This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2959991, IEEE Transactions on Knowledge and Data Engineering



# Hierarchical Taxonomy-Aware and Attentional Graph Capsule RCNNs for Large-Scale Multi-Label Text Classification

Hao Peng, Jianxin Li, *Member, IEEE*, Senzhang Wang, Lihong Wang, Qiran Gong, Renyu Yang, *Member, IEEE*, Bo Li, Lifang He, and Philip S. Yu, *Fellow, IEEE*

**Abstract**—CNNs, RNNs, GCNs, and CapsNets have shown significant insights in representation learning and are widely used in various text mining tasks such as large-scale multi-label text classification. Most existing deep models for multi-label text classification consider either the non-consecutive and long-distance semantics or the sequential semantics. However, how to coherently take them into account is still far from studied. In addition, most existing methods treat output labels as independent medoids, ignoring the hierarchical relationships among them, which leads to a substantial loss of useful semantic information. In this paper, we propose a novel hierarchical taxonomy-aware and attentional graph capsule recurrent CNNs framework for large-scale multi-label text classification. Specifically, we first propose to model each document as a word order preserved graph-of-words and normalize it as a corresponding word matrix representation preserving both the non-consecutive, long-distance and local sequential semantics. Then the word matrix is input to the proposed attentional graph capsule recurrent CNNs for effectively learning the semantic features. To leverage the hierarchical relations among the class labels, we propose a hierarchical taxonomy embedding method to learn their representations, and define a novel weighted margin loss by incorporating the label representation similarity. Extensive evaluations on three datasets show that our model significantly improves the performance of large-scale multi-label text classification by comparing with state-of-the-art approaches.

**Index Terms**—Multi-label classification, document modeling, graph rcnn, attention network, capsule network, taxonomy embedding

## 1 INTRODUCTION

As a fundamental text mining task, text classification aims to assign a text with one or several category labels such as topic labels and sentiment labels. Traditional approaches represent the text as sparse lexical features due to the simplicity and effectiveness [1]. For example, bag-of-words and n-gram are widely used to extract textual features, and then a general machine learning model such as Bayesian, logistic regression or SVM is utilized for text classification. Recent advances in deep learning techniques [2], [3] have enabled numerous variants of neural network based models from a large body of innovations, encompassing recurrent neural networks [4], [5], [6], [7], diversified convolutional neural networks [8], [9], [10], [11], [12], [13], capsule neural networks [14] and adversarial structures [15], [16]. These deep models have achieved inspiring performance gains on text classification due to their powerful capacity in representing the text as a fix-size feature map with rich semantics.

Recently, three popular deep learning architectures have attracted increasing research attention for representation learning of textual data, i.e., recurrent neural networks (RNNs) [6], [17], [7], [18], convolutional neural networks (CNNs) [8], [12], [10] and graph convolutional networks (GCNs) [11], [9]. Despite RNNs are suitable for capturing the semantics of short text [19], they are less effective to learn semantic features of long documents. Although the bi-directional block self-attention networks are proposed [7] to better model text or sentence, they consider documents as natural sequences of words, and ignore the long-distance semantic between paragraphs or sentences. CNNs and capsule networks simply evaluate the semantic composition of the consecutive words extracted with n-gram. However, n-gram may lose the long-distance semantic dependencies among the words [20]. Compared with RNNs and CNNs, GCNs can better capture the non-consecutive phrases and long-distance word dependency semantics [11], [9], but ignore the sequential semantic. To sum up, there is still a lack of a model that can simultaneously capture the non-consecutive, long-distance and sequential semantics of text. Meanwhile, as the text labels of some real-world text classification tasks are characterized by large hierarchies, there may exist strong dependencies among the class labels [21], [22], [23]. Exist-


- *Hao Peng, Jianxin Li, Qiran Gong and Bo Li are with Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing 100083, and also with the State Key Laboratory of Software Development Environment, Beihang University, Beijing 100083, China, China. E-mail:{penghao, lijx, libo}@act.buaa.edu.cn, allen_gong@buaa.edu.cn.*
- *Lihong Wang is with the National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China. E-mail: wlh@isc.org.cn.*
- *Senzhang Wang is with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China. E-mail: szwang@nuaa.edu.cn.*
- *Renyu Yang is with the School of Computing, University of Leeds, Leeds LS2 9JT, UK. E-mail: r.yang1@leeds.ac.uk.*
- *Lifang He is with the Department of Biostatistics and Epidemiology, University of Pennsylvania, Philadelphia, PA 19104, USA. E-mail: lfhe@pennmedicine.upenn.edu.*
- *Philip S. Yu is with the Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607, USA. E-mail: psyu@uic.edu.*


*Manuscript received May 6, 2019. (Corresponding author: Jianxin Li.)*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2959991, IEEE Transactions on Knowledge and Data Engineering



ing deep learning models cannot effectively and efficiently leverage the hierarchical dependencies among labels for improving the classification performance, either.

It is non-trivial to obtain a desirable performance for large-scale multi-label text classification due to the following challenges: First, although there are many methods for document modeling, how to represent a document by fully preserving its rich and complex semantic information still remains an open problem [24]. It is challenging to come up with a document modeling method that can fully capture the semantics of a document, including the non-consecutive, long-distance and sequential features. Second, limited by different document modeling methods, existing CNNs, RNNs and GCNs models can only capture partial semantic feature. It is imperative to design a deep learning model that can simultaneously capture multiple types of textual features mentioned above. Third, although some recursive regularization based hierarchical text classification models [25], [26], [11], [27] consider the pair-wise relationship between labels, they fail to consider their hierarchical relationships. The computation of the above regularized models is also expensive due to the use of Euclidean constraints. Therefore, how to make full use of the hierarchical label-dependencies among labels to improve the classification accuracy and reduce the computational complexity becomes extremely challenging.

To address these challenges, we propose HE-AGCRCNN, a novel Hierarchical taxonomy-awarE and Attentional Graph Capsule Recurrent CNNs framework, for large-scale multi-label text classification. Specifically, our framework consists of three major parts: word order preserved graph-of-words for document modeling, attentional capsule recurrent CNNs for feature learning, and hierarchical taxonomy-aware weighted margin loss for large-scale multi-label text classification:

**Word Order Preserved Graph-of-Words for Document Modeling.** We regard each unique word as a vertex, the word co-occurrence relationships within a sliding window as edges, and the positional index of a word appearing in the document as its attribute. We can therefore build a word order preserved graph-of-words to represent a document. Then we select top $w$ central words from the graph-of-words based on the *closeness centrality* feature, and construct a sub-graph for each central word from neighbors by breadth first search (BFS) and depth first search (DFS). To preserve local sequential, non-consecutive and long-distance semantics, we normalize each subgraph to blocks of word sequences that retain local word order information, and construct an arranged word matrix for the $w$ sub-graphs. To incorporate more semantic information, we use a pre-trained word embedding vectors based on word2vec [28], [29] as word representation in the arranged word matrix. Finally, each document is represented as a corresponding 3-D tensor representation whose three dimensions denote the selected central words, the ordered neighbor words sequence and the embedding vector of each word, respectively.

**Attentional Capsule Recurrent Convolutional Neural Networks.** An attentional capsule recurrent CNN (RCNN) model is designed to make use of the 3-D tensor as input for document feature learning. The proposal model first uses two attentional RCNN layers to learn different levels of text features with both non-consecutive, long-distance and local sequential semantics. Here, we not only guarantee the independence of the feature representation between sub-graphs, but also model different impacts among different blocks of word sequences. When the convolution kernel slides horizontally along the combining long-distance and local sequential ordering of words, the attentional LSTM unit is employed to encode the output of the previous step of CNN, and the output of current step of attentional LSTM to produce the final output feature map in the RCNNs layer. Subsequently a capsule network layer is used to implement an iterative routing process to learn the intrinsic spatial relationship between text features from lower to higher levels for each sub-graph. In the final DigitCaps layer, the activity vector of each capsule indicates the presence of an instance of each class and is used to calculate the classification loss.

**Hierarchical Taxonomy-Aware Weighted Margin Loss.** Considering the hierarchical taxonomy of the labels, we design two types of meta-paths, and leverage them to conduct random walk on the hierarchical taxonomy network to generate label sequences. Therefore, the hierarchical taxonomy relationship among the labels can be encoded in a continuous vector space with the skip-gram [29] on the sequences. In this way, the distance between two labels can be measured by calculating the cosine similarity of their vectors. By taking the distance between labels into consideration, we design a new weighted margin loss to guide the training of the proposed attentional capsule RCNN models.

We conduct extensive evaluations on the proposed framework by comparing against state-of-the-art methods on three benchmark datasets, traditional shallow models and recent deep learning models. The results show that our approach outperforms them by a large margin in both efficiency and effectiveness on large-scale multi-label text classification. The code of this work is publicly available at *https://github.com/RingBDStack/HE-AGCRCNN*.

The contributions of this paper are summarized below.

- A novel hierarchical taxonomy-aware and attentional graph capsule recurrent CNNs framework is proposed for large-scale multi-label text classification.
- A new word order preserved graph-of-words method is proposed to better model document and more effectively extract textual features. The new method preserves both non-consecutive, long-distance and local sequential semantics.
- A new word sequence block level attention recurrent neural network is proposed to better learn local sequential semantics of text.
- A novel hierarchical taxonomy-aware weighted margin loss is proposed to better measure the distance of classes and guide training of the proposed models.
- Extensive evaluations on three datasets demonstrate the efficiency and effectiveness of the proposal.

The rest of the paper is organized as follows. We first review related work in Section 2. We introduce the word order preserved document modeling in Section 3, and present the model architecture in Sections 4 and 5. The evaluation is conducted in Section 6. Finally, conclusion and future work are given in Section 7.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2959991, IEEE Transactions on Knowledge and Data Engineering



## 2 Related Work

As our work is closely related to traditional text classification models, multi-label text classification, traditional deep learning models and graph convolution networks for text classification, the related works are four-fold.

Tradition text classification models use feature engineering and feature selection to obtain features for text classification [1]. For example, Latent Dirichlet Allocation [30] has been widely used to extract topics from corpus, and then represent documents in the topic space. It performs better than bag-of-word (BOW) when the feature numbers are small. However, when the size of words in vocabulary increases, it does not show advantage over BOW on text classification [30]. In addition to statistical characteristics based TF-IDF, LDA, BMT, etc. [1], semantic role labels have also been proven to be used to enhance text representation [31]. There are also some existing work that tried to convert texts to graph-of-words [20]. Similar to our works, they used word co-occurrence to construct graphs from texts, and then they applied similarity measure on graph to define new document similarity and features for text [20].

