Research Article

A Graph Convolutional Network-Based Sensitive Information Detection Algorithm

Ying Liu,1 Chao-Yu Yang,1 and Jie Yang2

1School of Economics and Management, Anhui University of Science and Technology, Huainan, China
2Faculty of Engineering and Information Sciences, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW, Australia

Correspondence should be addressed to Chao-Yu Yang; yangchy@aust.edu.cn

Received 16 November 2020; Revised 26 January 2021; Accepted 14 March 2021; Published 24 March 2021

Academic Editor: M. Irfan Uddin

Copyright ©2021 Ying Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In the field of natural language processing (NLP), the task of sensitive information detection refers to the procedure of identifying sensitive words for given documents. The majority of existing detection methods are based on the sensitive-word tree, which is usually constructed via the common prefixes of different sensitive words from the given corpus. Yet, these traditional methods suffer from a couple of drawbacks, such as poor generalization and low efficiency. For improvement purposes, this paper proposes a novel self-attention-based detection algorithm using the implementation of graph convolutional network (GCN). The main contribution is twofold. Firstly, we consider a weighted GCN to better encode word pairs from the given documents and corpus. Secondly, a simple, yet effective, attention mechanism is introduced to further integrate the interaction among candidate words and corpus. Experimental results from the benchmarking dataset of THUC news demonstrate a promising detection performance, compared to existing work.

1. Introduction

Due to the explosive growth of information from the Internet, the content monitoring and modelling for user online behavior are essential [1, 2]. On the one hand, in the process of online review dissemination, web service providers need to identify abusive (such as aggregative, violent, sexual, and cyber-threatening) contents. In addition, the email systems also need to filter inappropriate contents like spams. On the other hand, the industry stakeholders and authorities are interested in user online behavior for better marketing and development purposes. For instance, smart app developers can gain a better understanding of users’ profile via their mobile usage [3]. The content providers can identify how opinion leaders influence their followers via online posts [4].

As such, the task of content monitoring and modelling is of great importance, which plays a significant role in natural language processing (NLP).

The sensitive word detection is a particular problem for content monitoring, which refers to the procedure of identifying target words from the given documents. The majority of existing detection algorithms are based on the concept of sensitive word tree (SMT) [5–7]. As a tree structure, the SMT is a variant of the hash tree. Each node from SMT (except the root node) contains one character, while different nodes from this SMT could contain the same character. Accordingly, a sensitive word can be represented as a series of nodes from the root to a certain leaf node. Once constructed, sensitive words are identified by detecting or checking candidate characters layer-by-layer. The SMT structure makes full use of the common prefix between sensitive words, thereby reducing query time through the common prefix of sensitive words and minimizing unnecessary character comparisons. Given its simplicity, the SMT plays an essential role in detecting sensitive words for given documents and finds its wide applications in many scenarios. Section 2.1 provides a more detailed discussion on existing sensitive words detection methods.

There are, however, still considerable research questions about the efficiency and generalization of word detection, as
the traditional SMT-based methods have several disadvantages. Firstly, the employed corpus is critical to the success of SMT-based detection. That is, using a comprehensive corpus (with a large number of sensitive words) could result in a model with a huge computational overhead, while applying a small corpus may not be sufficient to detect due to lack of information. Secondly, the SMT-based detection tends to identify explicit contents according to the corpus, with the potential risk of missing out-of-corpus words. Thirdly, even with the comprehensive corpus, the resultant SMT will have a large depth and width, due to the high character repetition rate among the sensitive words. As a result, there is an increasing need to develop alternative SMT-similar algorithms for the sensitive words detection.

In this paper, we propose an improved detection algorithm based on the self-attention mechanism and graph convolutional network. More precisely, the proposed algorithm follows an end-to-end training model, which consists of three layers. In the encoding layer, the word embedding is extracted using pretrained models, including words from the given documents and corpus; in the integration layer, we consider applying the graph convolutional network to jointly model implicit features among all word pairs of text (from the documents and corpus), while the self-attention mechanism is used to further refine extracted implicit features. At last, in the output layer, the probability of a word being the target is calculated via a softmax (similarity-based) operation. A notable advantage of the proposed method, compared to existing detection methods, is that our method is capable of identifying implicit sensitive words (out-of-corpus), due to the proposed similarity-based detection. As such, the main contribution of this paper is twofold. Firstly, we consider a weighted graph convolutional network to encode words. Secondly, a simple, yet effective, self-attention mechanism is further employed to understand the interaction between candidate words and corpus. We evaluate the method on the public THUC news dataset. The experimental result shows that the proposed algorithm substantially improves the detection performance over existing methods.

