IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 14, 2020, accepted February 5, 2020, date of publication February 10, 2020, date of current version February 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2972751

Hierarchical Graph Transformer-Based Deep Learning Model for Large-Scale Multi-Label Text Classification

JIBING GONG[1,2,3], ZHIYONG TENG[1,2], QI TENG[4], HEKAI ZHANG[1,2], LINFENG DU[4], SHUAI CHEN[1,2], MD ZAKIRUL ALAM BHUIYAN[5], JIANHUA LI[6], MINGSHENG LIU[7], AND HONGYUAN MA[8]

[1]School of Information Science and Engineering, Yanshan University, Qinhuangdao 066004, China
[2]Key Laboratory for Computer Virtual Technology and System Integration, Yanshan University, Qinhuangdao 066004, China
[3]Key Laboratory for Software Engineering of Hebei Province, Yanshan University, Qinhuangdao 066004, China
[4]School of Computer Science and Engineering, Beihang University, Beijing 100083, China
[5]Department of Computer and Information Sciences, Fordham University, New York, NY 10458, USA
[6]School of Information Science and Technology, Shijiazhuang Tiedao University, Shijiazhuang 050043, China
[7]College of Electrical Engineering, Hebei University of Technology, Tianjin 300401, China
[8]National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China

Corresponding author: Jianhua Li (lijh128@163.com)

ABSTRACT Traditional methods of multi-label text classification, particularly deep learning, have achieved remarkable results. However, most of these methods use word2vec technology to represent sequential text information, while ignoring the logic and internal hierarchy of the text itself. Although these approaches can learn the hypothetical hierarchy and logic of the text, it is unexplained. In addition, the traditional approach treats labels as independent individuals and ignores the relationships between them, which not only does not reflect reality but also causes significant loss of semantic information. In this paper, we propose a novel Hierarchical Graph Transformer based deep learning model for large-scale multi-label text classification. We first model the text into a graph structure that can embody the different semantics of the text and the connections between them. We then use a multi-layer transformer structure with a multi-head attention mechanism at the word, sentence, and graph levels to fully capture the features of the text and observe the importance of the separate parts. Finally, we use the hierarchical relationship of the labels to generate the representation of the labels, and design a weighted loss function based on the semantic distances of the labels. Extensive experiments conducted on three benchmark datasets demonstrated that the proposed model can realistically capture the hierarchy and logic of text and improve performance compared with the state-of-the-art methods.

INDEX TERMS Multi-label text classification, graph modeling, graph transformer, deep learning.

I. INTRODUCTION

Text classification is a significant and classical problem in natural language processing [1]. One of the major topics to be investigated in this field is multi-label hierarchical text classification (MLHTC), which aims to assign (tag) a text with multiple appropriate labels from hierarchical label structures. Such a structure can be a tree or a directed acyclic graph indicating the parent-child relations between labels [2]. MLHTC methods have been utilized in an extensive range of applications, including question answering, online shopping, and news tag organization [3]. It is an extremely meaningful classification problem in real-world situations. In contrast to single label or multi-class classification, the key challenges of MLHTC involve utilizing its powerful text representation, feature extraction, and label structure relationship exploration capabilities.

The text representation model is a fundamental but challenging issue for natural language processing tasks [4]. Traditional text representation utilizes the vector space model founded on the bag of words/phrases representation [5]. To handle downstream tasks, most natural language processing tasks leverage the vector space model to model (represent) text, owing to its simplicity and effectiveness [6], [7].

The associate editor coordinating the review of this manuscript and approving it for publication was Kim-Kwang Raymond Choo.

However, the vector space model loses a significant amount of structural and semantic information that is useful for text categorization. Recently, graph-based architecture has attracted increasing research attention for use in social networks and recommendation systems [8]–[10]. It utilizes graphs to organize different types of data, which better reflects real-world situations, and has achieved satisfactory results. In contrast, there is limited research on graph-based documentation representation [11]. Compared to bag of words/phrases representation, graph-based document modeling can preserve local sequential, non-consecutive, and long-distance semantics, and consequently provides improved classification accuracy [12], [13].

In recent years, deep learning models have achieved state-of-the-art results across many domains, including a wide variety of natural language processing (NLP) applications [14], [15]. These models can combine feature extraction with a classifier to perform end-to-end learning of text classification. Examples include recurrent neural networks (RNNs) [16]–[19] and convolutional neural networks (CNNs) [20]–[22]. RNNs can effectively process the semantics of short text but are less able to capture semantic features of longer text. Although bidirectional RNNs were proposed to solve the aforementioned problem and perform efficiently in many NLP tasks, the problem of training efficiency is still unsolved. Unlike RNNs, CNNs use different window sizes to perform one-dimensional convolution of word vectors for all words in a sentence (available for some information before and after, similar to an implicit n-gram), and then use the maximum pool to obtain the most important impact factor for processing downstream tasks. However, owing to their n-gram-like mechanisms, the long-distance semantic dependency among the words can be lost. In summary, the existing deep learning methods cannot simultaneously capture the non-consecutive, long-distance, and sequential semantics of text.

Furthermore, unlike general single label or multi-class tasks [23], an MLHTC must consider the independence and hierarchies of distinct labels and imbalanced label space. However, general NLP models based on deep learning can only learn the features of the text, and have difficulty learning the intrinsic relationships between discrete labels. While some proposed transfer learning methods consider the relations between hierarchical labels by sharing the weights from different local models, one critical issue is that the number of local classifiers depends on the depth of the label hierarchy. This makes transfer learning impracticable for large-scale text [24].

Inspired by recent work on graph representation learning and attention neural networks [25]–[27], we propose a novel Hierarchical Graph Transformer based deep learning model called HG-Transformer for large-scale multi-label text classification. Our framework is composed of three substantial components: graph-based document modeling, hierarchical transformer encoder architecture for features extraction, and weight-directed loss for label classification. The remainder of

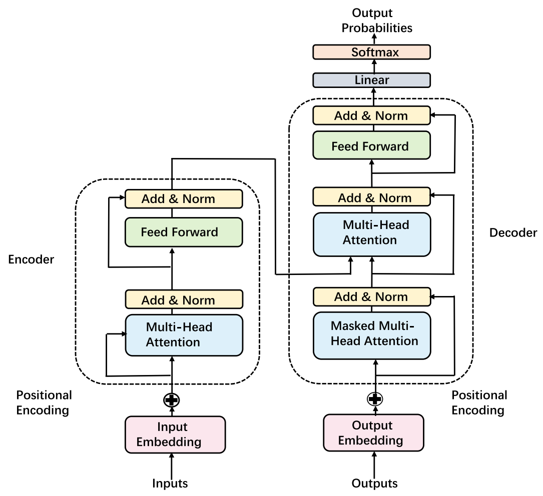


FIGURE 1. Transformer - model architecture.

this paper is organized as follows: preliminaries are described in Section 2, the architecture of our model is described in Section 3, and the effectiveness of the proposed model is demonstrated in Section 4, using tests on benchmark datasets and comparisons with the state-of-the-art methods.