Multi-label text classification is the problem of assigning each document a set of target labels, and is also an application scenario of multi-label learning [32]. Multi-label learning algorithm often needs problem transformation or algorithm adaptation from multi-class learning models. For instance, one-versus-rest binary support vector machines (BSVM), one-versus-rest binary logistic regression (BLR) and one-versus-rest binary multinomial naive bayes (BMNB) [33], [34] are typically transformed or adapted multi-label learning models. In our hierarchical large-scale multi-label text classification scenario, many efforts [26], [25], [11], [27], [35] have been put on leveraging the pairwise relation between labels as recursive regularization to improve the classification results.

For deep learning models, there have been RNNs, CNNs, and capsule models applied to text classification. For example, hierarchical RNN has been proposed for long document classification [17] and later attention model is also introduced to emphasize important sentences and words [6]. Similar to RNNs, the recently proposed self-attention based sentence embedding technologies [36], [18], [7] have shown effectively capturing both long-range and local dependencies in sentiment-level tasks. For example, Bi-BloSAN [7] is a bi-directional block self-attention network to learn text representation and models text as sequences. Different from the previous attention networks [6], [7], our attention model focuses on the different importance of different blocks of word sequences. For CNNs models, Kalchbrenner et al. [37] and Kim et al. [8] used simpler CNN for text classification, and showed significant improvements over traditional texts classification methods. Zhang et al. [38] and Conneau et al. [12] used a character level CNN with very deep architecture to compete with traditional BOW or n-gram models. The combination of CNNs and RNNs are also developed which shows improvements over topical and sentiment classification problems [39]. Capsule networks were proposed by Hinton et. [40], [41], [42] as a kind of supervised representation learning methods, in which groups of neurons are called capsules. Capsule network has been proved effective in learning the intrinsic spatial relationship between features [14], [43], [44]. [14] showed that Capsule networks can help to improve low-data and label transfer learning. However, as mentioned in the introduction, existing textual deep learning models are not compatible with diverse text semantic coherently learning. Compared with our work, these previous studies only considered N-gram or sequential text modeling, but ignored high level of non-consecutive and long-distance semantics of text.

GCN derived from graph signal processing [45], [46], and the graph convolution operation has been recognized as the problem of learning filter parameters that were replaced by a self-loop graph adjacency matrix, updating network weights, and extended by utilizing fast localized spectral filers and efficient pooling operations in [47], [48], [49]. With the development of GCN technologies, graphs embedding approaches, such as PSCN [50] and GCAPS-CNN [51], have been developed in graph classification tasks. Recently, the recursively regularized deep graph cnn [11] has been proposed to combine graph-of-words representation, graph CNN, and hierarchical label dependency for large-scale text classification. Then the Text GCN model [9] has been proposed to capture global word co-occurrence information and perform text classification without word embeddings or other external knowledge. Although long-distance and non-continuous text features are fully considered in the two graph convolution models [11], [9], they ignore the continuous and sequential semantics of words in the text when converting text to graph structures. Different from existing graph-based text classification models [20], [11], [9], the arranged word matrix representation was first proposed in our work. Unsupervised network representation learning technologies [52] provide an effective way to measure the distance between labels, which is different from the original method of measuring the distance between labels according to the edge relation in hierarchical taxonomy. In addition, the recursive regularization is usually time consuming due to the Euclidean constraint.

## 3 Word order Preserved graph-of-words for Document Modeling

In this section, we introduce how we model a document as a word order preserved graph-of-words, and how to extract central words and sub-graphs from it. Formally, let $\mathcal{X}$ denote the instance space of text and $\mathcal{Y}$ denote the label space. The task of multi-label text classification is to learn a mapping function $f : \mathcal{X} \rightarrow \mathcal{Y}$ from the training set $\{(x_i, Y_i)|1 \leq i \leq c\}$. Here, $x_i \in \mathcal{X}$ is an instance of text and $Y_i \subseteq \mathcal{Y}$ is the set of labels associated with $x_i$. For any unseen instance $x \in \mathcal{X}$, the multi-label classifier $f(\cdot)$ predicts $f(x) \subset \mathcal{Y}$ as the set of proper labels for $x$.

### 3.1 Word Order Preserved Graph-of-Words

In order to preserve more semantic information of text, we model a document as a word order preserved graph-of-words. We regard each unique word as a vertex, the word co-occurrence relationships within a sliding window as edges, and the positional index appearing in the document as its attribute, as shown in the step 1 of Figure 1.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2959991, IEEE Transactions on Knowledge and Data Engineering



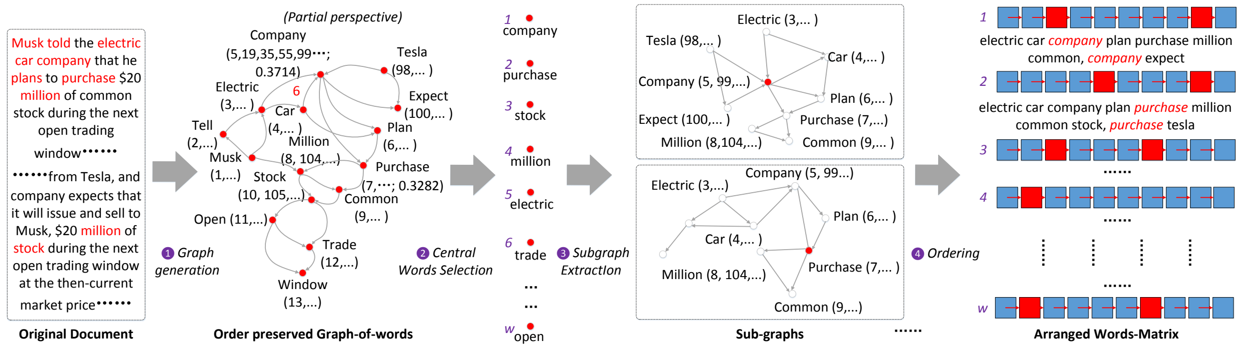



Fig. 1. Illustration of converting a document to an arranged word matrix representation. We first construct a word order preserved graph-of-words, and then a top $w$ nodes (words) sequence is selected from the ranking of each node's *closeness centrality* feature. For each node (word) in the sequence, a corresponding sub-graph is extracted and normalized as a sequence of words that retain local word order information.

We first split a document into a set of sentences and extract tokens using the Stanford CoreNLP tool. We also employ a lemmatization of each token using the Stanford CoreNLP, and remove the stop words. Then we construct an edge between two word nodes if they co-occur in a pre-defined fixed-size sliding window, and the edge weight is the times of their co-occurrence. Meanwhile, we record all the positional indexes where a word appears in the document as its attribute. For example, for the first sentence "*Musk told the electric car...*" shown in the document in Figure 1, we perform lemmatization on the second word "*told*" to get "*tell*" with attribute "2", and build a directed edge from "*Musk*" to each of the words in the sliding window. As shown in the graph-of-words of Figure 1, the word "*Company*" appears at the 5-th, 19-th, 35-th, etc. positions, respectively. Note that the graph-of-words is a weighted and directed graph with the positional indexes as the node attributes. For example, in the graph-of-words of Figure 1, the weight of the edge between nodes "*Company*" and "*Car*" is 6 meaning that "*Company*" and "*Car*" has a total of 6 co-occurrences in the document when sliding the window.

### 3.2 Arranged Word Matrix Generation

We denote the word order preserved graph-of-words as $\mathcal{G} = (V, E, W, A)$, where $V$ denotes the node set and $|V| = n$, $E$ denotes the edge set and $|E| = m$, $W$ denotes the weights of the edges and $A$ denotes the attributes of the nodes. Based on the assumption in [53], [54], the most relevant keywords about the content of the document tend to have high centrality in the graph-of-words. Consider the global and higher-order feature, we extract the top $w$ central words from $\mathcal{G}$ based on node's *closeness centrality* feature. It' also because compared with other centrality measures of linear computational complexity, such as degree centrality, betweenness centrality, the nodes selected by closeness centrality are more influential at node reachability perspective [55]. Here, in order to calculate *closeness centrality* for each node, we use $d(v, u)$ to denote the shortest-path distance between nodes $v$ and $u$ by using the Dijkstra algorithm. For each node $v$, its *closeness centrality* can be calculated by $C_v = (n-1)/\sum_{u \in V, u \neq v} d(v, u)$, and we arrange the nodes in order of largest to smallest. The larger the *closeness centrality*, more important the node is in the graph. As shown in the graph-of-words of Figure 1, the *closeness centrality* of word "*Company*" is the highest $0.3714$ among the words' in the graph. Then we select the top $w$ central nodes from the 1-th node "*Company*" to the $w$-th node "*Open*", as shown in the step 2 of Figure 1. Next, we introduce how to extract sub-graph.

First, we extract the nodes and edges from the neighborhood of each central node in the order of breadth first search (BFS) and depth first search (DFS) to build a subgraph. To be more specific, we use the BFS strategy to extract directly connected edges and nodes as first-order neighborhood. And then, we select the node with the highest *closeness centrality* from the first-order neighborhood to extract the connected edges and nodes with the DFS strategy. For example, as shown in the first subgraph of the step 3 in Figure 1, we extract the nodes of "Car", "Electric", "Tesla" , "Expect", "Purchase", and "Plan" connected to "Company" and their connected edges through the BFS strategy. And we start from the node "Purchase" and extract the nodes "Million" and "common" and their connected edges according to the DFS strategy. Meanwhile, we limit the number of nodes in the subgraph to be no more than $g$. In this way, the subgraph $\mathcal{G}(v)$ contains both the non-consecutive, long-distance and local sequential information of the central word $v$ in the document. To further save the above information of a subgraph, we order the words in the sub-graph $\mathcal{G}(v)$ by their nodes (words) attributes. For the most case where a subgraph contains multiple sentences, we guarantee that the long sentences are in the front and the short sentences are in the back. As shown in the first line of the arranged word matrix in Figure 1, we convert the first sub-graph $\mathcal{G}(v)$ as sequences liking"*electric car company plan purchase million common* and *company expect* ". As a result, we normalize each subgraph as sequences of nodes (words) that keep the same length $h$. If the number of words in the sequence is less than $h$, it is padded with zeros. Finally, we concatenate all the normalized sequences of the $w$ central words into an arranged word matrix, as shown in step 4 of Figure 1. The red nodes represent central words.