The remainder of the paper is organized as follows. Section 2 provides the related background, including an overview of the existing tree-based detection methods, and the self-attention mechanism and graph convolutional network. Section 3 proposes a novel detection algorithm, which consists of three layers, i.e., encoding, integration, and output, respectively. Section 4 first investigates the robustness of the proposed algorithm and then compares it with state-of-the-art feature selection algorithms on a collection of real-world detection applications. Section 5 presents concluding remarks and future works.

2. Related Work

This section offers some background information about the proposed work. At first, we discuss the existing tree-based detection methods. We then review the self-attention mechanism and graph convolutional network, which are the main focus of this paper.

2.1. Sensitive Words Detection. The task of sensitive words detection has attracted a lot of attention, due to the prevalence of online users’ generated content (UGC). The majority of detection algorithms are based on the concept of sensitive word tree (SMT), which represents one sensitive word by a node path from the root to a certain leaf node [5–7]. Note that common prefix characters from different sensitive words will usually occupy same nodes in the sensitive word tree.

During the SMT construction, given any candidate words, the word detection procedure starts from the root node and looks up the child nodes layer by layer. If the first character of the sensitive word is found in one of the child nodes, this node is regarded as a candidate character node; otherwise, a new node is created next the root node to store the character. As such, the word detection procedure continues to compare whether the character exists in the child node of the previous candidate character node. If found, use the subsequent child node as the current candidate character node; otherwise, create a new node under the previous character node to store the character. The procedure will keep on searching the SMT, until all characters from the sensitive words are compared. Figure 1 illustrates an example of SMT, from which each character forms a tree node.

Overall, the majority existing algorithms follow the procedure of tree construction and single-pass detection. That is, the occurrences of finite keywords from the corpus are employed to form a sensitive-word tree, and then candidate words from the given document are verified in a single-pass based on this tree [8]. For using the SMT to detect sensitive content, the most-common technique is the Deterministic Finite Automaton (DFA). This automaton method, which enables state transfer, given an automaton state and an input symbol, is applied widely for the purpose of transferring one state to the next automaton state. Let \( D \) be a quintet, \( K \) the set of nonempty states, \( \Sigma \) the set of nonempty input symbols, and \( M \) the transfer function (i.e., a mapping function to have \( K \times \Sigma \rightarrow K \)). Furthermore, assume that \( S \) represents the initial state, and \( F \) is the set of nonempty termination states. The DFA algorithm can be represented as \( D = (K, \Sigma, M, S, F) \). As such, given that the current state is \( k_i \) and the input symbol is \( a \), the state \( k_j \) is obtained through the conversion function \( M \), and we have \( M(k_i, a) = k_j, (k_i, k_j \in K) \).

There are some DFA-based applications to build the SMT for word detection. For instance, the work from [9] proposes a compressed representation for DFA, termed Delta finite automata. This method utilizes the Nth-order relationship in DFA, which helps in reducing number of states and transitions, and retaining the conversion of each character. Experimental results show that the Delta finite automata model achieves a higher search speed compared to traditional DFA. An improved algorithm based on deterministic finite automata (DFA) is proposed in [10], termed swift tree DFA (ST-DFA). This algorithm only needs to establish the sensitive word tree once via multiple filtering of words information. The target corpus is constructed by 3000 sensitive words from the Internet. Intensive experiments using 800 Tieba pages are employed, while the results
indicate a high detection efficiency. Furthermore, an improved work based on ST-DFA is presented in [11], from which a multistage depth filtering strategy is introduced. In particular, some spatial information, in addition to the sensitive words, is employed to build the sensitive corpus. The experimental results using data from Crustal Movement Observation Network of China show that the method leads to high accuracy of identifying sensitive information, as well as fast detection process.

Despite the wide application of tree-based detection, there are still some research problems. For instance, a corpus could suffer from the repetition of sensitive keywords. The repeated keywords lead to a huge SMT with large number of levels and high depth. Not only will it consume a large number of storage space, but also it is extremely time-consuming to build and apply the tree for word detection.

