spectral method named message passing mechanism (MPM) for convolution to extract information from local features. With MPM, the node can obtain information from the adjacent nodes and with the collected information and its initial representations, and updates its representations. This process can be defined as:

$$m_v = \sum_{u \in P(v)} M(h_u, e_{vu}) \tag{4}$$

$$M(h_u, e_{vu}) = concat(h_u, e_{vu}) \tag{5}$$

$$h'_v = U(h_v, m_v) \tag{6}$$

$$U(h_v, m_v) = \sigma(h_v m_v) \tag{7}$$

In the Equation (4), m_v denotes the messages that the node receives from its neighbor nodes. M is the message function to select the message to be received. $P(v)$ refers to the p neighbors of the node v. h_v, h_u refers to the initial representation of node v, u. e_{vu} is the edge weight from node u to node v. In Equation (5), $concat$ refers to the concatenation of the h_u and e_{vu}. In the Equation (6),U is node update function which indicates how much information should be kept to update the h_v. h'_v is the updated representation of node v. In the Equation (7), σ denotes the sigmoid function. This is the basic concept the MPM. Based on this concept, we introduce the GCN aggregator notated as ρ that mentioned in section 2.1 to make this method more suitable for the graph-structured data:

$$Edges : m_v = \rho(h_u, e_{vu}) \tag{8}$$

$$\rho(h_u, e_{vu}) = \sum_{u \in P(v)} \mathcal{N}(M_{gcn}(h_u, e_{vu})) \tag{9}$$

$$M_{gcn}(h_u, e_{vu}) = h_v e_{vu} \tag{10}$$

$$Nodes : h'_v = U_{gcn}(m_v, h_v) \tag{11}$$

$$U_{gcn}(h_v, m_v) = \sigma(H^{deg(v)} m_v) \tag{12}$$

In the above equations, Equation (8) is the operation on the edges, and Equation (11) is the operation on the nodes. In Equation (8), ρ demotes the GCN aggregator. In Equation (9), $\mathcal{N}$ refers to the normalization operation and it will normalizes the messages obtaining from the neighbor nodes. In Equation (10), Message function M_{gcn} combines the representations of the node v, u and the weight of the edge between node u, v to generate the message. Then, the aggregation function ρ will aggregate the received messages. The GCN aggregator will return the sum of the normalized messages from the neighbor nodes. Finally, the node update Equation (12) U_{gcn} will combine the aggregated information and the initial information of h_v to update the representation of the node v. $deg(v)$ is the degree of node v and $H^{deg(v)}$ is a learned matrix for the update function which contain the information of h_v.

The key concept of the MPM is to update the node representation by collecting the information from its neighbor nodes. So, the node representation or in other words, the meaning of the word, is influenced by the context which is consistent with the reality. Even if this word is an irony or a polysemous word, we can still predict its meaning through contextual meaning. And the messages of the node representations are passed in the whole graph since the graph that we construct is a connected graph. Therefore, each node representation contains global information.

IV. Experiments

We perform experiments on the IMDB datasets, the TweetEval datasets [40] and the SUBJ datasets [41]. IMDB dataset is about the movie reviews and it has two sentiment polarities: positive and negative. TweetEval is the Emotion Recognition datasets [42] which represents the emotion of the tweets where we summarize the anger and sadness emotion to be negative, the joy and optimism emotion to be positive. The SUBJ

TABLE I
STATISTICS OF EXPERIMENT RESULTS

Models	IMDB		TweetEval		SUBJ	
	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1
CNN	84.682%	84.681%	72.695%	67.123%	89.300%	89.300%
LSTM	88.881%	88.881%	79.546%	76.512%	90.121%	90.115%
Text-GCN	89.490%	89.487%	79.873%	76.402%	90.900%	90.899%
Text-GNN	87.754%	87.751%	77.344%	74.029%	89.214%	89.210%
CfC	88.990%	88.986%	66.854%	42.206%	83.700%	83.693%
UnICORNN	88.935%	88.935%	69.388%	50.344%	-	-
LSTM-GCN	**89.923%**	**89.917%**	**80.611%**	**77.976%**	**91.129%**	**91.129%**

dataset consists of hand-annotated documents drawn from the webpages retrieved by Yahoo! search engine. The annotations indicate whether the documents are subjective or objective. The statistic summaries of the datasets are shown in Table II:

TABLE II
THE STATISTICS OF DATASETS

Datasets	Train	Valid	Test	Cat.
IMDB	36000	4000	10000	2
TweetEval	3257	374	1421	2
SUBJ	8000	1000	1000	2

A. Implementation Details

For all the experiments of our model, we use the pretrained 300-dimensional GloVe vectors [32] to initialize the word embeddings. p mentioned in III-B2 is set to be 3 which means the nodes will build edges with their 6 nearest neighbor nodes. To alleviate overfitting, we apply dropout at a rate of 0.5 to the input word embeddings of the BiLSTM. The number of the GCN layer is set to be 3 and the dropout rate of the GCN is also 0.5. When constructing the graph, we choose to introduce the Stanfordnlp as the dependency parser. We use the Adam optimizer [43] with the initial learning rate of 0.01. Finally, we apply the early stop method if the validation loss does not decrease after 15 epochs.