### 3.3 Unified Representation of the Documents

For better representing the original words in the word matrix, we use word2vec [29], [28] to incorporate as much word semantic information as possible. Specifically, word2vec is trained on a larger corpus, i.e., Wikipedia. All parameters for word2vec are set to be default values. In this way, we have a 3-D tensor representation for each document, where the padded vectors are zero vectors with the same dimension. Then the convolution, recurrent and capsule

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2959991, IEEE Transactions on Knowledge and Data Engineering



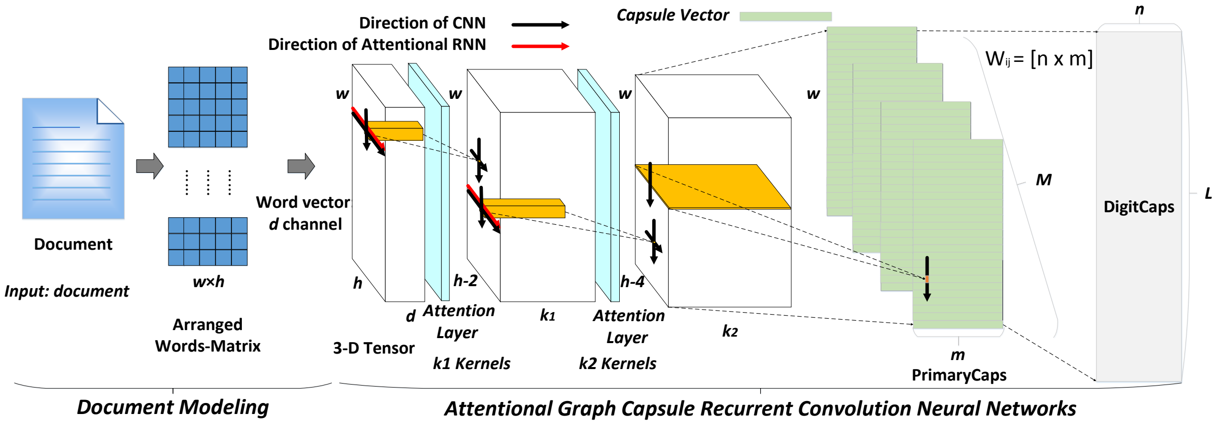



Fig. 2. Architecture of the proposed hierarchical taxonomy-aware and attentional graph capsule recurrent convolution neural network. It consists of document modeling, attentional capsule recurrent CNN, and hierarchical taxonomy-aware weighted margin loss. The network input is the original document. The length of the activity vector of each capsule in DigitCaps layer indicates presence of an instance of each class.

networks introduced in the next section will be operated over the unified representations of the documents.

## 4 Attentional Capsule Recurrent CNN

In this section, we introduce the proposed attentional capsule recurrent CNN model. Technically, the proposed models belong to graph embedding. After converting each document into a 3-D tensor representation, we design three layers of attention network to learn both the non-consecutive, long-distance and local sequential feature. From the input document to the output labels, the architecture is shown in Figure 2. Specifically, the three layers of neural networks contain two major parts: two layers of attentional recurrent convolution neural networks and one layer of capsule network for rich feature learning. Note that this is a general framework and the number of attentional recurrent convolution layers can be adjusted based on specific dataset for classification, and the parameter configuration of self-attentional recurrent operators and capsule networks can be customized in different text classification tasks.

### 4.1 Attentional Recurrent CNN

Different from the architecture of existing recurrent convolutional neural networks [39], which encode sentences or document as a dense vector for classification, our proposed attention network encodes whole document as 3-D feature map. The attention network takes the $w \times h \times d$ size of 3-D tensor extracted from the document and word embedding as the input, where $w$ is the number of central words, $h$ is the length of normalized sequence of words and $d$ is the dimension of word embedding, as shown in Figure 2. The output of the two layers of the attention network is the other 3-D feature map as input of the proposed capsule network.

In the first layer, the convolution operator filters the input tensor with $k_1$ kernels of size $1 \times 3 \times d$ with a horizontal stride of 2 elements and a vertical stride of 1 element, which is illustrated with the black convolution slide direction arrow in Figure 2. We use *ReLU* as the activation function to speed up the training process and avoid over-fitting. Here, convolution kernel serves as a composition of the semantics in the receptive field to extract the higher level semantic features. Meanwhile, we employ a masked soft

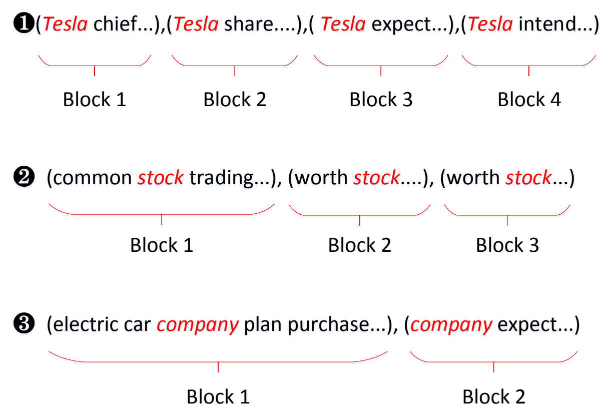



Fig. 3. Illustration of blocks of word sequences. The red word refers to the central word for each subgraph. The words in any block are the contexts of the central word at different locations in a document.

attentional recurrent neural operator to capture the local sequential semantic for each sub-graph $\mathcal{G}(v)$. The attentional recurrent neural operator acts on each horizontal words sequence, which is illustrated with the red recurrent slide direction arrow in Figure 2. However, as we known, we convert the subgraph $\mathcal{G}(v)$ into blocks of word sequences according to the properties of the nodes. We give three different blocks, as shown in Figure 3, to illustrate the blocks of word sequences, which consist of each line of the arranged word matrix. There are blocks of word sequences depending on the different contexts of central words. In order to measure the different impacts of different number of blocks on the local sequential semantic learning, we customize the masked attentional parameter shared in long short-term memory units to learn the rich local sequential semantic for each sub-graph $\mathcal{G}(v)$. We name the module as Attention-LSTM.

Since our proposed framework needs to learn feature for multiple documents, the attention-LSTM module guarantees that any subgraphs with the same order and the same number of blocks share the same attention parameters. For example, for the $t$-th subgraph $\mathcal{G}(u)$ from the document $D_i$, assuming that it contains the $q$ blocks of word sequences, the parameter of the masked attention module is the $\alpha_{B_{t,q,1}}, \alpha_{B_{t,q,2}}, \ldots, \alpha_{B_{t,q,q}}$. Similar, for the $t$-th subgraph $\mathcal{G}(v)$ from the document $D_j$, assuming that it contains the $p$ blocks of word sequences, the parameter of the masked attention module is the $\alpha_{B_{t,p,1}}, \alpha_{B_{t,p,2}}, \ldots, \alpha_{B_{t,p,p}}$. If $p$ and $q$

are equal, subgraphs $\mathcal{G}(u)$ and $\mathcal{G}(v)$ share the same attention parameters $\alpha_B$, otherwise not. But, among any one block, each word shares the same attention parameter. For example, in Figure 3, we assume they are converted from the top 3 subgraphs. In the 1-th block of the 3-rd subgraph, because there are 2 blocks, the words *electric*, *car*, *company*, *plan*, and *purchase* share the same masked attention parameter $\alpha_{B_{3,2,1}}$. Then, after the first attentional recurrent convolution layer, there is an $w \times (h-2) \times k_1$ size of feature map. Compared with traditional convolution and recurrent networks on text data [8], [12], [10], [11], [39], [7], the significant difference of our designed attentional recurrent CNN units is that it can integrate the long-distance, non-consecutive and local sequential semantics of the corresponding sub-graph $\mathcal{G}(v)$.

The second attentional recurrent convolution layer takes the output of the first attentional recurrent convolution as its input, and filters it with $k_2$ kernels of size $1 \times 3 \times k_1$ with a horizontal stride of 2 elements and a vertical stride of 1 element, which are illustrated with the black convolution slide direction arrow and red recurrent slide direction arrow in Figure 2. We still guarantee that each horizontal feature map characterizes the semantics of corresponding sub-graph $\mathcal{G}(v)$, and the attentional recurrent and convolution operators between different sub-graphs are independent. In the second layer, for each sub-graph $\mathcal{G}(v)$, the number of attentional parameter is same with the first layer, but they are separated in training. After the second attentional recurrent convolution layer's operation, a $w \times (h-4) \times k_2$ size of feature map is generated.

More formally, we give the definitions of convolution operator and Attentional-LSTM unit, respectively. The convolution operator can be defined as

$$x_j^l = f(\sum_{i \in M_j} x_i^{l-1} \cdot k_{ij}^l + b_j^l), \tag{1}$$

where $x_j^l$ represents the $j$-th feature map of the $l$-th layer of the convolution network, and $l \in \{1, 2\}$. This formula shows the convolution operation and the summation for all the associated feature maps $x_i^{l-l}$ and the $j$-th convolution kernel $k_{ij}^l$ of layer $l$, and then add an offset parameter $b_j^l$. Finally, a *ReLU* activation function $f$ is applied. Meanwhile, the Attentional-LSTM unit can be defined as following:

$$\begin{aligned} f_t =& \sigma_g(W_f \alpha_B x_t + U_f c_{t-1} + b_f), \\ i_t =& \sigma_g(W_i \alpha_B x_t + U_i c_{t-1} + b_i), \\ o_t =& \sigma_g(W_o \alpha_B x_t + U_o c_{t-1} + b_o), \\ c_t =& f_t c_{t-1} + i_t \sigma_c(W_c \alpha_B x_t + b_c), \\ h_t =& o_t \sigma_h(c_t), \end{aligned} \tag{2}$$

where $t$ refers to the index of the horizontal convolution sequence, $f_t$ refers to the forgotten gate, $i_t$ refers to the input gate, $o_t$ refers to the output gate, $c_t$ is the cell state, and $\alpha_B$ refers to the attentional parameter. Since the output of the convolution network is input to the LSTM, and the output of the LSTM is the feature map, $x_t = x_j^l$ and $x_{t+1} = x_{j+1}^l$.