II. PRELIMINARIES

Attention: The attention mechanism was first introduced by Bahdanau to resolve long source sentences in neural machine translation (NMT). More recently, Google redefined the attention mechanism known as "Scaled Dot-Product Attention." First, they use a neural network to map the original input into three matrices: query, key, and value. They then compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and adopt a softmax function to obtain the weights (attention) on the values. Next, they use the weighted value as the representation of each word. The complete calculation formula is as follows:

$$Q, K, V = ([W^Q, W^K, W^V])h \tag{1}$$

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2}$$

Transformer: This method was proposed by Google for seq2seq tasks (such as language translation) in which architectures forgo deep neural networks such as recurrent models and convolution models and instead rely completely on attention mechanisms to obtain global dependencies [28]. The main point of the model is to compute the self-attention between the different words in a sequence and then re-represent the words in a sensible manner. It proved to be extremely efficient compared with traditional deep neural networks. The overall structure of the transformer consists of an encoder and a decoder, as shown in the left and right halves of Fig. 1, respectively. The encoder is composed of a stack of six identical layers, and each layer contains two sublayers, namely a multi-head self-attention layer and a sample feed-forward network. In contrast, the decoder introduces

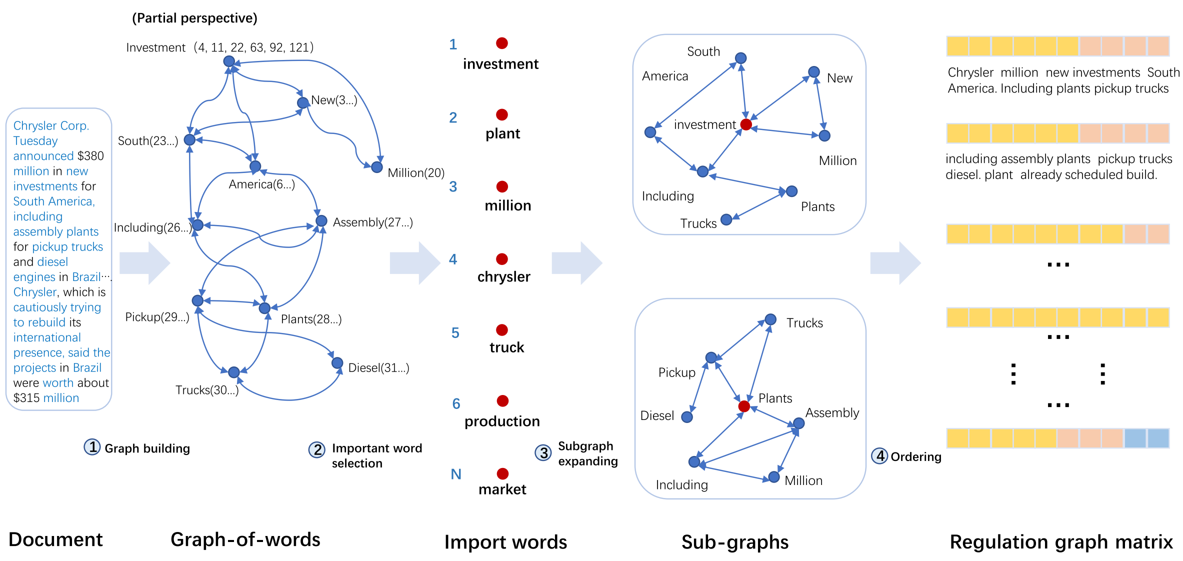

(Partial perspective)
Chrysler Corp. Tuesday announced $380 million in new investments for South America, including assembly plants for pickup trucks and diesel engines in Brazil··· Chrysler, which is cautiously trying to rebuild its international presence, said the projects in Brazil were worth about $315 million
Investment (4, 11, 22, 63, 92, 121)
New(3...)
South(23...)
Million(20)
America(6...)
Including(26...)
Assembly(27...)
Pickup(29...)
Plants(28...)
Diesel(31...)
Trucks(30...)
① Graph building
② Important word selection
③ Subgraph expanding
④ Ordering
1 investment
2 plant
3 million
4 chrysler
5 truck
6 production
N market
Chrysler million new investments South America. Including plants pickup trucks
including assembly plants pickup trucks diesel. plant already scheduled build.
Document
Graph-of-words
Import words
Sub-graphs
Regulation graph matrix

FIGURE 2. Graph-based document modeling.

multi-head attention over the output of the encoder in addition to the two sublayers. We note that the author also employs residual connections and layer normalization to avoid losing original information. Readers can refer to [28] and [29] for a more detailed description of the original architecture.

III. HIERARCHICAL GRAPH TRANSFORMER

We design a Hierarchical Graph Transformer for large-scale multi-label text classification. Given a certain document, we first covert sequence information to a graph of vectors. We then use a three-layer transformer encoder to perform self-attention at the word, sentence, and graph levels to obtain a richer text representation that considers the logic and internal hierarchy of the text. Finally, we design a novel loss function based on label similarity in order to learn the hierarchy and dependencies of the tags. Fig. 2 illustrates the overall framework of the HG-Transformer.

A. GRAPH-BASED DOCUMENT MODELING

Unlike in traditional text representation, the graph is used to model documents to better capture long-distance semantics and the internal hierarchy of the text. Graph-based document modeling consists of three components: the graph of word co-occurrence relationships, an important word expanded subgraph, and a regulation graph matrix.

1) GRAPH OF WORD CO-OCCURRENCE RELATIONSHIPS

Given a document, we first perform sentence tokenization processing to split the document into several sentences. Then, word tokenization is utilized to divide each sentence into its component words. Simultaneously, lemmatization and stop word removal processing standardize words to their base form and remove words with no significant meaningful features. For example, a sentence such as "My system keeps crashing! His crashed yesterday, ours crashes daily" will be transformed to "My system keep crash! his crash yesterday, ours crash daily." After we obtain the processed document, we adopt a sliding window to obtain local co-occurrence relationships and build the word co-occurrence matrix. We then regard the vertex and positional index of a word appearing in the document as its attribute, and the co-occurrence relationships as its edge; as such, we obtain a graph of word co-occurrence relationships, which can be denoted as G = (V, E, N, P). V denotes the word set and $|V| = n$, E denotes the co-occurrence set and $|E| = m$, N denotes the number of co-occurrences, and P denotes the word position in the document. For example, in the first section of Fig. 2 we obtain a graph of word co-occurrence relationships.

2) IMPORTANT WORD EXPANDED SUBGRAPH

After we obtain the graph of word co-occurrence relationships, we can select the top-N significant words according to their contribution ranking. Here, we adopt TF-IDF (term frequency-inverse document frequency) to calculate word contributions. First, we can calculate the TF-IDF feature for each word and rank them from large to small. The top-N words are selected as the root of the subgraph. We then expand each word into subgraphs using breadth-first search (BFS) and depth-first search (DFS). The size of the subgraphs is limited to K nodes. Finally, we obtain N subgraphs, and each subgraph contains both the non-consecutive and long-distance information for the important words in the text, as shown in step 2 and step 3 of Fig. 2. Next, we discuss how to build a regulation matrix that is readable by our learner model.

3) REGULATION GRAPH MATRIX

Although the subgraph can preserve both the non-consecutive and long-distance information of an important word v in the text, it loses the sequence information of the word in the original document. Thus, we must convert each subgraph into a sequence of words by considering the vertex attribute. For example, in a subgraph, we can use the node attribute that indicates the word's position in the original document to order the node. In this manner, we can generate reconstructed text that contains the non-consecutive, long-distance, and sequence information of the subgraph. As shown in the first line of the regulation graph matrix in step 4 of Fig. 2, we convert the first subgraph G(v) into a sequence such as "Chrysler million new investments South America. Including plants pickup trucks." We note that the word sequence may not match that of the sentence in the original document; thus, we split each word sequence into several sentence blocks by utilizing punctuation information.