2.2. Self-Attention Mechanism and Graph Convolutional Network. In this section, we will focus on the graph convolutional network (GCN) and self-attention mechanism, on which our method is built. GCN is one particular end-to-end learning framework operating on graphs. The main concept of GCN is to iteratively aggregate feature information from the neighborhood of a node, which is extremely effective for node classification tasks. Some of the existing GCN-based works are summarized in Table 1.

In [12], the author proposes a graph convolutional network with fusion of semantic and structure (termed FSS-GCN), which is applied to the field of sentiment analysis. FSS-GCN not only captures the semantic relevance, but also considers the dependency relationship between clauses in the document. Public datasets from different languages (Chinese and English) are selected for experimental purposes. By continuously injecting structural constraints into the GCN, the model is trained to learn the global and local structure, enabling the capability of selectively dealing with relevant terms for sentiment analysis of emotional causes. In addition, a combination of GCN and MapReduce model is introduced in [13] to build a recommendation system. In particular, the GCN is proposed to combine random walk and graph convolution operation to generate node embeddings. Empirically, the recommendation system is established using the Pinterest object graph, while the results are shown with four orders of magnitude accuracy better than the typical GCN implementation, and with better recommendation than other systems based on deep learning.

GCN has also found its applications in the field of named entities and relationships. For instance, in [15], the GCN is applied to solving the problem of semantic role labeling (SRL). More precisely, the syntactic dependency graph and the syntactic information are utilized. Later, this model is trained on the CoNLL-2009 dataset (using English and Chinese). The experiment demonstrates that bidirectional LSTM and syntax-based GCN have shown complementary modelling capabilities. An improved GCN is introduced in [16] for detecting controversial posts, termed Topic-Post-Comment Graph Convolutional Network (TPC-GCN). In particular, this method integrates information from graph structure with textual topic, post and comment content. Furthermore, an extension of TPC-GCN is considered to handle posts with different topics from the training set. In this paper, the Chinese Weibo dataset and the English Reddit dataset are selected for model training. Experiments have found that the model can effectively identify controversial posts and is superior to existing methods in terms of performance and generalization.

Self-attention, on the other hand, is an attention mechanism applied on different positions of a single sequence, in order to compute a representation of the sequence. The main purpose is to calculate the interactive significance of some particular positions. Assuming that the input is Q (Query), the context is stored in the form of key-value pairs (K, V) in Memory. Then, the attention mechanism can be formulated as a mapping function from Query to a series of key-value pairs (Key, Value) as follows:

$$\text{attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{n}}\right)V. \quad (1)$$

The self-attention mechanism has been proven to be extremely effective in many fields such as machine reading, text summary, or image description generation. In [17], the self-attention method is introduced to extract interpretable sentence embeddings. By using a 2D matrix to represent sentence embedding, this model reveals the semantic importance of words and phrases. Later, this model is tested on 3 different tasks: author profile, sentiment classification, and textual entailment. Experimental results show that this model is significantly better than other sentence embedding models, in particular with a long sentence input. The work from [18] proposes a neural network model with internal attention for addressing the problem of phrases summarization. In particular, the cross-entropy loss function is introduced for the optimization purpose. Later, this model is tested on the CNN/Daily Mail and New York Times datasets. The experimental results show that the attention-based...
model improves the readability of the generated summary and is more suitable for long output sequences. At last, an extension of self-attention is proposed in [19] to incorporate the relative position representation into the self-attention mechanism, which can be applied to a variety of graph-labeled inputs. The experiments on the WMT 2014 machine translation task have proved that the effectiveness of this method, which produces significantly improved translation results.

3. Proposed Sensitive Word Detection Algorithm

In this section, we describe the proposed detection algorithm for filtering sensitive words. Herein, our method follows an end-to-end training procedure, which consists of three main modules (or layers): encoding, integration, and output. In particular, the graph convolutional network is employed to encode words from given documents and the corpus, while a self-attention mechanism is also considered to refine the extracted features to understand the hidden interaction.

The overall idea of the proposed graph convolutional network and self-attention-based algorithm (GCSA) is described as follows: (1) obtain the word embedding from the given documents and corpus; (2) apply the graph convolutional network and self-attention mechanism to extract structural information and relationship (features) from keywords; (3) predict the probability of candidate keywords and output the final results. The pipeline of the proposed GCSA algorithm is also shown in Figure 2.