### 4.2 Capsule Network with Dynamic Routing

Since the capsule network can effectively learn some aspect features of textual representation [43], the output of the two layers of attentional recurrent convolution networks is $w \times (h-4) \times k_2$ size of feature map and is input to the next capsule networks with dynamic routing layer. In order to independently learn the features of each subgraph into the corresponding capsule vectors, different from existing textual capsule networks [14], our proposed capsule networks guarantee the independence of feature between sub-graphs, as shown in Figure 2.

The capsules contain groups of locally invariant neurons that learn to recognize the presence of features and encode their properties into vector outputs, with the vector length representing the presence of the features. The primary capsule layer is a convolution capsule layer with $M$ channels of capsules, as shown in Figure 2. Each primary capsule contains $m$ convolution units with a $\frac{h+12}{9} \times k_2$ size of kernel and a vertical stride of 1, and can be seen as the output of all $w \times \frac{h+12}{9} \times k_2$ convolution units. Here, we guarantee the independence of the representation of sub-graph $\mathcal{G}(v)$. In total, the primary capsules have $w \times M$ capsule outputs, and each output is a $m$-dimensional vector, as shown in Figure 2. We can see all the primary capsules as a convolution layer with Eq. 3 as its block non-linearity.

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \cdot \frac{s_j}{\|s_j\|}, \tag{3}$$

where $v_j$ is the output of capsule $j$, and $s_j$ is its total input. For all but the first layer of capsules, the total input to a capsule $j$ is a weighted sum over all the prediction vectors $\hat{u}_{j|i}$ from the capsules in the layer below, and is calculated by multiplying the output $u_i$ of a capsule in the layer below by a weight matrix $W_{ij}$ as following:

$$s_j = \sum_i c_{ij} \hat{u}_{j|i}, \qquad \hat{u}_{j|i} = W_{ij} u_i, \tag{4}$$

where $c_{ij}$ is the coupling coefficient that is determined by the iterative dynamic routing process. The coupling coefficients between capsule $i$ and all the other capsules in the layer above sum to 1. They are determined by a *routing softmax* whose initial logits $b_{ij}$ are the log prior probabilities that capsule $i$ should be coupled to capsule $j$. The $c_{ij}$ can be calculated as following:

$$c_{ij} = \frac{\exp(b_{ik})}{\sum_k \exp(b_{ik})}. \tag{5}$$

The final DigitCaps layer has $n$ capsules per digit class and each of these capsules receives input from all the other capsules in the layer below. $W_{ij}$ is a weight matrix between each $u_i, i \in (1, M \times w)$ in primary capsules and $v_j$, $j \in (1, L)$, where $L$ refers to the number of classes.

As the length of the capsule's output vector represents the presence of a class, the length $\|v_k\|$ of each capsule in the final layer can then be viewed as the probability of the text belonging to a particular class $k$. The length of the activity vector of each capsule in DigitCaps layer is used to calculate the classification loss. This encourages the network to learn a more general representation of text with classification task. Different from the capsule networks [41], [42] applied in the field of computation vision, our capsule network abandons the use of activity vector to reconstruct the original input data during training. We consider that the distance between the raw text and the output of the reconstructed representation are relatively large in the word

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2959991, IEEE Transactions on Knowledge and Data Engineering



embedding space, and the reconstruction loss will be more uncertainty. Next, we introduce how to design a weighted margin loss to measure distance of classes in hierarchy and guide the training of Attentional Capsule Recurrent CNN.

## 5 Hierarchical Taxonomy-aware Weighted Margin Loss

Intuitively, the distances between any two classes on the hierarchy are different, but popular margin loss in capsule network [41] and other distance measures in multi-label learning [32] between classes did not consider the hierarchical relations among labels. So, we explore a hierarchical taxonomy-aware weighted margin loss to guide the training of the proposed model in Section 4. Note that the following proposed weighted margin loss is not suitable for multi-label learning tasks without dependencies between labels.

More formally, we denote the hierarchical taxonomy structure of the labels as $\mathcal{HG} = \{\mathcal{V}, \mathcal{E}\}$, where vertices $\mathcal{V}$ are classes $\mathcal{S}$ and the directed edges $\mathcal{E}$ represent the hierarchical parent-child relationship among the labels. In large-scale multi-label text classification, for a document $D_s$ and corresponding positive labels set $T_s$, the number of labels in $T_s$ is usually much smaller than the number of remaining negative ones in $S$, leading to a large loss of the objective function. In fact, in hierarchical label network, the closer the edge relationship between nodes is, the shorter the semantic distance between labels will be. In order to conveniently capture the relationship between labels on the hierarchical label structure, we design two meta-paths to guide random walk on the label structure, and generate label sequences to learn label representation.

Figure 4 illustrates the hierarchical taxonomy structure of the labels, where each node refers to a label/class, and each directed edge represents a parent-child relationship. Note that the taxonomy network is not a strict hierarchical structure, and may contain cycles. For example, the two hierarchical relations of "*Economic*"-"*International trade*"-"*Arms sales*" and "*Economic*"-"*Defense economy*"-"*Arms sales*" can form a cycle. Both hierarchical structure of taxonomy and graph structure of taxonomy are common in practice, and can be modeled as a hierarchical graph-of-labels. For conveniently calculating the distance between any two labels, we measure the cosine similarity through their representation vectors. As shown in Figure 4, we extract the following two types of meta-paths from the hierarchical graph-of-labels, "*Child1 - Father0 - Child2*" and "*Father1 - Child0 - Father2*". Actually, the two types of meta-paths control the directions of the random walk to the upper and lower layers, respectively. So, we perform meta-paths guided random walk to generate sequences of labels. Here, we set that the probability of selecting two meta-paths is equal during random walking. Similar to metapath2vec [56] and Deepwalk [57], we also use the skip-gram negative sampling [28], [29] to encode the relations among the labels into a continuous vector space. We optimize the following objective function, which maximizes the log-probability of observing a network neighborhood $N_S(l)$ for a node $l$ conditioned on its feature representation, given by $g$ :

$$\max_{g} \sum_{l \in V} [-\log Z_l + \sum_{n_i \in N_S(l)} f(n_i) \cdot f(l)], \tag{6}$$

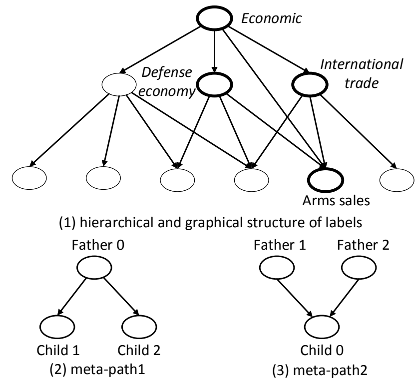



Fig. 4. Illustrations of the hierarchical taxonomy and the two meta-paths.

where $g(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function. Since it will be time consuming to compute $Z_l = \sum_{nl \in V} \exp(g(l) \cdot g(nl))$ for large network, we approximate it using the negative sampling technology. We optimize the Eq. 6 by using stochastic gradient method.

Thus, given a tag label/class $l \in \mathcal{V}$, we can approximate the semantic distance between any other label $l_i, i \in [1, L]$ by calculating their cosine similarity from their embedding vectors as following:

$$d(l, l_i) = 1 - \cos(vec(l), vec(l_i)). \tag{7}$$

Next, in order to take advantage of dependencies among labels to guide the training of the proposed attentional capsule recurrent CNN model, we design a hierarchical taxonomy-aware weighted margin loss objective function:

$$\begin{aligned} \mathcal{L} = \sum_{k=1}^{L} [T_k \max(0, m^+ - \|v_k\|)^2 \\ + \lambda \cdot p \cdot \alpha_k \cdot (1 - T_k) \cdot \max(0, \|v_k\| - m^-)^2], \end{aligned} \tag{8}$$

where $T_k = 1$ if and only if a digit of class $k$ is present, and $m^+$ and $m^-$ are the given thresholds for the upper and lower bounds. The $\lambda$ down-weighting of the loss for absent digit classes stops the initial learning from shrinking the lengths of the activity vectors of all the digit capsules, such as 0.5 in the original capsule networks [41], [42]. $\alpha_k \in [0, 1]$ is the minimum semantic distance from negative label $k$ to the positive labels set in the hierarchical label network. The total loss is the sum of the losses of all the digit capsules. Formally, for a document $D_s$, the positive label set is $T_s \subset \mathcal{S}$. And for any negative label $k$, the minimum semantic distance $\alpha_k$ is:

$$\alpha_k = 1 - \max_{t \in T_s}(\cos(vec(t), vec(k))). \tag{9}$$

Meanwhile, to make an unbiased and smooth overall objective function after integrating the semantic distance, we add an adjustment factor $p$ that satisfies $\sum_{k=1}^{L} p \cdot \alpha_k = 1$. We can approximate the distribution of the semantic distance $\alpha_k$ by $1 - e^{-x}, x \in [1, L]$ to obtain an approximation of the adjustment factor $p$ for different datasets.

## 6 Experiments

In this section, we conduct experiments to evaluate the performance of the proposed framework. We will first introduce the used datasets, the evaluation metrics, methods for

TABLE 1
Statistics of the three datasets. Training, Validation , Testing, and Class-Labels denote the total number of training, testing samples and labels, respectively. Words/Sample is the average number of words per sample. Labels/Sample is the average number of labels per sample, and Sample/Labels is the average number of documents per label.

| Datasets | Training | Validation | Testing | Class-Labels | Depth | Words/Sample | Labels/Sample | Samples/Label |
|---|---|---|---|---|---|---|---|---|
| RCV1 | 23,149 | - | 784,446 | 103 | 6 | 268.95 | 3.24 | 729.67 |
| EUR-Lex | 15,449 | - | 3,865 | 3,956 | 4 | 1229.77 | 5.32 | 15.59 |
| Reuters-21578 | 5800 | 600 | 300 | 10 | - | 257.32 | - | - |

comparison, and experimental settings. Then, we will compare our methods with baselines, and provide the analysis and discussions on the results.

### 6.1 Datasets

Both RCV1 and EUR-Lex are the classic and widely used multi-label datasets [58], [59], [10], [60] with unbalanced label distribution. We use two datasets including RCV1 and EUR-Lex for large-scale multi-label text classification. To make the baselines as competitive as possible, we use the same instance-based thresholding strategy as proposed and used in [61], [59], [25] to divide the RCV1 and EUR-Lex datasets. We use the Reuters-21578 [62] to evaluate the effectiveness of the proposed capsule network in transferring from single-label to multi-label classification [14]. The statistics of the datasets is shown in Table 1.