We also rank sentence blocks by their length and select the top-M sentence blocks. Meanwhile, we limit the number of sentence blocks in each subgraph to S and limit the number of words in each sentence block to W, which guarantees the consistency and regularity of text data. For example, we can choose the top 10 sentences as a candidate sentence block and choose the average length of each sentence block as its final length. If the number of sentences blocks in a sub-graph less than 10, we use zero to pad it and if the number of words less than average length, we also use zero to pad it. Finally, for each document, we obtain an $N \times M \times W$ three-dimensional matrix, where N denotes the top-N important words, M denotes the number of sentence blocks in each subgraph, and W denotes the number of words in a sentence block. To further improve the validity of the text representation, we use word2vec to represent each word as a dense vector of real numbers [30], [31]. Specifically, we set the dimension of the word vector to D. Through this step, we can finally obtain an $N \times M \times W \times D$ four-dimensional matrix, where D denotes the dimension of the word vector, as shown in Fig. 2.

B. HIERARCHICAL TRANSFORMER ENCODER ARCHITECTURE

In this section, we introduce the hierarchical transformer encoder model. After converting each document into a 4-D matrix representation, we utilize a three-layer transformer encoder model to learn both the hierarchy and logic features of the text. Unlike the standard transformer, we only use the encoder component, because we require the text extraction feature for the classification task. The overall framework of our model is shown in Fig. 3.

Let $\mathcal{D}$ denote a document comprised of a sequence of N_d subgraphs. $\mathcal{D} = \{g1, g2, \ldots, gN_d\}$. Each subgraph g is comprised of a sequence of sentence blocks $\mathcal{G} = \{s1, s2, \ldots, sN_g\}$, where Ng denotes the number of sentences blocks. Each sentence block is comprised of a sequence of tokens $\mathcal{S} = \{w1, w2, \ldots, wN_s\}$, where N_s denotes the length of the sentence. Additionally, each word w is represented by an embedding vector.

We first employ the transformer encoder over the word level, in order to take into account the relationships between

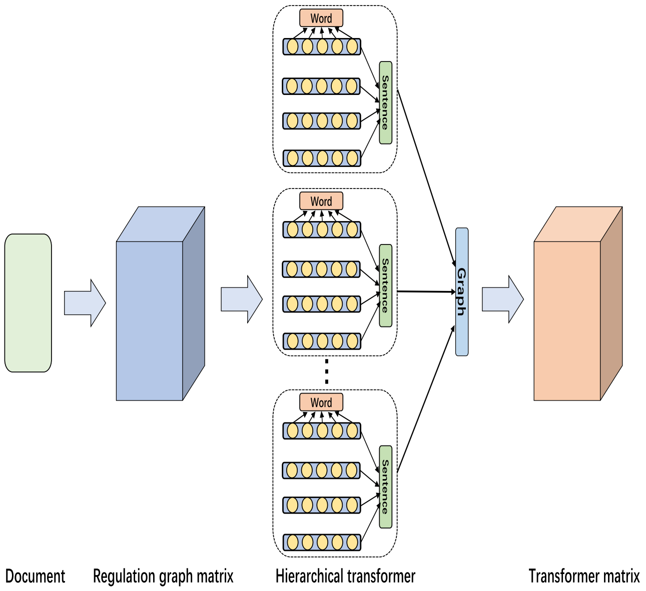


FIGURE 3. Hierarchical transformer model.

words in the same sentence. The words in a sentence have a common context, and each word is a unit of expression for the semantics of the sentence; thus, it is wise to limit the focus of words to the scope of the sentence. In fact, the transformer encoder maps an input sequence of word representations $(w_1, w_2, \ldots, wNs)$ to a sequence of continuous representations $z = (z1, z2, \ldots zNs)$. The procedure can be summarized as follows:

$$Attention(S) = softmax(\frac{SS^T}{\sqrt{d_k}})S \tag{3}$$

$$Z^1 = Norm(S + Attention(S)) \tag{4}$$

$$Z^2 = Norm(Z^1 + FFN(Z^1)) \tag{5}$$

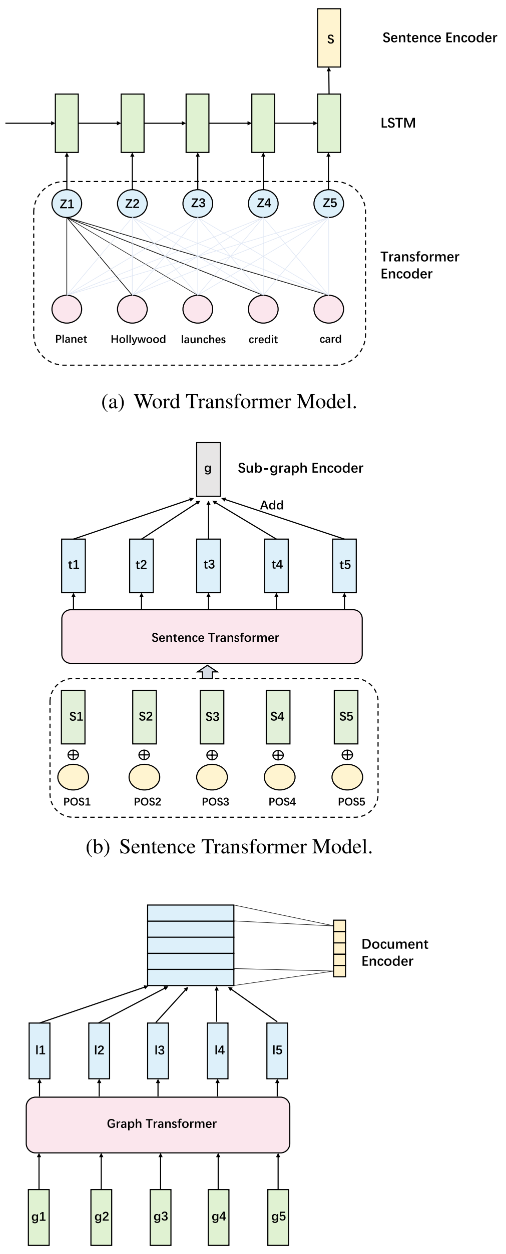
where $S \in \mathbb{R}^{(N_s,d_w)}$, $Z^2 \in \mathbb{R}^{(N_s,d_z)}$. We simply set $d_w = d_z$

After the transformer encoder process, each word captures the semantics of the other words in the sentence. However, because we do not encode the position of the word, we will lose some of the position information. To alleviate this disadvantage, we introduced LSTM (long short-term memory networks) to sequence-model the re-encoded words. Note that although ordinary LSTM will lose the semantic information of the previous text because of the length of the sentence, the words re-encoded by the transformer will retain the context information, thus solving the inherent shortcomings of LSTM. The process of LSTM is as follows:

$$h_t^z(enc) = LSTM_{encode}^{word}(z_t, h_{(t-1)}(enc)) \tag{6}$$

As Fig.4 (a) shows, the vector output at the ending time-step is used to represent the entire sentence.

After we obtain the sentence encoder, we use the transformer encoder at the sentence level. Unlike the word-level transformer, we consider the position of a sentence in a subgraph and combine it with the sentence semantic vector



(a) Word Transformer Model.

(b) Sentence Transformer Model.

(c) Subgraph Transformer Model.

FIGURE 4. Hierarchical transformer encoder architecture.

to represent the sentence. In this manner, the transformer can capture the relationship between different sentences and the position information of the sentences. The sentence-level transformer can map a sequence of sentence block representations$(s1, s2, \ldots, sN_g)$ to a sequence of continuous representations $t = (t1, t2, \ldots tN_g)$.