3.1. Encoding Layer. This layer is the first module of our proposed GCSA algorithm, which is used to encode the textual information from the given documents and corpus. In general, it takes as input documents \( D \) and the corpus \( C \) and computes each token from them with a context-aware representation. Some pretrained models can be applied, such as BERT and GloVe.

\[ h_i = \text{BEST}(\text{word}_i), \quad i = [1, N_S]. \]  

As such, the entire sentence can be represented as \( H_S = \{h_1, h_2, \ldots, h_{N_S}\} \). Similarly, we also apply the same encoding layer to words from the corpus and obtain \( H_C = \{d_1, d_2, \ldots, d_{N_C}\} \), where \( |C| \) is the size of the corpus \( C \).

3.2. Integration Layer. This layer is designed to integrate the word embedding from the encoding layer with their structural information from their neighborhood words. As the core module, the integration layer using the graph-attention (or the combination of graph convolutional network and self-attention mechanism) technique is able to capture the interaction of word pairs from the given documents and corpus.

To do so, we employ the dependency parser to create a dependency tree for the input sentence, which provides a representation of grammatical relationships between words in a sentence. Based on the dependency analysis, we are then able to model both revealed syntactical structures or hidden relationships among particular words. As an example, a dependency tree from the input sentence “玛格丽特 福特是位精神分析师，为帮助病人比利，她实地深入赌场了解赌徒的心理状况。（翻译：Margaret Ford is a specialist in psychic analysis. For helping the patient Billy, she purposely went to the casino to investigate the psychological condition of the gambler）” is shown in Figure 3.

Based on the dependency tree, the proposed GCN is designed to form an undirected graph for any given sentences. The refined word embedding is then formulated as follows:

\[ h_u^{t+1} = \text{ReLU} \left( \sum_{v \in N(u)} (W^h h_v + b^h) \right), \]  

where \( u \) is the target node, \( N(u) \) represents the set of neighborhoods of \( u \) (from the dependency tree), \( h_v \) denotes the word embedding for the node \( v \), and \( W \) and \( b \) are learnable weights. At last, we apply the combination of the original word embedding and GCN embedding as the final feature: \( k_i = [h_i; v_i] \), which represents the vector concatenation operation.

Next, we will need to formulate the similarity calculation for candidate words and the corpus, which is done by a self-attentive mechanism in our paper:

\[ \alpha_{ij} = \exp(d_j W h_i), \]

where \( h_i \) is the final feature from the i-th word, and \( W \) is a trainable vector. As observed, \( \alpha_{ij} \) can be considered as a similarity measurement between \( h_i \) and \( d_j \) (or words from the given documents and the corpus). On the other hand, the attention weights \( \alpha_{ij} \) will be larger if sentences are more relevant to words from the corpus. Again, the attended

| Table 1: Representative works on GCN. |
|---------------------------------------|
| References | Method | Application |
| [12] Bi-LSTMs + attention guided GCNs + self-attention mechanism | Emotion cause analysis |
| [13] GCN + MapReduce | Web-scale recommender systems |
| [14] Bi-LSTM + Bi-GCN + GCN | Entity and relation extraction |
| [15] GCN + LSTM | Semantic role labeling |
| [16] GCN + attention mechanism + two-branch multitask architecture | Controversy detection |

Complexity
Figure 2: (a) The pipeline of the proposed GCSA algorithm, which consists of three main modules, including (b) applying the pretrained model to obtain the word embedding, (c) extracting dependency relationship and employing graph-attention network to estimate the similarity between candidate words and corpus, and (d) performing the subsequent classification.

Figure 3: An example of dependency tree for an input sentence. Conventional abbreviations include the following: SBV = subject of verb; VOB = object of verb; IOB = indirect object; FOB = fronting object; DBL = double roles: subject and object; ATT = attribute; ADV = adverbial; CMP = complement; COO = coordinate; POB = preposition-object; LAD = left adjunct; RAD = right adjunct. More details about the dependency types can be found in http://ltp.readthedocs.io/zh_CN/latest/appendix.html#id5.
vector \( r \) also can be thought as a self-attentive result to show the interaction between sentence and sensitive words.

### 3.3. Output Layer

In this layer, we adopt a linear output layer to calculate the probability of each token from a given sentence to be the target words. That is, the prediction score is normalized using softmax to get the final probability distribution:

\[
P = \frac{\exp(w^T \hat{o})}{\sum_j \exp(w^T o_j)}
\]

where \( o \) is the output by the integration layer, and \( w \) is the trainable parameter. The training objective is the log-likelihood as follows:

\[
L = -\frac{1}{N} \sum_{j=1}^{N} (\log p),
\]

where \( N \) is the number of keywords from the input document. At inference time, the word with maximum \( p \) is chosen as the sensitive word.