• **Reuters Corpus Volume I (RCV1)** [58] is a manually labeled newswire collection of Reuters News from 1996-1997. It consists of over 800,000 manually categorized newswire stories by Reuters Ltd for research purposes. Multiple topics can be assigned to each newswire story and there are 103 topics in total. The news documents are categorized with respect to three controlled vocabularies: industries, topics and regions. The relations among the labels are typically graphic structure with self-loops. We use the topic-based hierarchical classification because it has been widely adopted in evaluation.

• **EUR-Lex** [60] is a collection of documents about European Union law. It contains many different types of documents, including treaties, legislation, case-law and legislative proposals, which are indexed according to several orthogonal categorization schemes to allow for multiple search facilities. The most important categorization is provided by the EUROVOC descriptors, which forms a topic hierarchy with almost 4000 categories regarding different aspects of European law. Directory code classes are organized in a hierarchy of 4 levels with a typical tree structure. Since the dataset contains several European languages, we choose English version of documents.

• **Reuters-21578** [62], [14] is a collection appeared on the Reuters newswire in 1987. We follow [14] to choose 6,700 documents from the Reuters financial newswire service, where each document contains either multiple labels or a single label. And we also focus 10 popular topics, and reprocess the corpus to evaluate the capability of capsule networks of transferring from single-label to multi-label text classification. For validation and training, we only use the single-label documents in the validation and training sets. For testing, we only uses the multi-label documents in testing dataset. Note that this dataset is only for testing the advantages of the transferring from single-label to multi-label classification task of our capsule network that incorporates multiple semantics.

### 6.2 Evaluation Metrics and Baselines

Consider the imbalance between classes, we use the standard label-based evaluation metrics [32], [59], [25], [63] to measure the performance of all the methods.

• **Micro-**$F_1$ is a metric considering the overall precision and recall of all the labels. Let $TP_t$, $FP_t$, $FN_t$ denote the true-positives, false-positives and false-negatives for the $t$-$th$ label in label set $\mathcal{S}$ respectively. The $Micro$-$F_1$ is defined as:

$$Micro\text{-}F_1 = \frac{2PR}{P+R}, \tag{10}$$

where:

$$Precision(P) = \frac{\sum_{t\in\mathcal{S}} TP_t}{\sum_{t\in\mathcal{S}} TP_t + FP_t},$$
$$Recall(R) = \frac{\sum_{t\in\mathcal{S}} TP_t}{\sum_{t\in\mathcal{S}} TP_t + FN_t}.$$

• **Macro-**$F_1$ is a metric which evaluates the averaged $F_1$ of all the different class-labels. Different from Micro-F1 that gives equal weight to all the instances, $Macro$-$F_1$ gives equal weight to each label in the averaging process. Formally, $Macro$-$F_1$ is defined as:

$$\begin{aligned} Macro - F_1 =& \frac{1}{|\mathcal{S}|}\sum_{t\in\mathcal{S}} \frac{2P_tR_t}{P_t+R_t}, \text{ where} \\ P_t = \frac{TP_t}{TP_t+FP_t},& \qquad R_t = \frac{TP_t}{TP_t+FN_t}, \end{aligned} \tag{11}$$

Meanwhile, we compare our model with both traditional text classification methods and recent state-of-the-art deep learning based text classification methods.

• **Flat baselines.** This type of methods generally first extract the TF-IDF features from the document, and then input them into the classification model such as one-versus-rest binary support vector machines (BSVM), one-versus-rest binary logistic regression (BLR) and one-versus-rest binary multinomial naive bayes (BMNB) [33], [34]. For these models, a separate binary classifier for each label would be trained on all documents. We call them flat baselines since they ignore both the relations among the words and the relations among the labels.

• 3-gram, sequence-of-words or graph-of-words based models. These methods extract 3-gram features, sequence-of-words or graph-of-words from the document as the input of classification models. These features are suitable for deep learning models, such as **CNN-non-static** [8], **RCNN** [39], **Deep CNN** [12], **XML-CNN** [10], **DGCNN-3** [11], Hierarchical LSTM (**HLSTM**) [64], Hierarchical Attention Network [6] (**HAN**) and Bi-directional Block Self-Attention Network [7] (**Bi-BloSAN**) etc. For example, **HLSTM** model learns sentence representations based on words sequences, and then use RNN models to encode document representations based on the learned sentence representations.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2959991, IEEE Transactions on Knowledge and Data Engineering



**HAN** uses a global attention mechanism to attend useful words and sentences. **Bi-BloSAN** further splits the sequence into several blocks and employs intra-block and inter-block self-attentions to capture both local and long-range context dependencies, respectively.

• **Hierarchical models.** These methods make use of the hierarchical or graphical label network to design hierarchical classification classifiers, such as Top-down Support Vector Machines (**TD-SVM**) [65], Hierarchical SVMs **HSVM** [66], Hierarchically Regularized Logistic Regression (**HR-LR**), Hierarchically Regularized Support Vector Machines (**HR-SVM**) [26], [25], and Hierarchically Regularized Deep Graph CNN (**HR-DGCNN-3**) [11], etc. Note that HSVM model is inherently multi-class method and are not applicable in multi-label scenario. We replace the softmax function with multiple binary logistic functions.

• **Sequence generation model.** These methods view the multi-label classification task as a sequence generation problem, and apply a sequence generation model, such as **SGM+GE** [67], with decoder structure to solve it.

• **Capsule Neural Networks.** These methods are shown to be effective in capturing the spatial features of text, including Capsule Networks with Dynamic Routing (**Capsule-A** and **Capsule-B**) [14]. These capsule networks rely on N-gram convolution networks to extract shallow features and then use dynamic or static routing to learn the relationships between features. **Capsule-B** model employs parallel networks with different sizes of filters. It has been proven to have a better effect than the Capsule-A.

• **Variations of HE-AGCRCNN.** We implement several variants of our proposed method. The first variant model without sorting normalization (Sorting), Long Short-Term Memory units (LSTM), two layers of attentional LSTM (A-LSTM), capsule network (Capsule) and hierarchical weighted margin loss (WML), was named **TGCNN(No-R)**. The second variant model without LSTM units, two layers of attentional LSTM, capsule network and hierarchical weighted margin loss, was named **TGCNN**. The third variant model without two layers of attentional LSTM, capsule network and hierarchical weighted margin loss, was named **TGRCNN**. In order to clearly present the components of the variations of HE-AGCRCNN, we give a table 2 of models that enhance functionality. For other variations of HE-AGCRCNN, we can see the Table 2. Here, for the non-capsule neural network models, we use a 2-layer fully connected networks and a sigmoid layer, and the popular cross entropy or the hierarchical taxonomy-aware weighted margin loss as objective function. For the capsule network models, we use the original margin loss or hierarchical taxonomy embedding-aware weighted margin loss as the objective function.

For deep learning models, we employ the popular cross-entropy loss function. We use the implementations or open source codes of these models released by authors and other researchers, and report the best performance of the results.

### 6.3 Experimental Settings

All our experiments were performed on 64 core Intel Xeon CPU E5-2680 v4@2.40GHz with 512GB RAM and 8×NVIDIA Tesla P100-PICE GPUs. The operating system and software platforms are Ubuntu 5.4.0 and Python 3.5.2. The training and testing datasets are shown in Table 1. Since the RCV1 dataset contains an average of 94.3 unique words per document, the Reuters-21578 contains an average of 93.7 unique words and the EUR dataset contains an average of 183.8 words. In document modeling, the top $w$ numbers of central words are set to 100 (RCV1 and Reuters-21578) and 200 (EUR-Lex). For the sub-graph, the upper bound value $g$ (the maximum number of vertices) is set to 25. The length $h$ of the normalized word sequence is set to 20. The dimension $d$ of word embedding is set to 50. Here, we use word2vec technology to train 50 dimensional word embedding over the 100 billion words from Wikipedia corpus based on Skip-gram with Negative Sample model with window size of 5. For the hierarchical taxonomy embedding, we employ 50 threads to execute the random walk in parallel, and for each walk we use 500 steps. The dimension of label embedding vector is set to 200. For all the deep learning based models, the common parameters of training the models are empirically set, such as batch size $= 32$ and learning rate $= 0.001$ with Adam optimization algorithm.

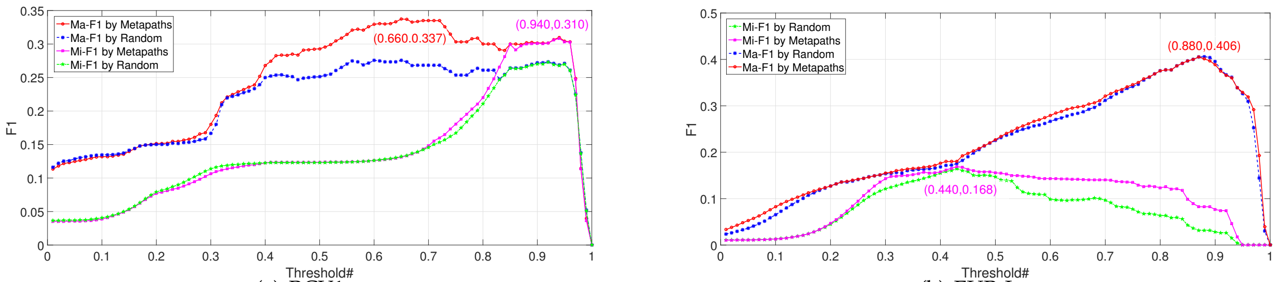
TABLE 2
Comparison of main functions among variations of HE-AGCRCNN.

| Models | CNNs | Sorting | LSTM | A-LSTM | Capsule | WML |
|---|---|---|---|---|---|---|
| TGCNN(No-R) | ✓ | | | | | |
| TGCNN | ✓ | ✓ | | | | |
| TGRCNN | ✓ | ✓ | ✓ | | | |
| GCCNN | ✓ | ✓ | | | ✓ | |
| TAGRCNN | ✓ | ✓ | | ✓ | | |
| GCRCNN | ✓ | ✓ | ✓ | | ✓ | |
| AGCRCNN | ✓ | ✓ | | ✓ | ✓ | |
| HE-TGCNN | ✓ | ✓ | | | | ✓ |
| HE-TGRCNN | ✓ | ✓ | ✓ | | | ✓ |
| HE-GCCNN | ✓ | ✓ | | | ✓ | ✓ |
| HE-TAGRCNN | ✓ | ✓ | ✓ | | | ✓ |
| HE-GCRCNN | ✓ | ✓ | ✓ | | ✓ | ✓ |
| HE-AGCRCNN | ✓ | ✓ | | ✓ | ✓ | ✓ |