Unlike the word-level transformer, we do not use LSTM to process the transformer-recoded sentences, but use a simple addition operation to represent the final subgraph encoding. This not only saves a substantial amount of computing resources, but also generates the ideal subgraph encoding. The model is shown in Fig.4 (b)

Similar to the previous process, after obtaining the graph encoder we employ the transformer encoder over the subgraph level to take into account the relationships between the subgraphs in a document. Unlike at the word level and sentence level, we do not consider location information at the graph level. Because the subgraph is extended from important words, we assume that each important word has equal status in terms of position. The graph level transformer can map a set of subgraph representations $(g1, g2, \ldots, gN_g)$ to a set of continuous representations $l = (l1, l2, \ldots lN_l)$. Unlike the sequential structure of the words in the previous sentence, the sentence and word information in the subgraph do not have sequence characteristics. Therefore, we introduce convolutional neural networks (CNNs) for feature extraction in subgraphs. The model is shown in Fig.4 (c)

The final result sequentially extracts the features of the text at the word, sentence, and subgraph levels, which can further capture the hierarchy and logic information of the document. We note that a word will not only pay attention to other words in a sentence but also generate attention for words in other sentences through the sentence-level transformer. Similarly, a sentence can obtain information from the sentences in other graphs through the graph-level transformer. Traditional transformers treat words as the fundamental elements of a document. Each word will pay attention to all the words in the document, and ignore the context in which the word is located. Hierarchical transformers are different from traditional flat transformers. We believe that a word should pay more attention to the words in the sentence it belongs to than to the words in other sentences. Similarly, the attention between sentences in one sense-group will be greater than their attention to sentences in other sense-groups (we can think of a subgraph as a sense-group). Therefore, the hierarchical transformer can obtain the semantic information of the document in a more reasonable and efficient manner. After obtaining the encoding of the document, we can use it for various tasks downstream.

C. HIERARCHICAL SIMILARITY-BASED WEIGHTED CROSS ENTROPY LOSS

The traditional classification method treats a label as an independent individual, without considering the relationships between labels. This can result in a significant loss of tag information, which will affect classification accuracy. Moreover, the unbalanced distribution of labels (that is, fewer instances of most leaf labels will appear) will cause the training process to be insufficient, and no corresponding features will be learned for a small number of labels. Therefore, we propose a weight-directed loss function based on label similarity. It considers the similarity between tags and can solve the problem of uneven distribution of labels.

First, we define the hierarchical structure of the label as $\mathcal{L} = (L, E)$. L represents the label, and E represents the parent-child relationship between the labels. We must choose the label representation to calculate the similarity of the labels. Naturally, we can use the word vector of the label

TABLE 1. Dataset statistics.

Dataset	Class Labels	Training	Test	Avg. Instances per Label	Avg. Labels per Instances
RCV1	103	23,149	784,446	729.67	3.24
RCV 1-2K	2,456	623,847	155,962	1218.56	4.79
AmazonCat-14K	13,330	1,186,239	306,782	448.57	5.04

to represent the label, because the semantic information of the label specifying the parent-child relationship is similar. However, the simple semantic word vector only considers the meaning of the labels literally and does not consider the hierarchical relationship between the labels. In this manner, the representation of the label will lose the most important structured information, which is not conducive to the measurement of similarity.

Inspired by the recent work on node embedding, we use random walks to generate label sequences as corpus information, and then use skip-gram to train the label sequences to convert label representations into a continuous vector space $V_l \in \mathbb{R}^D$. D denotes the embedded dimension of the label.

After obtaining the representation of the label, we use the Euclidean distance to calculate the similarity between the different labels:

$$Dis(i,j) = \sqrt{\sum_{k=1}^{D}(V_k^{l_i} - V_k^{l_j})^2} \tag{7}$$

Then, the similarity can be defined as:

$$Sim_{i,j} = 1 - \frac{Dis(i,j)}{\sum_{k=1}^{|L|} Dis(i,k)} \tag{8}$$

Next, in order to explore the relevance and hierarchy of labels, we designed a similarity-based weight loss function based on label similarity:

$$L = -\sum_{i=1}^{n}\sum_{k=1}^{L}[y_k^i \log \hat{y}_k^i + \lambda\alpha_k^i(1-y_k^i)\log(1-\hat{y}_k^i)] \tag{9}$$

where $y_k = 1$ if and only if a digit of class k is present. λ is a hyperparameter to control the weight of similarity. N denotes the number of documents and L denotes the number of labels. $\hat{y} \in [0, 1]$ denotes the positive probability. $\alpha_k \in [0, 1]$ is the minimum distance from negative label k to the positive labels set. Specifically, for a text ts, the positive label set is $P_s \subset \mathcal{S}$. Additionally, for any negative label k, the α_k is:

$$\alpha_k = \max_{l \in P_s}(Sim_{l,k}) \tag{10}$$

IV. EXPERIMENTS

A. EXPERIMENT SETUP

Datasets: We conducted extensive experiments using three publicly available datasets from various domains (summarized in Table 1). The first two datasets, RCV1 and RCV 1-2K, are related to news categorization. The third dataset is AmazonCat-13K, which is related to product categorization. The detailed descriptions of each dataset are as follows:

- **Reuters Corpus Volume I (RCV1)** [32]. RCV1 dataset is a manually labeled newswire collection of Reuters News from 1996 - 1997. The news documents are categorized with respect to three controlled vocabularies: industries, topics, and regions. We use the topic-based hierarchical classification as it has been the most popular in previous evaluations. There are 103 categories, including all classes in the hierarchy (except for root).
- **RCV1-2K**. The RCV1-2K dataset has the same features as the original RCV1 dataset but its label set has been expanded by forming new labels from pairs of original labels. There are 2456 labels in the RCV1-2K dataset.
- **AmazonCat-13K**. This dataset includes reviews (ratings, text, helpfulness votes), product metadata (descriptions, category information, price, brand, and image features), and links (also viewed/also bought graphs). In this paper, we focus on category information.
- **Evaluation Metrics.** We use standard rank-based evaluation metrics P@K (precision at k) and NDCG@K (normalized discounted cumulative gain at k) to measure the performance of all the methods [33]. For P@K and NDCG@K, each $document_t$ has a set of $|\mathcal{L}|$ ground truth labels $L_t = \{l_0, l_1, l_2 \ldots, l_{|\mathcal{L}|-1}\}$ and a list of $\mathcal{Q}$ predicted labels, in order of decreasing probability $P_t = [p_0, p_1, p_2 \ldots, p_{\mathcal{Q}-1}]$. The precision at k is $P@K = \frac{1}{k}\sum_{j=0}^{min(|\mathcal{L}|,k)-1} rel_{L_i}(P_t(j))$, where

$$rel_L(p) = \begin{cases} 1 & if\ p \in L, \\ 0 & otherwise. \end{cases}$$

The NDCG at k is $NDCG@K = \frac{1}{IDCG(L_i,k)} \sum_{j=0}^{n-1} \frac{rel_{L_i}(P_t(j))}{\ln(j+1)}$, where $n = \min(\max(|P_i|, |L_i|), k)$

B. COMPARED METHODS

Flat baselines: These methods generally extract discrete features such as the TF-IDF of the document and utilize them to train a classification model. The typical representatives of this method are LR (logistic regression) and SVMs (support vector machines). We define them as ïĆat baselines because they ignore both the relations among the words and the relations among the labels and simply train a multi-class classifier.

Neural Network Models: Many neural network models have been proposed for multi-label text classification problems. For comparisons, we mainly use improved CNN- and RNN-based models such as XML-CNN [3], Deep CNN [34], HLSTM [35], HMCN-F [36], and HAN [37]. XML-CNN introduces the CNN structure to handle multi-label text classification problems. RNN models such as Hierarchical Long Short-term Memory Network (HLSTM) and Hierarchical Attention Network (HAN) both utilize a two-layer RNN structure to extract text features at the word level and sentence level, respectively; HAN also introduces attention weights to account for the effects of different locations. HMCN-F fits its CNN layers to the label hierarchy, and each CNN layer focuses on predicting the labels at the corresponding hierarchical level.

