Research Article

A Hierarchical Neural-Network-Based Document Representation Approach for Text Classification

Jianming Zheng , Yupu Guo, Chong Feng, and Honghui Chen

Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Hunan, China

Correspondence should be addressed to Jianming Zheng; zhengjianming12@nudt.edu.cn

Received 28 October 2017; Revised 7 February 2018; Accepted 26 February 2018; Published 29 March 2018

Academic Editor: Xiangyu Meng

Copyright © 2018 Jianming Zheng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Document representation is widely used in practical application, for example, sentiment classification, text retrieval, and text classification. Previous work is mainly based on the statistics and the neural networks, which suffer from data sparsity and model interpretability, respectively. In this paper, we propose a general framework for document representation with a hierarchical architecture. In particular, we incorporate the hierarchical architecture into three traditional neural-network models for document representation, resulting in three hierarchical neural representation models for document classification, that is, TextHFT, TextHRNN, and TextHCNN. Our comprehensive experimental results on two public datasets, that is, Yelp 2016 and Amazon Reviews (Electronics), show that our proposals with hierarchical architecture outperform the corresponding neural-network models for document classification, resulting in a significant improvement ranging from 4.65% to 35.08% in terms of accuracy with a comparable (or substantially less) expense of time consumption. In addition, we find that the long documents benefit more from the hierarchical architecture than the short ones as the improvement in terms of accuracy on long documents is greater than that on short documents.

1. Introduction

Text representation, as a challenging task in Natural Language Processing (NLP), converts text spans into real-valued vectors or matrix, which is crucial for machine to understand the semantics of the text. From the text generation frame (words form phrases or sentences, and sentences form a document [1]), text representation can be divided into the following levels: word-level representation (e.g., word2vec [2], GloVe [3]), phrase-level representation [4], sentence-level representation [5], and document-level representation [6]. In this paper, we focus on document-level representation. The document-level representation has broad applications such as sentiment classification [7], text retrieval [8], and text ranking [9].

The most common and simplest approaches for text representation are bag-of-words (BoW) [10] and n -grams with TF-IDF [11]. However, such statistical-based methods suffer from the problem of data sparsity and dimensionality when they are applied on a large-scale corpus. Recently, plenty of approaches based on different neural-network architectures or on their combinations have been proposed

for text presentation, for example, FastText (a hidden layer based) [12], TextCNN (convolutional neural networks based) [13], TextRNN (recurrent neural networks based) [14], and TextRCNN (recurrent convolutional neural networks based) [15]. Such neural-network-based models can generate low-dimensional vectors to represent text, overcoming the problem of data sparsity. In addition, compared to BoW and n -grams based approaches, neural-network-based models can capture a better semantic relationship between words [12]. However, the existing neural-network-based text presentation models are individually trained for one or multiple specific tasks, for example, sentiment analysis [16] or text classification [14], ignoring the internal structure features of the text itself, for example, the words-sentence and sentences-text relationship, which we argue can be regarded as prior knowledge to help generate a better text representation.

Hence, in this paper, we propose a general structure for document representation by injecting hierarchical architecture into neural networks. Our proposal mainly consists of a sentence representation at the word level and then a document representation at the sentence level. At the word

level, each sentence can be represented by utilizing specific neural networks to aggregate the embeddings of words in the sentence. Similarly, at the sentence level, the document is represented by aggregating all sentences generated from the former step. We implement our proposal on public large-scale text datasets for document classification. Our experimental results indicate that the hierarchical architecture does help to improve the performance after being incorporated into the existing neural networks based baselines, for example, FastText [12], TextCNN [13], and TextRNN [14].

Our major contributions are summarized as follows:

- (i) We tackle the challenge of document representation for text classification by incorporating the hierarchical architecture into neural networks models.
- (ii) We theoretically analyze the computational complexity of our new neural models after injecting the hierarchical architecture into existing neural networks models.
- (iii) We conduct comprehensive experiments for document classification on large-scale public datasets. We find that our proposals significantly outperform the corresponding state-of-the-art baselines, achieving an improvement of around 8% in terms of accuracy with comparable or substantially less computation expense.