For capsule based models, the dimension of capsule vector $m$ is 16, the channel of convolution capsule $M$ is 64, the dimensions of DigitCaps are $32 \times 103$ for the RCV1 dataset, $64 \times 3956$ for the EUR-Lex, and $32 \times 10$ for the reprocessed Reuters-21578, and $m^{+} = 0.9$, $m^{-} = 0.1$, $\lambda = 0.5$. Considering the number of the class labels and the average number of labels per sample, we set the adjustment factor $p$ to 0.01 for RCV1, 0.001 for EUR-Lex and 0.1 for the reprocessed Reuters-21578 in the Eq. 8. All convolution kernels are $1 \times 3$ in size. The numbers of convolution kernels per layer are 64 and 128. The LSTM operator contains 128 hidden layer units. The numbers of neurons in the fully connected layers are 1024 and 512 in RCV1, and 2048 and 4096 in EUR-Lex. Our models can achieve the best performance results among 20 to 70 epoches. For the experiment of transferring single-label model to multi-label classification, on the one hand, in order to be consistent with baseline [14], we select the same number of training, validation and testing data as shown in Table 1. On the other hand, as the labels of the reprocessed Reuters-21578 is part of RCV1's, we reuse the label embedding of RCV1 in the transferring experiment.

### 6.4 Evaluation on Label Embedding

In order to study whether the proposed meta-paths based random walk can learn desirable label embedding that




(a) RCV1 (b) EUR-Lex

Fig. 5. The performance comparisons of reconstructing label network by the two hierarchical taxonomy embedding methods on various thresholds.

reflects the hierarchical taxonomy relations among them, we use the meta-paths guided random walk and traditional random walk to generate the two label sequences respectively, and then generate two label vectors by the same skip-gram method. After obtaining the two vectors, we calculate the cosine distance between them, and use it to reconstruct the relations among the labels. When the distance between two label vectors is larger than the threshold, we add a edge between the two labels. We employ the Macro-$F_1$ and Micro-$F_1$ to evaluate the performance of reconstructing the relations in hierarchy.

Figure 5(a) and figure 5(b) show the results of the two label embedding vectors on the relation reconstruction task in the two datasets. One can see that overall the meta-paths guided random walk approach performs better for capturing the hierarchical taxonomy semantics than the traditional random walk approach. For RCV1, the most suitable thresholds of meta-paths based taxonomy embedding are $0.660$ and $0.940$, and the highest Macro-$F_1$ and Micro-$F_1$ are $0.337$ and $0.310$, respectively. For the traditional random walk based taxonomy embedding, the highest Macro-$F_1$ and Micro-$F_1$ are $0.275$ and $0.272$, respectively. For EUR-Lex, the most suitable thresholds of meta-paths based taxonomy embedding are $0.440$ and $0.880$, and the highest Macro-$F_1$ and Micro-$F_1$ are $0.168$ and $0.406$, respectively. For the traditional random walk based taxonomy embedding, the highest Macro-$F_1$ and Micro-$F_1$ are $0.164$ and $0.405$, respectively. One can observe that the performance difference between the two random walk methods is relatively small on EUR-Lex. This is probably because the taxonomy label structure of EUR-Lex is a hierarchical tree. Although the method of measuring the cosine similarity between any two labels based on unsupervised heterogeneous network representation learning vector is approximate, it's a convenient method to estimate label distance for any two labels.

### 6.5 Performance Evaluation on RCV1

Next, we evaluate the performance on the RCV1 dataset through the multi-label text classification task. RCV1 is a dataset that training samples are much fewer than testing samples, as shown in Table 1.

The experiment results are shown in Table 3. Among the traditional text classification algorithms, one can see that the HR-SVM performs better than TD-SVM, BSVM, HSVM, HR-LR, BLR and BMNB. For deep learning approaches, one can see that the performance of RNN based algorithms HLSTM and HAN are comparable to BSVM and BLR. RCNN performs worse on both settings. For fine-grained topical classification, the above recurrent models may not have advantages because it compresses the whole document as a dense vector for classification. The RNN models are more suitable to sentiment classification for short text, but is not suitable to learn features for long document [11]. For CNN models, it is shown that XML-CNN does not perform very well on RCV1. However, the deeper model DCNN improves the performance by 9% in terms of Macro-F1 and 4% in terms of Micro-F1. For capsule network, one can see that the Capsule-B achieves comparable performance with DCNN model. For sequence generation model, the SGM+GE improves the performance by 2% in terms of Macro-F1 and Micro-F1 compared with the XML-CNN model. For GCNN models, both DGCNN-3 and HR-DGCNN-3 improve the performance by 4% in terms of Macro-F1 and 3% in terms of Micro-F1 compared with Capsule-B. It demonstrates that graph-of-words representation is effective in modeling documents in multi-label text classification. For the popular bi-directional block self-attention network, the Bi-BloSAN improves the performance by 8% in terms of Macro-F1 and 3% in terms of Micro-F1 compared with the HAN model. Among the baselines, XML-CNN, HR-DGCNN-3 and Bi-BloSAN are state-of-the-art methods for multi-label text classification.

TABLE 3
Comparison of results on RCV1 and EUR-Lex.

| Models | RCV1 | | EUR-Lex | |
|---|---|---|---|---|
| | Macro-$F_1$ | Micro-$F_1$ | Macro-$F_1$ | Micro-$F_1$ |
| BLR | 0.328 | 0.692 | 0.181 | 0.522 |
| BSVM | 0.330 | 0.691 | 0.185 | 0.551 |
| BMNB | 0.314 | 0.685 | 0.179 | 0.507 |
| HSVM | 0.333 | 0.693 | 0.189 | 0.567 |
| TD-SVM | 0.337 | 0.696 | 0.198 | 0.571 |
| HR-LR | 0.322 | 0.716 | 0.180 | 0.583 |
| HR-SVM | 0.386 | 0.728 | 0.223 | 0.609 |
| HLSTM | 0.310 | 0.673 | 0.183 | 0.562 |
| HAN | 0.327 | 0.696 | 0.184 | 0.566 |
| RCNN | 0.293 | 0.686 | 0.168 | 0.554 |
| XML-CNN | 0.301 | 0.695 | 0.179 | 0.583 |
| DCNN | 0.399 | 0.732 | 0.231 | 0.611 |
| DGCNN-3 | 0.432 | 0.761 | 0.237 | 0.632 |
| HR-DGCNN-3 | 0.433 | 0.762 | 0.241 | 0.649 |
| SGM+GE | 0.348 | 0.719 | 0.216 | 0.628 |
| Bi-BloSAN | 0.401 | 0.720 | 0.219 | 0.619 |
| Capsule-B | 0.399 | 0.739 | 0.226 | 0.600 |
| TGCNN(No-R) | 0.443 | 0.745 | 0.244 | 0.648 |
| TGCNN | 0.472 | 0.747 | 0.257 | 0.655 |
| GCCNN | 0.480 | 0.749 | 0.261 | 0.658 |
| TGRCNN | 0.484 | 0.754 | 0.265 | 0.667 |
| TAGRCNN | 0.490 | 0.759 | 0.270 | 0.673 |
| GCRCNN | 0.488 | 0.765 | 0.275 | 0.668 |
| AGCRCNN | 0.494 | 0.769 | 0.283 | 0.675 |
| HE-TGCNN | 0.482 | 0.751 | 0.283 | 0.683 |
| HE-GCCNN | 0.491 | 0.754 | 0.290 | 0.688 |
| HE-TGRCNN | 0.495 | 0.762 | 0.292 | 0.680 |
| HE-TAGRCNN | 0.504 | 0.773 | 0.298 | 0.685 |
| HE-GCRCNN | 0.505 | 0.772 | 0.297 | 0.684 |
| HE-AGCRCNN | **0.513** | **0.778** | **0.330** | **0.688** |

Compared with these models, our model fully demonstrates the importance of extracting multiple semantics when text modeling and feature learning.

For the proposed models, we try different model configurations listed in Table 2. The results are shown in Table 3. One can see that the LSTM units, attentional LSTM units, capsule networks and hierarchical taxonomy-aware weighted margin loss are all helpful to improve the classification performance. The proposed HE-AGCRCNN model outperforms the HR-DGCNN-3 by 8% in terms of Macro-F1. The simplified model TGCNN(No-R) also achieves comparable Macro-F1 and Micro-F1 with DGCNN-3. Meanwhile, without attentional LSTM units, capsule network and hierarchical label dependencies, the TGCNN also outperforms most of the baselines. Based on the arranged word matrix representation, LSTM units and attentional LSTM units, the TGRCNN and TAGRCNN models achieve 5%-6% improvements in terms of Macro-$F_1$ over HR-DGCNN-3. This improvements show the importance of local sequential semantics for text features. Among the proposed models, one can see that capsule networks averagely achieve 1% gain in both Macro-$F_1$ and Micro-$F_1$. Overall, the hierarchical taxonomy-aware weighted margin loss can also improve the performances by 2% in terms of Macro-$F_1$ and 1% in terms of Micro-$F_1$. Finally, the proposed HE-AGCRCNN model achieves the highest $0.513$ Macro-$F_1$ and $0.778$ Micro-$F_1$ performance. The results of the different document modeling methods show that by representing document as arranged word matrix, the proposed model can gain performance improvement for multi-label text classification in RCV1. One can also see that HR-SVM, HR-DGCNN-3 and HE-AGCRCNN represent two different ways of using the hierarchical label dependencies, and both improve the classification performance over RCV1 dataset.

### 6.6 Performance Evaluation on EUR-Lex

As the number of labels in EUR-Lex is large, we use more neurons in the fully connected layers and set a larger dimension of capsule vector in the DigitCaps layer, as presented in Section 6.3. For the proposed models, we also try different configurations, and the results are shown in Table 3.