Hierarchical Models: These methods employ hierarchical or graphical label networks to build hierarchical classification classifiers. Examples of hierarchical models include Hierarchically Regularized Logistic Regression (HR-LR), Hierarchically Regularized Support Vector Machines (HR-SVM) [38], and Hierarchically Regularized Deep Graph CNN (HR-DGCNN-3) [11]. For example, HR-DGCNN uses graph structures to extract text features and utilizes regulars to consider label relationships.

Tree and Embedding Based Methods: These methods mainly focus on large-scale multi-label classification. Examples of such methods include FastXML [39], SLEEC [40], and Parabel [41]. SLEEC (Sparse Local Embeddings for Extreme Classification) projects labels into low-dimensional vectors that can capture label relations and uses the k-nearest neighbors when predicting. FastXML builds a tree-based extreme multi-label classifier to handle MLHTC problems, and includes a novel node partitioning formulation to speed up the training process. Parabel [41] learns a balanced tag hierarchy and generalizes the hierarchical softmax model to save computing resources.

Variations of HG-Transformer: The Transformer model has achieved state-of-art performance in most NLP tasks. The basic structure of our proposed model is inspired by Transformer architecture. In order to distinguish the two, we call the general Transformer a Flat Transformer, and refer to our proposed model as the Hierarchical Transformer. We implemented several variants of these two types of Transformers as follows. Flat Transformer (F-Transformer): general Transformer without graph-based document modeling and hierarchical similarity-based weighted cross entropy loss. Flat Graph Transformer (FG-Transformer): without hierarchical similarity-based weighted cross entropy loss. Flat Transformer with Weighted Loss (F-Transformer-W): without graph-based document modeling. Hierarchical Graph Transformer (HG-Transformer(No W)): without hierarchical similarity-based weighted cross entropy loss.

C. EXPERIMENTAL SETTINGS

All our experiments were performed on a 64-core Intel Xeon CPU E5-2680 v4@2.40GHz with 512GB RAM and eight NVIDIA Tesla P100-PICE GPUs. The operating system and software platforms were Ubuntu 5.4.0, Python 3.6.2, and Pytorch 0.4.0. The training and testing datasets are shown in Table1. In the document modeling part, the top-N number of important words is set to 100 (RCV1). The maximum number of nodes in each subgraph is set to 50. The maximum number of sentences per subgraph is set to 5, and the maximum length of each sentence is set to 10. We use GloVe [42] (Global Vectors for Word Representation) with size 50 as word embeddings for last document representation. For label representation, we use node2Vec technology to generate vector representation of the labels, with a dimension of 50. All models are trained using an Adam optimizer with an initial learning rate of 1e-6 and a weight decay of 1e-6.

D. PERFORMANCE COMPARISON

We compare the performance of our proposed method HG-Transformer to state-of-the-art MLHTC methods and show the results in Tables 2.

On RCV1, we can see that for traditional methods, HR-SVM performs better than LR, SVM, and HR-SVM. One of the main reasons is that HR-SVM is more complex and can capture the nonlinear space feature.

For deep neural network methods, we can see that HLSTM, HAN, and RCNN even achieve worse performance than traditional methods. The results above illustrate that the traditional deep neural network model does not provide as much of an advantage as it does in other tasks. These recurrent models are not suitable for long text tasks. However, DCNN achieves the best performance among deep neural network methods and it illustrates that the CNN model is more suitable for text classification than the RNN model. Specifically, HR-DGCNN achieves better performance than HMCN-F on P@K metrics while HMCN-F outperforms HR-DGCNN on NDCG@K metrics. They all take advantage of the label hierarchy information.

For embedding and tree-based models, one can see that SLEEC achieves better performance than FastXML. One of the main reasons is that SLEEC can learn embeddings that preserve pairwise distances between only the nearest label vectors. Parabel achieves better performance than SLEEC on all five metrics, which illustrates the significance of label hierarchy information.

For the Transformer models, one can see that the graph-based document modeling, hierarchical transformer mechanism, and hierarchical similarity-based weighted cross entropy loss are all instrumental to improving classification performance. Specifically, F-Transformer outperforms most traditional models and neural network models, and demonstrates that the Transformer model is effective in extracting text features in multi-label text classification. Meanwhile, F-Transformer-W achieves better performance than F-Transformer, which shows that the model benefits from the hierarchical similarity-based weighted cross entropy loss. Moreover, FG-Transformer achieves better performance than F-Transformer, thus demonstrating that graph-based

TABLE 2. Results in P@k and NDCG@k.

Datasets	RCV 1					RCV 1-2K					AmazonCat-13K				
Metrics	P@1	P@3	P@5	NDCG@3	NDCG@5	P@1	P@3	P@5	NDCG@3	NDCG@5	P@1	P@3	P@5	NDCG@3	NDCG@5
LR	72.65	53.75	41.68	67.15	70.54	70.21	51.92	40.1	65.43	68.42	68.45	49.22	39.89	63.93	66.10
SVM	74.91	53.87	42.11	67.64	70.60	71.33	53.37	41.52	66.32	68.10	69.23	50.09	40.56	64.50	66.73
HR-LR	75.24	54.96	42.38	68.47	71.22	72.58	54.89	42.65	65.15	67.4	69.97	51.33	40.98	65.07	67.42
HR-SVM	76.8	55.01	44.59	70.73	72.42	72.47	55.03	42.78	68.22	69.34	70.1	52.77	41.67	65.82	67.93
HLSTM	73.14	54.81	42.61	67.80	71.03	71.6	52.73	42.11	68.23	70.60	69.58	50.34	39.94	66.64	68.69
HAN	73.96	54.07	43.22	67.91	71.45	72.18	53.1	43.7	68.92	70.87	70.32	51.26	40.52	66.81	69.12
RCNN	74.18	54.90	43.71	66.50	71.97	73.84	54.78	44.78	67.10	68.94	71.20	52.90	41.31	67.33	67.19
DCNN	78.57	58.3	47.35	69.04	76.12	75.22	55.99	45.72	68.70	70.01	73.66	53.47	43.07	69.70	70.32
HR-DGCNN	95.29	80.32	55.38	90.02	90.28	95.07	78.02	55.38	90.11	89.65	94.38	81.32	63.64	87.77	86.04
FastXML	93.25	76.37	53.21	88.64	89.84	93.23	77.52	53.96	89.11	87.05	93.11	78.20	63.41	87.07	85.16
SLEEC	94.45	78.60	54.24	90.05	90.32	94.33	77.59	55.14	90.08	87.17	90.53	76.33	61.52	84.96	82.77
Parabel	95.27	78.86	54.44	90.69	90.44	93.95	76.61	55.47	88.65	86.22	93.03	79.16	64.52	87.72	86.00
HMCN-F	95.35	78.95	55.90	90.14	90.82	95.11	78.23	55.44	90.02	88.64	94.43	79.64	64.91	88.14	88.82
F-Transformer	89.33	72.65	49.45	84.78	85.94	88.93	71.44	50.84	84.33	84.03	87.36	70.58	50.36	83.17	82.49
F-Transformer-W	91.5	73.08	50.22	85.54	87.16	89.89	72.91	51.72	86.17	85.46	88.16	71.63	51.47	84.86	83.57
FG-Transformer	92.78	74.51	52.97	86.44	89.3	92.19	74.2	52.86	88.31	88.63	91.33	74.55	54.7	85.31	84.66
HG-Transformer(No W)	94.76	79.59	54.34	88.7	90.05	94.65	76.11	54.28	89.43	89.09	92.3	79.21	63.85	87.94	87.62
HG-Transformer	95.8	80.98	55.96	90.03	91.96	95.46	78.18	55.67	91.05	90.18	94.44	81.37	64.95	89.03	89.15

document modeling can preserve more text semantic information than sequential text modeling. Compared to the flat transformer, one can see that HG-Transformer(No W) performs better. These improvements show that the combination of graph-based document modeling and the hierarchical transformer can better extract text features. Finally, HG-Transformer achieves the best performance on the five metrics, which again demonstrates that graph-based document modeling, the hierarchical transformer mechanism, and hierarchical similarity-based weighted cross entropy loss are all useful in improving classification performance.