For the baseline models, we use their default structures and parameters as in their original papers or implementations. We also used 300-dimensional GloVe word embeddings for baseline models if needed.

B. Baseline Models

We compared our model with five model and the brief descriptions are as follows:

- CNN [44], classifies texts with convolution and max pooling method to get text representation.
- LSTM [10], utilizes the BiLSTM for text classification whose last hidden state is the text representation.
- Text GCN [19], performs the graph convolutional operation on a single text graph for a corpus in text classification task.
- Text GNN [18], performs the GNN techniques on text classification in text level.
- CfC [45], proposes a Closed-form Continuous-depth networks which is fast for the time-series prediction tasks.
- UnICORNN [46], design a RNN for learning very long time dependencies.

C. Comparison Results

To evaluate the sentiment analysis models, we use the accuracy and macro-averaged F1 score as the main evaluation metrics. The main experiment results are in Table I. According to Table I, we can see that our LSTM-GCN model outperforms other models which means that our model can effectively extract and integrate the representations of the texts. Besides, the LSTM-GCN model performs better with the complicated reviews. Compared with the method that cannot obtain the contextual information like CNN, our model can extract the information of context because of the GCN structure. And message passing mechanism can deliver more useful information to update the text representations than the time series model such as LSTM, CfC and UnICORNN. Moreover, although the traditional GCN edge building method can make each word contain a certain degree of global information, its performance is still not good enough for comments that are too long or too complex. The introduction of dependency parser enables our model to directly connect related words even if they have multiple nodes between them which make our LSTM-GCN model performs better on the long reviews.

D. Ablation Study

In this section, we conduct some ablation studies to further investigate the role of each module in our model. The result is shown in Table III.

The BiLSTM model uses only the word embeddings and the BiLSTM layer to do the sentiment task. Similarly, the GCN model applies the word embeddings and GCN layers on the task, besides, it also utilizes the PMI and dependency parser. And we can see that the performance of the GCN model is better than that of the BiLSTM model which means that the GCN model can extract more useful and global information than the BiLSTM model. LSTM-GCN w/o PMI indicates that we remove the PMI so that the trainable edges of the graph will be trained with 1 as initial weight. Therefore, the performance degrades on the IMDB dataset. LSTM-GCN w/o DP means we remove the dependency parser which results in that each

TABLE III
EXPERIMENT RESULTS OF ABLATION STUDY

Models	IMDB		TweetEval		SUBJ	
	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1
BiLSTM	88.954%	88.947%	79.617%	76.580%	90.222%	90.217%
GCN	89.123%	89.116%	77.131%	74.037%	89.819%	89.812%
LSTM-GCN w/o PMI	89.862%	89.856%	79.119%	75.994%	89.718%	89.718%
LSTM-GCN w/o DP	89.474%	89.468%	77.699%	75.008%	88.811%	88.806%
LSTM-GCN	**89.923%**	**89.917%**	**80.611%**	**77.976%**	**91.129%**	**91.129%**

TABLE IV
EXPERIMENT RESULTS OF IMPACT OF THE AGGREGATOR TYPES

Aggregators	IMDB		TweetsEval		SUBJ	
	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1
POOL	50.690%	36.272%	66.122%	39.803%	90.327%	90.321%
MEAN	88.772%	88.765%	78.338%	75.227%	90.323%	90.322%
LSTM	50.424%	50.415%	76.421%	74.259%	89.817%	89.817%
GCN	**89.923%**	**89.917%**	**80.611%**	**77.976%**	**91.129%**	**91.129%**

node in the graph can only establish edges with its three neighbor nodes. This experiment indicates that the dependency parser increases the propagation quality of global information in graphs. Overall, every module of our LSTM-GCN model is necessary and with them all, our model performs effectively.

E. Impact of the aggregator types in GCN

In this section, we investigate the impact of the types of aggregators in GCN. We separately conduct the experiments with GCN, LSTM, MEAN, POOL aggregator separately on the IMDB dataset. The result is shown in Table IV.

According to it, we can see that the GCN aggregator performs the best in all types of aggregators. In contrast, we can see that LSTM falls short in unordered graphs. Compared the elementwise mean method of Mean aggregator, the Pool aggregator performs max pooling operation which might introduce more noise. Besides, the Pool and LSTM aggregators are not suitable for the long texts.