### 3.4. Summary

In this section, the proposed GCSA algorithm for identifying sensitive words using the concept of graph convolutional network and self-attention mechanism is shown in Algorithm 1. During the data preprocessing stage, the original textual data are processed to remove junk characters, abbreviations, and emotional icons, etc. After preprocessing all the candidate documents, the GCSA model is applied to encode them using the pretrain models (such as BERT); the proposed model further encodes each word with their dependency information before applying the self-attention strategy to compute the similarity between candidate words and the target corpus.

### 4. Experiment and Result Analysis

To investigate the efficiency of the proposed GCSA algorithm for sensitive word detection, we employ a real-world benchmark problem. The datasets and the evaluation criterion are presented in Section 4.1. The comparison of the proposed framework with conventional work is discussed in Section 4.2.

#### 4.1. Experimental Setup

To evaluate the proposed algorithm, the THUC-News dataset has been employed. This data resource consists of 740,000 news, from more than 14 classes, like social, sport, tourism, etc. In this paper, we selected news (or documents) with the maximal length of 1000 characters. As such, our primary aim is to identify sensitive words from given documents. Meanwhile, to remove noise and cleanse the raw data, the preprocessing procedure is firstly employed. On one hand, there is a large amount of junk characters needed to be removed, including (1) isolated @ signs; (2) user names; (3) URL links; (4) punctuation (such as “and,” “!,” “,” “,” “#,” and “$”); (5) stop words, such as “的 (of),” “在 (in),” “噢 (oh),” and “呀 (yah).”

Additionally, other preprocessing steps are also considered, such as word stemming and converting all letters to lowercase.

In addition, the reference corpus is also detailed. There are 4038 words within the employed corpus, which consists of six types of sensitive words, including Violent, Social, Sexual, Reactionary, and Corruption. Some example words from the corpus are “混蛋玩意儿 (bastard),” “神精病 (neuropathy),” “二货 (dummy),” “傻逼 (stupid),” “傻逼 (stupid),” “乌龟王八蛋 (son of a bitch),” “滚蛋 (fuck off),” “憨批 (stupid batch),” “傻子 (dumbass),” and “混球 (ass-hole).” The overall statistic of the employed corpus is shown in Table 2.

Furthermore, the training parameters for the proposed algorithm are shown in Table 3.

At last, the entire dataset is randomly partitioned into two independent sets: a training and testing set. The size of the training and testing sets is set as 80% and 20%, respectively. In terms of the result measurement, we use the following metric to evaluate the performance of word detection:

\[
P_{\text{accuracy}} = \frac{TP + TN}{TP + TN + FP + FN}
\]

where TP is the positive sample predicted by the model to be positive, FP is the negative sample predicted to be positive by the model, TN is the negative sample predicted to be negative by the model, and FN is the positive sample predicted to be negative by the model.

#### 4.2. Performance Analysis

In this section, the proposed algorithm is compared with three conventional algorithms, namely, DFA, FSS-GCN, and TPC-GCN. Note that we have briefly introduced those algorithms in Section 2, and their hyperparameters follow the original work.

Table 4 presents the average training and test accuracy obtained from different methods. Compared with conventional methods, the proposed algorithm achieves a significant improvement in terms of detection accuracy. For instance, DFA only achieves approximately 50% accuracy on the testing sets. One main reason could be that this model adopts explicitly the corpus to match the sentence; as such, DFA fails to detect sensitive words if they are missing from the corpus. On the other hand, three other methods (that is, GCSA, FSS-GCN, and TPC-GCN) are adopted to estimate the possibility of candidate words as the sensitive ones, and the detection performance is much higher. In addition, the proposed algorithm outperforms its peers in terms of both training and testing sets. One particular reason could be the employed dependency tree as an additional feature, which plays a significant role in determining the latent relationship between candidate words and target ones.

Next, the experiment compares different sizes of corpus, and the main purpose is to verify the robustness of the algorithm if the corpus is incomplete. The results of the experimental tests are shown in Table 5.