On RCV 1-2K and AmazonCat-14K, similar results are observed: the graph-based text modeling method, Transformer model, and hierarchical label similarity methods perform well, and our proposed method HG-Transformer again achieves the best performance on all metrics.

V. EVALUATION ON GRAPH-BASED DOCUMENT MODELING

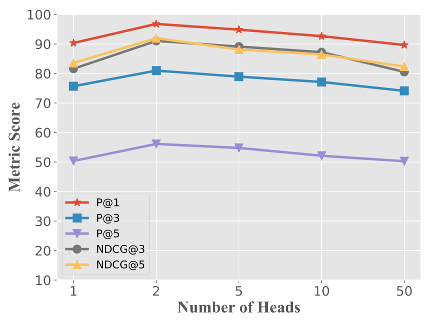
In order to better capture the semantic features of the text, we explore the number of subgraphs in graph-based document modeling. We tested with different numbers of subgraphs using the RCV1 dataset, and the results are shown in Table3. One can see that the number of subgraphs has a significant effect on classification performance. Subgraphs are extended from important words and the number of subgraphs is the same as important words. Additionally, the nodes in each subgraph, that is, words, are directly or indirectly related to important words. Therefore, the selection of the appropriate number of important words will affect classification performance. If the number of selected important words is too small, the semantic information in the text will be lost and the model cannot fully extract the semantics. If the number of selected important words is too large, some redundant information will be introduced and the model will learn useless information. As the table shows, the model has the lowest performance on all metrics at 20 subgraphs, and at this time, the graph-based document will lose some semantic information. When the number of subgraphs is set to 100, the model performs best on all classification metrics. It can be considered that the graph-based document at this time can fully extract the original semantics of the text. When the number of subgraphs is 120, the model will provide poor classification accuracy owing to the introduction of redundant information. It is vital to choose an appropriate number of subgraphs at the graph-based document modeling process to optimize classification performance.

TABLE 3. Comparison of different numbers of subgraphs on RCV1.

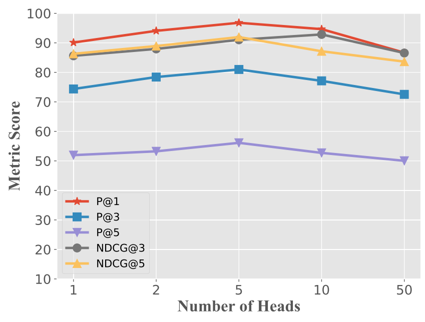
Number of subgraphs	P@1	P@3	P@5	NDCG@3	NDCG@5
20	90.23	74.54	50.07	84.75	86.39
40	91.85	76.98	52.46	85.14	87.93
60	92.2	77.61	54.19	87.36	88.12
80	93.85	78.06	55.4	89.04	90.07
100	95.8	80.98	56.12	90.03	91.96
120	92.47	77.72	53.14	88.51	89.6

VI. EVALUATING DIFFERENT NUMBERS OF HEADS

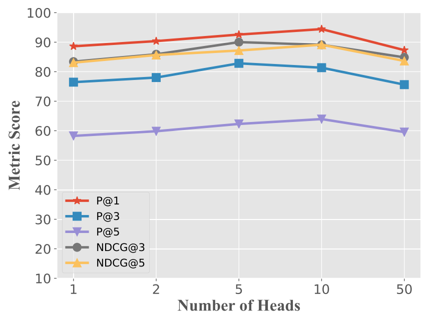
A significant hyperparameter of the Transformer model is the number of heads. We compared results obtained with different numbers of heads through tests on the three datasets. The experimental results are shown in Fig. 5. Fig. 5 shows that on the RCV1 dataset, when the number of heads is two,



(a) RCV1



(b) RCV 1-2K



(c) AmazonCat-13K

FIGURE 5. Comparing performance obtained with different numbers of heads on three datasets.

HG-Transformer performs best on all five metrics. When the number of heads is greater than two, the experimental effect will rapidly decline. Fig. 5 b shows that on the RCV 1-2K dataset, when the number of heads is set to 5, HG-Transformer performs best on P@K and NDCG@5, and when the number of heads is set to 10, NDCG@3 exhibits the highest performance. Fig. 5 c shows that on the AmazonCat-13K dataset, when the number of heads equals 10, p@1, p@5, and NDCG@5 reach their maximum scores. When the number of heads is set to 5, P@3 and NDCG@3 achieve the maximum scores. Overall, the number of heads is directly proportional to the magnitude of labels in the dataset. Because multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions, it can capture features in different dimensions. Each head of the transformer's multi-head mechanism corresponds to a feature subspace. An appropriate number

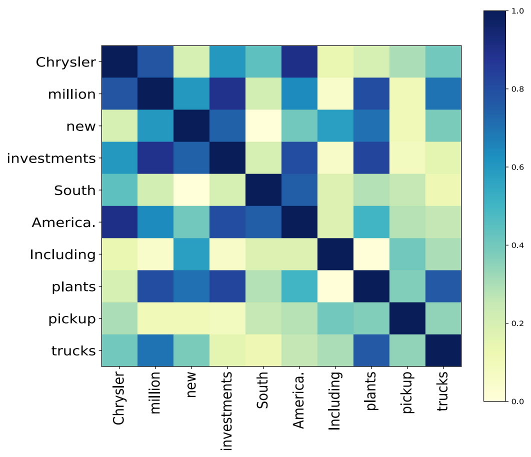


FIGURE 6. Word-level self-attention visualizations for the 2287newsML sample in RCV1.

of feature subspaces can fully learn the potential various feature relationships of the text and its mapping to corresponding labels. Under the premise of our experimental setup, the labels of one hundred, one thousand, and ten thousand levels correspond to 2, 5, and 10 heads, respectively.

VII. EVALUATION ON HIERARCHICAL SIMILARITY-BASED WEIGHTED CROSS ENTROPY LOSS

In order to study whether the proposed hierarchical similarity-based weighted cross entropy loss can obtain better classification results, we utilize the hierarchical similarity-based weighted cross entropy loss and traditional cross entropy loss to train our model; the comparison results are shown in Table4.

On the RCV-1 dataset, we can see that the model trained using hierarchical similarity-based weighted cross entropy loss improves the performance by 1% in terms of P@1 and P@3 and 2% in terms of P@5, NDCG@3, and NDCG@5, compared with the model trained using traditional cross entropy loss. For RCV1 2-K and AmazonCat-13K, we can see that our proposed loss function outperforms the traditional cross entropy loss function and provides an improvement of 1% on average. It can be concluded that our proposed loss function can help improve model performance compared with the traditional cross entropy loss function. One of the main reasons is that the hierarchical similarity-based weighted cross entropy loss function can solve the problem of uneven distribution of categories in the multi-label classification task, and those labels with a small number of samples can be fully trained through the hierarchical structure between the labels.

VIII. CASE STUDY

To further explore the hierarchical transformer mechanism captured in a document, we visualize portions of the word level self-attention probability using heatmaps, as shown in Fig. 6. Fig. 6 shows the self-attention between different words in a sentence block, taking the sentence "Chrysler million new investments South America. Including plants

TABLE 4. Comparison of loss functions.