F. Impact of the Hyper Parameter p

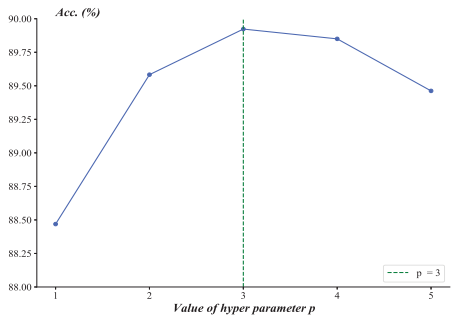

Fig. 4. This figure shows the change in accuracy for different values of the hyper parameter p on IMDB dataset

In this section, we will explore the impact of the hyper parameter p. The variable p denotes that node n will build edges with its p neighbor nodes. We perform the experiments on the IMDB dataset and the result is shown in Figure 4. And we can see that when the variable p is set to be 3, the model achieves the best performance. On one hand, when the value of p is too small, the nodes cannot obtain enough information from its neighbor nodes to update it representation, on the other hand, there will be much noise from the irrelevant node in the message passing process if the value of p is too large and it will result in a huge amount of computation.

V. CONCLUSION

In this paper, we propose the LSTM-GCN architecture to solve the disadvantages of the traditional GCN networks for sentiment analysis task. Our LSTM-GCN model captures the sentiment representations from context-based and syntax-based views to address the drawbacks of traditional graph construction methods. LSTM-GCN can contextualize the pre-trained word vectors and extract the semantics information due to the introduction of the BiLSTM and dependency parser. Furthermore, the utilization of the message passing mechanism (MPM) and the pointwise mutual information (PMI) method makes our model transmit semantic information efficiently. Extensive experiments show that our LSTM-GCN model outperforms baselines, which means that the introduction of multiview representation can increase the accuracy of sentiment analysis.

VI. ACKNOWLEDGEMENT

This work was supported in part by the National Key Research and Development Program of China (Grant No. 2019YFA0706200), in part by the Natural Science Foundation of Gansu Province,China(Grant No. 22JR5RA488), in part by the National Natural Science Foundation of China (Grant No.61632014, No.61627808).

REFERENCES

[1] B. Liu, "Sentiment analysis and opinion mining," *Synthesis lectures on human language technologies*, vol. 5, no. 1, pp. 1–167, 2012.

[2] J. Li and E. Hovy, "Reflections on sentiment/opinion analysis," in *A practical guide to sentiment analysis*. Springer, 2017, pp. 41–59.

[3] I. Titov and R. McDonald, "Modeling online reviews with multi-grain topic models," in *Proceedings of the 17th international conference on World Wide Web*, 2008, pp. 111–120.

[4] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao, "Target-dependent twitter sentiment classification," in *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, 2011, pp. 151–160.

[5] S. Kiritchenko, X. Zhu, C. Cherry, and S. Mohammad, "Nrc-canada-2014: Detecting aspects and sentiment in customer reviews," in *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, 2014, pp. 437–442.

[6] L. H. Lee, D. Isa, W. O. Choo, and W. Y. Chue, "High relevance keyword extraction facility for bayesian text classification on different domains of varying characteristic," *Expert Systems with Applications*, vol. 39, no. 1, pp. 1147–1155, 2012.

[7] H. Kang, S. J. Yoo, and D. Han, "Senti-lexicon and improved naïve bayes algorithms for sentiment analysis of restaurant reviews," *Expert Systems with Applications*, vol. 39, no. 5, pp. 6000–6010, 2012.

[8] V. N. Phu, V. T. N. Tran, V. T. N. Chau, N. D. Dat, and K. L. D. Duy, "A decision tree using id3 algorithm for english semantic analysis," *International Journal of Speech Technology*, vol. 20, no. 3, pp. 593–613, 2017.

[9] R. Bib, U. Qamar, M. Ansar, and A. Shaheen, "Sentiment analysis for urdu news tweets using decision tree," in *17th IEEE/ACIS International Conference on Software Engineering Research, Management and Applications (SERA 2019)*, 2019.

[10] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," *arXiv preprint arXiv:1605.05101*, 2016.

[11] O. Araque, I. Corcuera-Platas, J. F. Sánchez-Rada, and C. A. Iglesias, "Enhancing deep learning sentiment analysis with ensemble techniques in social applications," *Expert Systems with Applications*, vol. 77, pp. 236–246, 2017.

[12] L. Jingsheng and J. Ting, "Hierarchical text classification based on bp neural network," *Journal of Computational Information Systems*, vol. 5, pp. 581–590, 2009.

[13] Y. Chen, "Convolutional neural network for sentence classification," Master's thesis, University of Waterloo, 2015.

[14] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015.