From the results one can see that LSTM units, attentional LSTM units, capsule networks and hierarchical taxonomy-aware weighted margin loss are all helpful to improve classification performance on the EUR-Lex dataset. HE-AGCRCNN model achieves about 6% improvements in terms of Macro-$F_1$ and 4% gains in terms of Micro-$F_1$ over the HR-DGCNN-3 model. Without using hierarchical label dependencies, LSTM units, attentional LSTM units and capsule networks, the TGCNN also performs better than HR-DGCNN-3, and the results are $0.257$ and $0.655$ for Macro-$F_1$ and Micro-$F_1$, respectively. When we do not order the words in the sub-graph, the results of TGCNN(No-R) are $0.244$ and $0.648$ for Macro-$F_1$ and Micro-$F_1$, respectively. The performance gap between TGCNN(No-R) and TGCNN shows the importance of local sequential semantics for text classification with the same three layers of Graph CNN models. Based on the arranged word matrix representation, the LSTM units can help to improve 1% performance comparing GCCNN and GCRCNN. The performance gap between TGRCNN and TAGRCNN shows that the masked attentional units can help to improve about 0.5% performance. The hierarchical taxonomy-aware weighted margin loss also helps to improve about 1%-3% performances in terms of Macro-$F_1$ or Micro-$F_1$. Compared with the improvements of the hierarchical taxonomy-aware weighted margin loss in the RCV1 dataset, the improvements in the EUR-Lex dataset are greater by using the same weighted margin loss. Finally, HE-AGCRCNN model achieves $0.330$ Macro-$F_1$ and $0.688$ Micro-$F_1$, which are both the highest performance. The experimental results again demonstrate that by representing document as an arranged word matrix and incorporating the proposed deep models, one can gain benefits from non-consecutive, long-distance and sequential semantics for topical multi-label text classification.

TABLE 4
Comparison of the transferring capacity from single-label to multi-label text classification on the reprocessed Reuters-21578 dataset.

| Models | P | R | Micro-$F_1$ |
|---|---|---|---|
| LSTM | 0.867 | 0.547 | 0.635 |
| BiLSTM | 0.823 | 0.559 | 0.643 |
| CNN-non-static | 0.920 | 0.597 | 0.704 |
| Capsule-A | 0.882 | 0.801 | 0.820 |
| Capsule-B | 0.954 | 0.820 | 0.858 |
| GCCNN | 0.962 | 0.856 | 0.905 |
| GCRCNN | 0.970 | 0.871 | 0.917 |
| AGCRCNN | 0.973 | 0.875 | 0.921 |
| HE-GCCNN | 0.965 | 0.862 | 0.910 |
| HE-GCRCNN | 0.974 | 0.879 | 0.924 |
| HE-AGCRCNN | **0.978** | **0.882** | **0.927** |

### 6.7 Performance Evaluation on Reuters-21578

A significant advantage of capsule network is that it performs much better in the transferring single-label to multi-label classification task [14]. Different from traditional classification models that are based on fully connected network, capsule networks use activity vectors of each capsule in DigitCaps layer to indicate the presence of an instance of each class. We also perform the model transfer capacity experiment of the proposed capsule network on the reprocessed Reuters-21578 dataset [14].

The comparison results are shown in Table 4. Since the traditional multi-label learning methods, such as BSVM, TD-SVM, HSVM, HR-SVM, BLR, HR-LR and BMNB, cann't transfer single-label to multi-label classification. We ignore the traditional machine learning models in the current experiment. The baseline results are also reported from the work [14], and HE-AGCRCNN outperforms all the baselines. Compared with capsule-based models, the performances of LSTM, BiLSTM and CNN-non-static are the worst. From the results one can see that the proposed models all have achieved about 5%-7% improvements in terms of Micro-$F_1$ over the existing best baseline Capsule-B. Even without the attentional LSTM units, the simplified GCCNN can achieve $0.905$ performance in terms of Micro-$F_1$. Compared with the popular Capsule-A and Capsule-B models, the proposed models integrate more non-consecutive, long-distance and local sequential semantics, and make use of the hierarchical label dependencies. Finally, the proposed HE-AGCRCNN model achieves the highest $0.927$ performance in terms of Micro-$F_1$. The experimental improvements again prove the effectiveness of our proposed capsule models in learning rich textual features.

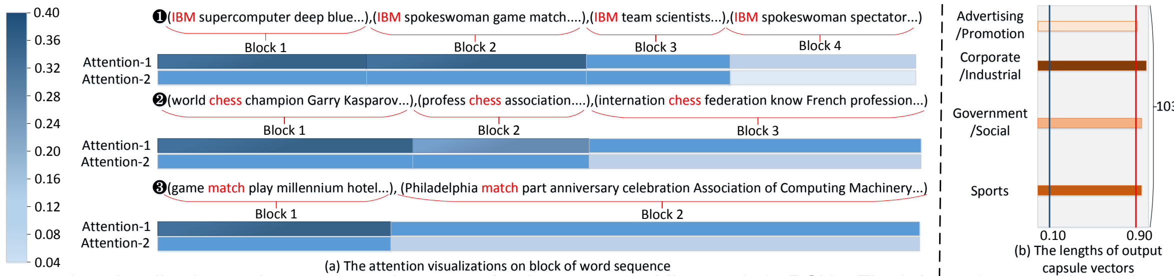



Fig. 6. The attention visualization and output capsule vectors for the 3093newsML sample in RCV1. The left one is parts of attention visualization on blocks of word sequences. We show the importance weight of the top-3 central words in different contexts at the two layers of attention network. The right one is the length of output capsule vectors.

TABLE 5
Comparison of time consumption(hr.).

| Models | RCV1 | EUR-Lex | Reuters-21578 |
|---|---|---|---|
| Preprocess | 0.612 | 1.264 | 0.151 |
| TGCNN | 0.166 | 1.100 | - |
| GCCNN | 0.537 | 3.415 | 0.131 |
| TGRCNN | 0.381 | 2.579 | - |
| TAGRCNN | 0.382 | 2.580 | - |
| GCRCNN | 1.116 | 6.327 | 0.267 |
| AGCRCNN | 1.117 | 6.328 | 0.267 |
| HE-TGCNN | 0.167 | 1.167 | - |
| HE-GCCNN | 0.542 | 3.421 | 0.132 |
| HE-TGRCNN | 0.385 | 2.583 | - |
| HE-TAGRCNN | 0.386 | 2.584 | - |
| HE-GCRCNN | 1.118 | 6.334 | 0.136 |
| HE-AGCRCNN | 1.119 | 6.335 | 0.136 |

### 6.8 Efficiency Evaluation

To evaluate the training efficiency of the model, we next show the preprocessing and training time of the proposed model and its variants on both RCV1, EUR-LEX and Reuters-21578 datasets in Table 5. Here, for the RCV1, since the number of samples in the test set is about 34 times larger than the number of samples in the training set, as shown in Table 1, we perform the testing by using multi-core CPUs. The preprocessing includes document modeling, converting the original text into 3-D tensor representation, and hierarchical taxonomy embedding. We run the document modeling and hierarchical taxonomy embedding on different devices, and then count the maximum time consumption. For the Reuters-21578, we only test the training time of the model with capsule network because only the capsule network could achieve the transferring capacity from single-label to multi-label text classification.

For the pre-processing with 64 core multi-threading, RCV1 takes 0.612 hours, ERU 1.264 hours and Reuters-21578 0.151 hours. Although the number of training sample in EUR-LEX is smaller, the average words of its text is larger than that of RCV1, which leads to longer preprocessing time in EUR-LEX. Due to the minimum amount of training sample, the pre-processing time of Reuters-21578 dataset is 0.151 hours. Compared with the improvement of the proposed models, the time consumption of preprocessing is worth it. One can observe that most of these models can quickly achieve a promising classification result with less than 3 hours except for the models with the LSTM or capsule unites. For example, the TGCNN and HE-TGCNN models converge quickly with less than 0.2 hours on RCV1 dataset and less than 1.2 hours on EUR-Lex dataset. Meanwhile, the training time on RCV1 dataset is much less than EUR-Lex. This is mainly because the EUR-Lex dataset has a larger document representation and more parameters, according to Table 1. We also verify that the models integrating more feature extraction operators, such as LSTM units, attentional LSTM units and capsule networks, will take longer time to train for achieving a desirable classification performance. Although HE-AGCRCNN model takes 1.119 hours, 6.335 hours, and 0.136 hours to train for RCV1, EUR-Lex and Reuters-21578 datasets, respectively, it achieves the highest classification performance. One can also see that the hierarchical taxonomy-aware weighted margin loss does not add much computational time compared with the recursively regularized optimization models [11], [26], [25]. Usually, the time consumptions of the above recursive regularization based models are expensive for the large number of parameters and constraints on the Euclidean distance of the parameters. In particular, the time consumptions of recursive regularization optimized deep learning model, such as HR-DGCNN-3, is generally measured in days [11].

### 6.9 Case study

To gain a closer view of what's the two attention layers and output capsules in a document captured by our models, we visualize parts of the attention probability or alignment score by heatmaps in Figure 6. The red words are central words in the document, each block of word sequence is context of central words. For each central word, there are two layers of masked attention. We choose the blocks of word sequences from the 1-th, 2-nd and 3-rd central words. One can see that the weights of the upper left parts are higher than other places. This is probably because the contextual semantics of the front central words are more representative of the subject of the article. For the output capsule vectors, there are 4 vectors whose modulus length is greater than 0.9, corresponding to the category of output. Note that Sports category comes out due to the words game, match, defeat and win, although chess is not really a physical activity.

## 7 Conclusion and Future Work

In this paper, we present a novel end-to-end hierarchical taxonomy-aware and attentional graph capsule recurrent CNN framework for large-scale multi-label text classification. We first propose to convert each document as an arranged word matrix that preserves both the non-consecutive, long-distance and local sequential semantics for fully representing the document. Based on our document modeling, we next propose a HE-AGCRCNN model to coherently learn multiple types of textual features. In order to better learn local sequential semantics, we design a masked attentional LSTM to model the different impacts among different blocks of word sequences, and enhance the sequential feature learning. To incorporate the hierarchical relations among the labels, we further propose a novel hierarchical taxonomy-aware weighted margin loss to improve the performance of multi-label text classification. The advantageous performance of our proposed models over other competing methods is evident as it obtained the best results on all the RCV1 and EUR-Lex datasets in our comparative evaluation. Compared to the N-gram based textual capsule networks, we verify the effectiveness of our proposed capsule models in learning rich textual features in transferring single-label to multi-label classification task. The experimental results show the effectiveness and efficiency of our model in multi-label text classification.