Datasets	Loss function	P@1	P@3	P@5	NDCG@3	NDCG@5
RCV-1	Traditional cross entropy loss	94.74	78.85	54.04	89.26	89.21
	New cross entropy loss	95.8	80.98	56.12	90.03	91.96
RCV 1-2K	Traditional cross entropy loss	94.91	77.62	53.72	89.16	88.67
	New cross entropy loss	95.46	78.18	55.67	91.05	90.18
AmazonCat-13K	Traditional cross entropy loss	93.35	80.28	62.4	87.69	88.1
	New cross entropy loss	94.44	81.37	63.95	89.03	89.15

pickup trucks'' as an example. We can see that each word pays the most attention to itself. The word Chrysler pays more attention to the words ''American'' and ''million.'' The word ''millions'' has a greater focus on words ''investments'' and ''plans.'' It can be found that allocation of the attention weight of each word to other words in the sentence is consistent with real-world intuition. This proves that our model can capture text features well and is interpretable.

IX. RELATED WORK

In this section, we review related work in the following two aspects.

A. TEXT REPRESENTATION

Text representation is the basis of natural language processing, and reasonable text representation can significantly improve the efficiency of downstream tasks; e.g., text classification, machine translation, sentiment analysis, question and answer systems [43]. In recent years, text representation approaches have been developed significantly for various applications [44]. Many text classification studies focus on the bag-of-words (BOW) approach [45], [46], in which each feature corresponds to a single vocabulary word. Term frequency-inverse document frequency (TF-IDF) is a very common method for transforming text into a meaningful representation of numbers and is widely used for feature construction [47], [48]. Owing to advances in deep learning, the vector representations of words learned by word2vec models based on deep learning have been shown to carry semantic meanings and perform effectively in many NLP tasks [49]. In other instances, researchers have attempted to transform texts into graphs by utilizing word co-occurrence [50].

B. TEXT FEATURE EXTRACTING

Recently, deep learning has shown its powerful capability to handle various NLP tasks [51], [52]. For example, recurrent convolutional neural networks use different window sizes for one-dimensional convolution of word vectors for all words in a sentence to capture text semantic features [53]. Recurrent neural networks (RNNs) can remember some information regarding a sequence by maintaining a hidden state, and are better suited to processing NLP tasks [16]. Additionally, long short-term memory networks (LSTM) and gated recurrent units (GRUs) were proposed to improve the original RNN model, and achieved significant performance improvements [54]–[56]. Recently, the Transformer architecture based on the self-attention mechanism has exhibited state-of-the-art performance in most NLP tasks, which inspired us to propose a hierarchical transformer encoder model.

X. CONCLUSION AND FUTURE WORK

In this paper, we propose a novel Hierarchical Graph Transformer for large-scale multi-label text classification. We first model the text into a graph structure that can embody the different semantics of the text and the connections between them. To fully capture textual information, we introduce a multi-layer transformer structure with a multi-head attention mechanism at the word, sentence, and graph levels to fully capture the features of the text and observe the importance of the different parts. To explore the hierarchical relationship between tags, we utilize the hierarchical relationship of the labels to generate the representation of the label and design a weighted loss function based on the semantic distance of the label. Extensive experiments conducted on three benchmark datasets demonstrated that the proposed model can realistically capture the hierarchy and logic of the text and hierarchical relationship of the labels.

In the future, we plan to upgrade our transformer model by utilizing more powerful BERT (Bidirectional Encoder Representations from Transformers) pre-trained models and increment embedding [57], [58].

REFERENCES

[1] T. Ali and S. Asghar, ''Multi-label scientific document classification,'' *J. Internet Technol.*, vol. 19, no. 6, pp. 1707–1716, 2018.

[2] L. Liu, F. Mu, P. Li, X. Mu, J. Tang, X. Ai, R. Fu, L. Wang, and X. Zhou, ''Neuralclassifier: An open-source neural hierarchical multi-label text classification toolkit,'' in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics, Syst. Demonstrations*, 2019, pp. 87–92.

[3] J. Liu, W.-C. Chang, Y. Wu, and Y. Yang, ''Deep learning for extreme multi-label text classification,'' in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2017, pp. 115–124.

[4] J. K. Frank Cooper, ''Multiobjective feature selection: Classification using educational datasets in an ensemble validation scheme,'' *Data Sci. Pattern Recognit.*, vol. 3, no. 1, pp. 9–34, 2019.

[5] C. C. Aggarwal and C. Zhai, *Mining Text Data*. Berlin, Germany: Springer, 2012.

[6] R. A. Stein, P. A. Jaques, and J. F. Valiati, ''An analysis of hierarchical text classification using word embeddings,'' *Inf. Sci.*, vol. 471, pp. 216–232, Jan. 2019.

[7] M. B. Revanasiddappa and B. S. Harish, ''A novel text representation model to categorize text documents using convolution neural network,'' *Int. J. Intell. Syst. Appl.*, vol. 11, no. 5, pp. 36–45, May 2019.

[8] M. Xie, H. Yin, H. Wang, F. Xu, W. Chen, and S. Wang, "Learning graph-based POI embedding for location-based recommendation," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2016, pp. 15–24.

[9] J. Li, H. Peng, L. Liu, G. Xiong, B. Du, H. Ma, L. Wang, and M. Zakirul Alam Bhuiyan, "Graph CNNs for urban traffic passenger flows prediction," in *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Oct. 2018, pp. 29–36.

[10] H. Peng, J. Li, Q. Gong, Y. Song, Y. Ning, K. Lai, and P. S. Yu, "Fine-grained event categorization with heterogeneous graph convolutional networks," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 1–9.

[11] H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, and Q. Yang, "Large-scale hierarchical text classification with recursively regularized deep graph-cnn," in *Proc. World Wide Web Conf., Int. World Wide Web Conf. Steering Committee*, 2018, pp. 1063–1072.

[12] H. Peng, J. Li, S. Wang, L. Wang, Q. Gong, R. Yang, B. Li, P. Yu, and L. He, "Hierarchical taxonomy-aware and attentional graph capsule RCNNs for large-scale multi-label text classification," *IEEE Trans. Knowl. Data Eng.*, to be published.

[13] H. Peng, B. Du, H. Ma, M. Z. A. B. Bhuiyan, L. Jianwei, L. Wang, and P. S. Yu, "Spatial temporal incidence dynamic graph neural networks for traffic flow forecasting," *Inf. Sci.*, to be published.

[14] Y. He, J. Li, Y. Song, M. He, and H. Peng, "Time-evolving text classification with deep neural networks," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2241–2247.

[15] C.-C. Kao, J.-W. Chang, T.-I. Wang, Y.-M. Huang, and P.-S. Chiu, "Design and development of the sentence-based collocation recommender with error detection for academic writing," *J. Internet Technol.*, vol. 20, no. 1, pp. 229–236, 2019.

[16] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," 2016, *arXiv:1605.05101*. [Online]. Available: http://arxiv.org/abs/1605.05101

[17] T. Shen, T. Zhou, G. Long, J. Jiang, and C. Zhang, "Bi-directional block self-attention for fast and memory-efficient sequence modeling," 2018, *arXiv:1804.00857*. [Online]. Available: http://arxiv.org/abs/1804.00857

[18] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2048–2057.

[19] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 677–691, Apr. 2017.

[20] C. Dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in *Proc. 25th Int. Conf. Comput. Linguistics (COLING)*, 2014, pp. 69–78.

[21] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 675–678.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[23] M. H. Arif, J. Li, M. Iqbal, and H. Peng, "Optimizing XCSR for text classification," in *Proc. IEEE Symp. Service-Oriented System Eng. (SOSE)*, Apr. 2017, pp. 86–95.