[15] L. Mou, G. Li, L. Zhang, T. Wang, and Z. Jin, "Convolutional neural networks over tree structures for programming language processing," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[16] J. Liao, S. Wang, and D. Li, "Identification of fact-implied implicit sentiment based on multi-level semantic fused representation," *Knowledge-Based Systems*, vol. 165, pp. 197–207, 2019.

[17] E. Zuo, H. Zhao, B. Chen, and Q. Chen, "Context-specific heterogeneous graph convolutional network for implicit sentiment analysis," *IEEE Access*, vol. 8, pp. 37 967–37 975, 2020.

[18] L. Huang, D. Ma, S. Li, X. Zhang, and H. Wang, "Text level graph neural network for text classification," *arXiv preprint arXiv:1910.02356*, 2019.

[19] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 7370–7377.

[20] M. Ghiassi, J. Skinner, and D. Zimbra, "Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial neural network," *Expert Systems with applications*, vol. 40, no. 16, pp. 6266–6282, 2013.

[21] A. Tripathy, A. Agrawal, and S. K. Rath, "Classification of sentiment reviews using n-gram machine learning approach," *Expert Systems with Applications*, vol. 57, pp. 117–126, 2016.

[22] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*. PMLR, 2017, pp. 1263–1272.

[23] P. Melville, W. Gryc, and R. D. Lawrence, "Sentiment analysis of blogs by combining lexical knowledge with text classification," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 1275–1284.

[24] T. Chen, R. Xu, Y. He, and X. Wang, "Improving sentiment analysis via sentence type classification using bilstm-crf and cnn," *Expert Systems with Applications*, vol. 72, pp. 221–230, 2017.

[25] O. Appel, F. Chiclana, J. Carter, and H. Fujita, "A hybrid approach to the sentiment analysis problem at the sentence level," *Knowledge-Based Systems*, vol. 108, pp. 110–124, 2016.

[26] T. T. Thet, J.-C. Na, and C. S. Khoo, "Aspect-based sentiment analysis of movie reviews on discussion boards," *Journal of information science*, vol. 36, no. 6, pp. 823–848, 2010.

[27] W. Xue and T. Li, "Aspect based sentiment analysis with gated convolutional networks," *arXiv preprint arXiv:1805.07043*, 2018.

[28] M. Yang, Y. Ma, Z. Liu, H. Cai, X. Hu, and B. Hu, "Undisturbed mental state assessment in the 5g era: a case study of depression detection based on facial expressions," *IEEE Wireless Communications*, vol. 28, no. 3, pp. 46–53, 2021.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[30] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[31] D. Zeman, M. Popel, M. Straka, J. Hajic, J. Nivre, F. Ginter, J. Luotolahti, S. Pyysalo, S. Petrov, M. Potthast *et al.*, "Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies," in *CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, 2017, pp. 1–19.

[32] Y. Lai, L. Zhang, D. Han, R. Zhou, and G. Wang, "Fine-grained emotion classification of chinese microblogs based on graph convolution networks," *World Wide Web*, vol. 23, no. 5, pp. 2771–2787, 2020.

[33] K. Sun, R. Zhang, S. Mensah, Y. Mao, and X. Liu, "Aspect-level sentiment analysis via convolution over dependency tree," in *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, pp. 5679–5688.

[34] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[35] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[36] D. Marcheggiani and I. Titov, "Encoding sentences with graph convolutional networks for semantic role labeling," *arXiv preprint arXiv:1703.04826*, 2017.

[37] K. Church and P. Hanks, "Word association norms, mutual information, and lexicography," *Computational linguistics*, vol. 16, no. 1, pp. 22–29, 1990.

[38] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[39] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," *Advances in neural information processing systems*, vol. 28, 2015.

[40] F. Barbieri, J. Camacho-Collados, L. Espinosa-Anke, and L. Neves, "TweetEval:Unified Benchmark and Comparative Evaluation for Tweet Classification," in *Proceedings of Findings of EMNLP*, 2020.

[41] B. Pang and L. Lee, "Using very simple statistics for review search: An exploration," in *Proceedings of COLING: Companion volume: Posters*, 2008, pp. 73–76.

[42] S. Mohammad, F. Bravo-Marquez, M. Salameh, and S. Kiritchenko, "Semeval-2018 task 1: Affect in tweets," in *Proceedings of the 12th international workshop on semantic evaluation*, 2018, pp. 1–17.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[44] Y. Kim, "Convolutional neural networks for sentence classification," *Eprint Arxiv*, 2014.

[45] R. Hasani, M. Lechner, A. Amini, L. Liebenwein, M. Tschaikowski, G. Teschl, and D. Rus, "Closed-form continuous-depth models," *arXiv preprint arXiv:2106.13898*, 2021.

[46] T. K. Rusch and S. Mishra, "Unicornn: A recurrent model for learning very long time dependencies," in *International Conference on Machine Learning*. PMLR, 2021, pp. 9168–9178.