In the future, we plan to invest more centrality measures, stratified sampling, self attention models [7], [18], BERT [68] pre-training based models and less memory consumption, and popularize to other complex text classification datasets and applications.

## References


[1] C. C. Aggarwal and C. Zhai, "A survey of text classification algorithms," in *Mining Text Data*, 2012, pp. 163–222.

[2] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. The MIT Press, 2016.

[4] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *ACL*, 2015, pp. 1556–1566.

[5] Y. Wang, M. Huang, X. Zhu, and L. Zhao, "Attention-based LSTM for aspect-level sentiment classification," in *EMNLP*, 2016, pp. 606–615.

[6] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *NAACL*, 2017, pp. 1480–1489.

[7] T. Shen, T. Zhou, G. Long, J. Jiang, and C. Zhang, "Bi-directional block self-attention for fast and memory-efficient sequence modeling," in *ICLR*, 2018.

[8] Y. Kim, "Convolutional neural networks for sentence classification," in *EMNLP*, 2014, pp. 1746–1751.

[9] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *AAAI*, vol. 33, 2019, pp. 7370–7377.

[10] J. Liu, W. Chang, Y. Wu, and Y. Yang, "Deep learning for extreme multi-label text classification," in *SIGIR*, 2017, pp. 115–124.

[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 *WWW*, 2018, pp. 1063–1072.

[12] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," in *EACL*, 2016, pp. 1107–1116.

[13] Y. He, J. Li, Y. Song, M. He, and H. Peng, "Time-evolving text classification with deep neural networks," in *IJCAI*, 2018.

[14] M. Yang, W. Zhao, J. Ye, Z. Lei, Z. Zhao, and S. Zhang, "Investigating capsule networks with dynamic routing for text classification," in *EMNLP*, 2018, pp. 3110–3119.

[15] P. Liu, X. Qiu, and X. Huang, "Adversarial multi-task learning for text classification," in *ACL*, 2017, pp. 1–10.

[16] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial training methods for semi-supervised text classification," in *ICLR*, 2017.

[17] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *EMNLP*, 2015, pp. 1422–1432.

[18] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang, "Disan: Directional self-attention network for rnn/cnn-free language understanding," in *AAAI*, 2018.

[19] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *JMLR*, pp. 1137–1155, 2003.

[20] F. Rousseau, E. Kiagias, and M. Vazirgiannis, "Text categorization as a graph classification problem," in *ACL*, 2015, pp. 1702–1712.

[21] A. Sun and E.-P. Lim, "Hierarchical text classification and evaluation," in *ICDM*, 2001, pp. 521–528.

[22] G.-R. Xue, D. Xing, Q. Yang, and Y. Yu, "Deep classification in large-scale text hierarchies," in *SIGIR*, 2008, pp. 619–626.

[23] S. Gopal, Y. Yang, B. Bai, and A. Niculescu-Mizil, "Bayesian models for large-scale hierarchical classification," in *NIPS*, 2012, pp. 2420–2428.

[24] M. W. Berry, *Survey of Text Mining*. Berlin, Heidelberg: Springer-Verlag, 2003.

[25] S. Gopal and Y. Yang, "Hierarchical bayesian inference and recursive regularization for large-scale classification," *TKDD*, 2015.

[26] G. Siddharth and Y. Yiming, "Recursive regularization for large-scale classification with hierarchical and graphical dependencies," in *KDD*, 2013, pp. 257–265.

[27] S. Xie, X. Kong, J. Gao, W. Fan, and S. Y. Philip, "Multilabel consensus classification," in *ICDM*, 2013, pp. 1241–1246.

[28] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

[29] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[30] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *JMLR*, vol. 3, pp. 993–1022, 2003.

[31] R. A. Sinoara, R. G. Rossi, and S. O. Rezende, "Semantic role-based representations in text classification," in *ICPR*, 2016, pp. 2313–2318.

[32] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *TKDE*, vol. 26, no. 8, pp. 1819–1837, 2014.

[33] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *IJDWM*, vol. 3, no. 3, pp. 1–13, 2007.

[34] A. C. de Carvalho and A. A. Freitas, "A tutorial on multi-label classification techniques," in *Foundations of Computational Intelligence Volume 5*, 2009, pp. 177–195.

[35] A. Garg, J. Noyola, R. Verma, A. Saxena, and A. Jami, "Exploring correlation between labels to improve multi-label classification," *arXiv preprint arXiv:1511.07953*, 2015.

[36] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," in *ICLR*, 2017.

[37] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *ACL*, 2014.

[38] X. Zhang, J. J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *NIPS*, 2015, pp. 649–657.

[39] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *AAAI*, 2015, pp. 2267–2273.

[40] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *ICANN*, 2011, pp. 44–51.

[41] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *NIPS*, 2017, pp. 3859–3869.

[42] G. Hinton, N. Frosst, and S. Sabour, "Matrix capsules with em routing," in *ICLR*, 2018.

[43] L. Xiao, H. Zhang, W. Chen, Y. Wang, and Y. Jin, "Mcapsnet: Capsule network for text with multi-task learning," in *EMNLP*, 2018, pp. 4565–4574.

[44] N. Zhang, S. Deng, Z. Sun, X. Chen, W. Zhang, and H. Chen, "Attention-based capsule networks with dynamic routing for relation extraction," in *EMNLP*, 2018, pp. 986–992.

[45] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, pp. 83–98, 2013.

[46] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," in *ICLR*, 2014.

[47] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NIPS*, 2016, pp. 3837–3845.

[48] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[49] D. K. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *NIPS*, 2015, pp. 2224–2232.

[50] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *ICML*, 2016, pp. 2014–2023.

[51] S. Verma and Z.-L. Zhang, "Graph capsule convolutional neural networks," *arXiv preprint arXiv:1805.08090*, 2018.

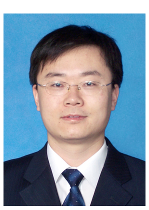
[52] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *TKDE*, vol. 31, no. 5, pp. 833–852, 2018.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2959991, IEEE Transactions on Knowledge and Data Engineering



[53] G. K. Palshikar, "Keyword extraction from a single document using centrality measures," in *PRAI*, 2007, pp. 503–510.
[54] A. Onan, S. Korukoğlu, and H. Bulut, "Ensemble of keyword extraction methods and classifiers in text classification," *Expert Systems with Applications*, vol. 57, pp. 232–247, 2016.
[55] R. Goldstein and M. S. Vitevitch, "The influence of closeness centrality on lexical processing," *Frontiers in psychology*, vol. 8, p. 1683, 2017.
[56] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *SIGKDD*, 2017, pp. 135–144.
[57] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*, 2014, pp. 701–710.
[58] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *JMLR*, 2004.
[59] S. Gopal and Y. Yang, "Recursive regularization for large-scale classification with hierarchical and graphical dependencies," in *KDD*, 2013, pp. 257–265.
[60] E. Loza Mencía and J. Fürnkranz, "Efficient pairwise multilabel classification for large-scale problems in the legal domain," in *ECML/PKDD*, 2008, pp. 50–65.
[61] S. Gopal and Y. Yang, "Multilabel classification with meta-level features," in *SIGIR*, 2010, pp. 315–322.
[62] D. D. Lewis, "An evaluation of phrasal and clustered representations on a text categorization task," in *SIGIR*, 1992, pp. 37–50.
[63] A. F. Giraldo-Forero, J. A. Jaramillo-Garzón, and C. G. Castellanos-Domínguez, "Evaluation of example-based measures for multi-label classification performance," in *ICBBE*, 2015, pp. 557–564.
[64] H. Chen, M. Sun, C. Tu, Y. Lin, and Z. Liu, "Neural sentiment classification with user and product attention," in *EMNLP*, 2016.
[65] T. Liu, Y. Yang, H. Wan, H. Zeng, Z. Chen, and W. Ma, "Support vector machines classification with a very large-scale taxonomy," *SIGKDD Explorations*, vol. 7, no. 1, pp. 36–43, 2005.
[66] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *JMLR*, vol. 6, pp. 1453–1484, 2005.
[67] P. Yang, X. Sun, W. Li, S. Ma, W. Wu, and H. Wang, "Sgm: sequence generation model for multi-label classification," in *ACL*, 2018.
[68] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019, pp. 4171–4186.

**Hao Peng** is currently a Ph.D. candidate at the State Key Laboratory of Software Development Environment, and Beijing Advanced Innovation Center for Big Data and Brain Computing in Beihang University. His research interests include representation learning, text mining and urban computing.



**Jianxin Li** is currently a Professor with the State Key Laboratory of Software Development Environment, and Beijing Advanced Innovation Center for Big Data and Brain Computing in Beihang University. His current research interests include machine learning, distributed system, trust management and network security.

**Senzhang Wang** is currently an Associate Professor with the Collage of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing. His current research interests include data mining, urban computing and social network analysis.

**Lihong Wang** is a professor in National Computer Network Emergency Response Technical Team/Coordination Center of China. Her current research interests include information security, cloud computing, big data mining and analytics, information retrieval and data mining.

**Qiran Gong** is currently a B.E. candidate at the State Key Laboratory of Software Development Environment in Beihang University, Beijing, China. His research interests include social network mining and text mining.

**Renyu Yang** is a research fellow with University of Leeds, UK. He received BSc and PhD degree from Beihang University in 2011 and 2017. His research interests include reliable distributed systems and data analytics.

**Bo Li** , is currently an Associate Professor with the State Key Laboratory of Software Development Environment, and Beijing Advanced Innovation Center for Big Data and Brain Computing in Beihang University. His current research interests include big data computing theory, machine learning and computer security.

**Lifang He** is currently a Postdoctoral Research Associate at the Department of Biostatistics and Epidemiology at the University of Pennsylvania. Her current research interests include machine learning, data mining, tensor analysis, biomedical informatics.

**Philip S. Yu** is a Distinguished Professor and the Wexler Chair in Information Technology at the Department of Computer Science, University of Illinois at Chicago. Before joining UIC, he was at the IBM Watson Research Center, where he built a world-renowned data mining and database department. He is a Fellow of the ACM and IEEE. Dr. Yu has published more than 1,100 referred conference and journal papers cited more than 103,000 times with an H-index of 155. He has applied for more than 300 patents. Dr. Yu was the Editor-in-Chiefs of ACM Transactions on Knowledge Discovery from Data (2011-2017) and IEEE Transactions on Knowledge and Data Engineering (2001-2004).