As shown in Table 5, the proposed algorithm is much robust than its peers; for instance, the average accuracy (on
In conclusion, it can be empirically confirmed that the proposed GCSA algorithm achieves comparative detection performance, compared with the existing methods. Again, GCSA employs the graph convolutional network to extract latent structural and textual information, while the similarity between the candidate words and corpus is further estimated using a self-attention mechanism. Experimental results show that the proposed method leads to a more accurate detection performance, even with the incomplete corpus.

5. Conclusion

In this paper, we propose a novel word detection algorithm to identify sensitive words from the given documents. Traditional methods reply on building a sensitive word tree and measure character-level similarity between the corpus and candidate content, which suffer from inaccurate detection and big computational cost. To address this issue, we propose a graph-attention-based detection network, which follows the end-to-end training model. More precisely, we first apply pretrained model to encode keywords from the given document and corpus. Then, we apply graph-attention network to extract structural and textual information, which is later used to form the classification. Experiments are conducted by using one real-world News dataset. The detection results clearly indicate a better detection performance from the proposed method, compared with other existing methods.

Data Availability

The experimental data in the paper can be found in https://github.com/18855482286/P1.git.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. A. Kelly, M. Ghafurian, R. L. West, and D. Reitter, “Indirect associations in learning semantic and syntactic lexical relationships,” Journal of Memory and Language, vol. 115, Article ID 104153, 2020.
[2] W. Sihang, W. Jiapeng, M. Weihong, and J. Lianwen, “Precise detection of Chinese characters in historical documents with deep reinforcement learning,” Pattern Recognition, vol. 107, Article ID 107503, 2020.
[3] Y. Jie, J. Ma, and S. K. Howard, “Usage profiling from mobile applications: a case study of online activity for australian primary schools,” Knowledge-Based Systems, vol. 191, Article ID 105214, 2019.
[4] Y. Jie, Y. Brian, and Z. Peter, “Characteristics of Chinese online movie reviews and opinion leadership identification,”
International Journal of Human–Computer Interaction, vol. 36, no. 3, pp. 211–226, 2020.

[5] T. Jiang, H. Yu, and B. Zhang, "Tibetan text classification using distributed representations of words," in Proceedings of the 2015 International Conference on Asian Language Processing (IALP), pp. 123–126, Suzhou, China, October 2016.

[6] G. Xu, C. Wang, H. Yao, and Q. Qi, "Research on Tibetan hot words, sensitive words tracking and public opinion classification," Cluster Computing, vol. 22, no. 4, pp. 9977–9990, 2017.

[7] C. Xu, J. Su, and S. Chen, "Exploring efficient grouping algorithms in regular expression matching," PLoS One, vol. 13, no. 10, Article ID e0206068, 2018.

[8] A. V. Aho and M. J. Corasick, "Efficient string matching," Communications of the ACM, vol. 18, no. 6, pp. 333–340, 1975.

[9] D. Ficara, A. Di Pietro, S. Giordano, G. Procissi, F. Vitucci, and G. Antichi, "Differential encoding of DFAs for fast regular expression matching," IEEE/ACM Transactions on Networking, vol. 19, no. 3, pp. 683–694, 2011.

[10] X. Pengqiang, "Sensitive information filtering algorithm based on text information network," Computer Engineering and Design, vol. 09, pp. 2447–2452, 2016.

[11] Z. Yunxiang, B. Jianming, and J. Shengcan, "Multi-level deep filtering simulation study of terrestrial network sensitive information," Computer Simulation, vol. 05, pp. 466–470, 2019.

[12] H. Guimin, L. Guangming, and Z. Yi, "FSS-GCN: a graph convolutional networks with fusion of semantic and structure for emotion cause analysis - sciencedirect," Knowledge-Based Systems, vol. 212, Article ID 106584, 2020.

[13] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 974–983, New York, NY, USA, August 2018.

[14] T. J. Fu and W. Y. Ma, GraphRel: Modeling Text as Relational Graphs for Joint Entity and Relation Extraction, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1409–1418, 2019.

[15] D. Marcheggiani and I. Titov, Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1506–1515, 2017.

[16] Z. Lei, C. Juan, S. Qiang, G. Junbo, and W. Ziang, Integrating Semantic and Structural Information with Graph Convolutional Network for Controversy Detection, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 515–526, 2020.

[17] Z. Lin, M. Feng, C. Santos et al., "A structured self-attentive sentence embedding," in Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, May 2017.

[18] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," in Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, July 2018.

[19] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, vol. 2, no. 2, pp. 464–468, 2018.