[24] L. Dong and H. Zhao, "Hierarchical feature selection with orthogonal transfer," *J. Internet Technol.*, vol. 20, no. 4, pp. 1205–1212, 2019.

[25] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6338–6347.

[26] J. Ye, J. Ni, and Y. Yi, "Deep learning hierarchical representations for image steganalysis," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 11, pp. 2545–2557, Nov. 2017.

[27] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Unsupervised learning of hierarchical representations with convolutional deep belief networks," *Commun. ACM*, vol. 54, no. 10, p. 95, Oct. 2011.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[29] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*. [Online]. Available: http://arxiv.org/abs/1810.04805

[30] B. Chiu, G. Crichton, A. Korhonen, and S. Pyysalo, "How to train good word embeddings for biomedical NLP," in *Proc. 15th Workshop Biomed. Natural Lang. Process.*, 2016, pp. 166–174.

[31] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Feb. 2003.

[32] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *J. Mach. Learn. Res.*, vol. 5, pp. 361–397, Dec. 2004.

[33] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma, "Multi-label learning with millions of labels: Recommending advertiser bid phrases for Web pages," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 13–24.

[34] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," 2016, *arXiv:1606.01781*. [Online]. Available: http://arxiv.org/abs/1606.01781

[35] H. Chen, M. Sun, C. Tu, Y. Lin, and Z. Liu, "Neural sentiment classification with user and product attention," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 1650–1659.

[36] J. Wehrmann, R. Cerri, and R. Barros, "Hierarchical multi-label classification networks," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5225–5234.

[37] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2016, pp. 1480–1489.

[38] S. Gopal and Y. Yang, "Hierarchical Bayesian inference and recursive regularization for large-scale classification," *ACM Trans. Knowl. Discov. Data*, vol. 9, no. 3, pp. 1–23, Apr. 2015.

[39] Y. Prabhu and M. Varma, "Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 263–272.

[40] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain, "Sparse local embeddings for extreme multi-label classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 730–738.

[41] Y. Prabhu, A. Kag, S. Harsola, R. Agrawal, and M. Varma, "Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising," in *Proc. World Wide Web Conf.*, 2018, pp. 993–1002.

[42] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.

[43] Y. Liu, H. Peng, J. Li, Y. Song, and X. Li, "Event detection and evolution in multi-lingual social streams," *Frontiers Comput. Sci.*, pp. 1–23, 2019.

[44] B. Myroniv, C. Wu, Y. Ren, A. Christian, E. Bajo, and Y. C. Tseng, "Analyzing user emotions via physiology signals," *Data Sci. Pattern Recognit.*, vol. 1, no. 2, pp. 11–25, 2017.

[45] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: A statistical framework," *Int. J. Mach. Learn. Cyber.*, vol. 1, nos. 1–4, pp. 43–52, Dec. 2010.

[46] D. Filliat, "A visual bag of words method for interactive qualitative localization and mapping," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 3921–3926.

[47] J. Ramos, "Using TF-IDF to determine word relevance in document queries," in *Proc. 1st Instructional Conf. Mach. Learn.*, Piscataway, NJ, USA, vol. 242, 2003, pp. 133–142.

[48] A. Aizawa, "An information-theoretic perspective of TF–IDF measures," *Inf. Process. Manage.*, vol. 39, no. 1, pp. 45–65, Jan. 2003.

[49] B. Wu, C. Li, and B. Wang, "Event detection and evolution based on entity separation," in *Proc. 8th Int. Conf. Fuzzy Syst. Knowl. Discovery (FSKD)*, Jul. 2011, pp. 1–7.

[50] F. Rousseau, E. Kiagias, and M. Vazirgiannis, "Text categorization as a graph classification problem," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics, 7th Int. Joint Conf. Natural Language Process.*, vol. 1, 2015, pp. 1702–1712.

[51] H. Peng, J. Li, Q. Gong, S. Wang, Y. Ning, and P. S. Yu, "Graph convolutional neural networks via motif-based attention," 2018, *arXiv:1811.08270*. [Online]. Available: http://arxiv.org/abs/1811.08270

[52] Q. Mao, J. Li, S. Wang, Y. Zhang, H. Peng, M. He, and L. Wang, "Aspect-based sentiment classification with attentive neural turing machines," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 5139–5145.

[53] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2267–2273.

[54] S. Hochreiter and J. Schmidhuber, "LSTM can solve hard long time lag problems," in *Proc. Adv. Neural Inf. Process. Syst.*, 1997, pp. 473–479.

[55] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014, *arXiv:1406.1078*. [Online]. Available: http://arxiv.org/abs/1406.1078

[56] Y.-M. L. Ko-Wei Huang, C.-C. Lin, and Z.-X. Wu, "A deep learning and image recognition system for image recognition," *Data Sci. Pattern Recognit.*, vol. 3, no. 2, pp. 1–23, 2019.

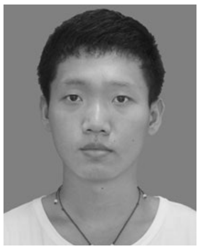
[57] H. Peng, J. Li, Y. Song, and Y. Liu, "Incrementally learning the hierarchical softmax function for neural language models," in *Proc. 31st AAAI Conf. Artif. Intell.* London, U.K.: AAAI Press, 2017, pp. 3267–3273.

[58] H. Peng, M. Bao, J. Li, M. Bhuiyan, Y. Liu, Y. He, and E. Yang, "Incremental term representation learning for social network analysis," *Future Gener. Comput. Syst.*, vol. 86, pp. 1503–1512, Sep. 2018.

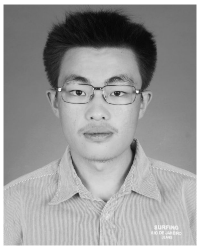
JIBING GONG received the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, China. He is currently a Professor with the School of Information Science and Engineering, Yanshan University. He is also the Head of the Knowledge Engineering Group (KEG) Research Team, Yanshan University. His main research interests include big data analytics, heterogeneous information networks, machine learning, and data fusion.

ZHIYONG TENG is currently pursuing the master's degree with the Department of Information Science and Engineering, Yanshan University. His researches focus on deep learning and NLP.

QI TENG is currently pursuing the B.E. degree with the State Key Laboratory of Software Development Environment, Beihang University. His research interests include social network mining and graph mining.



HEKAI ZHANG is currently pursuing the master's degree with the Department of Information Science and Engineering, Yanshan University. His main research interests include machine learning and graph neural networks.



LINFENG DU is currently pursuing the B.E. degree with the Department of Computer Science and Engineering, Beihang University (BUAA), Beijing, China.

SHUAI CHEN received the bachelor's degree in software engineering. He is currently pursuing the master's degree with the School of Information Science and Engineering, Yanshan University. His main research interests include deep learning and NLP.

MD ZAKIRUL ALAM BHUIYAN is currently an Assistant Professor with the Department of Computer and Information Sciences, Fordham University, New York, NY, USA. His researches focus on dependable cyber-physical systems, WSN applications, network security, urban computing, and sensor-cloud computing. He is also an Associate Editor of IEEE Access.

JIANHUA LI is currently an Associate Professor with the School of Information Science and Technology, Shijiazhuang Tiedao University, Shijiazhuang, China. His research interests include pattern recognition, multimedia processing, machine learning, and evolution computing.

MINGSHENG LIU is currently a Professor with the College of Electrical Engineering, Hebei University of Technology. His researches focus on network and information security, and information management and application.

HONGYUAN MA is currently a Senior Engineer with the National Computer Network Emergency Response Technical Team/Coordination Center of China. His current research interests include information retrieval, big data mining, and analytics.