The remainder of this paper is organized as follows: we describe the related works in Section 2. Our proposals are described in Section 3. Section 4 presents our experimental setup. In Section 5, we report and discuss our experimental results. Finally, we conclude in Section 6.

2. Related Work

In this section, we briefly summarize the approaches for document classification based on various text representation schemes, that is, the traditional statistical representation (see Section 2.1) and the neural-network-based representation (see Section 2.2). In addition, we present the major differences between our proposal and previous works.

2.1. Statistical Representation Based Document Classification.

As a word is the most basic unit of semantics, the traditional one-hot representation model converts the word in vocabulary into a sparse vector with a single high value (e.g., 1) in its position and all the others low (e.g., 0), which is employed in the bag-of-words (BoW) model [10] to reflect the word frequency information. However, the BoW model can only symbolize the words and cannot reflect the semantic relationship between words. In view of that, the bag-of-means model [11] is proposed to cluster the words embeddings learned by the word2vec model [2] for text representation. Furthermore, the bag-of- n -grams [11] is developed to take the n -gram words orders into account for text representation, which selects the most frequent n -grams (up to 5 grams) as the vocabulary in the BoW model. In addition, with some extra statistical information added to the BoW model, for

example, TF-IDF [17], a better text representation is achieved. Besides, incorporating the text features into the representation learning process, for example, the noun phrases [18] and the tree kernels [19], can further improve the accuracy of document classification.

Clearly, a progressive step has been made to text classification based on statistical representation. However, such traditional statistical representation approaches inevitably suffer from the problem of data sparsity and dimensionality, which leads to no applications to large-scale corpus. In addition, such approaches are simply built on shallow statistics and deeper semantic information of text has not been well developed. Instead, our proposal in this paper based on deep learning of neural network has the ability to learn the low-dimension vectors to overcome such problems.

2.2. Neural Representation Based Document Classification.

Since Bengio et al. [20] first employed the neural-network architecture to train the language model, a great attention has been paid to proposing the neural-network related models for document classification. For instance, the FastText model proposed by Joulin et al. [12] uses 1 hidden layer to integrate all input information and present considerable results. However, this model only concerns the mean of word vectors and discards the signal of word order. In order to overcome the problem of insufficient training data which often appears in a single-task supervised learning, Liu et al. [14] use the multitask learning framework with RNN structure to jointly learn across multiple related tasks. Compared to RNN, CNN is easier to train and capture the global text information. For instance, Kim [13] employs the CNN to classify documents, while Zhang et al. [11], in character level, also employ a CNN to represent documents. Furthermore, a combination of these neural-network models can integrate the advantages of a single neural-network. For instance, RCNN is proposed to adopt the recurrent structure to grasp the context information and can identify the key components in text by employing a max-pooling layer [15].

Although the neural-network models above utilize a complex neural-network structure to deep learning and are able to develop the hidden text features, such models are built on one or multiple tasks lack of interoperability. In addition, such approaches directly employ the neural-network architecture to get the document representation vectors without considering the structure features of texts, making models less interpretable. Instead, our proposal pays special attention to the process of generating texts, which is based on hierarchical architecture and can improve the interpretability of neural network based models.

3. Approach

In this section, we first formally describe our proposal, that is, the hierarchical neural representation in Section 3.1. Then, in Section 3.2, we detail three new proposed models based on our hierarchical neural architecture, that is, TextHFT, TextHRNN, and TextHCNN, which basically combine the

FIGURE 1: The framework of the hierarchical neural architecture for document classification.

hierarchical architecture with the corresponding models, that is, FastText, RNN, and CNN, respectively. In addition, a comprehensive computational complexity analysis is conducted on all discussed models.

3.1. General Framework. First of all, we propose a general framework for document representation with a hierarchical neural architecture. Figure 1 illustrates the major workflow of our proposal for document representation. Let us make a brief illustration. Given a document d that consists of n_2 sentences with each sentence s_i having n_3 words, we denote the document as $d = \{s_1, s_2, \dots, s_{n_2}\}$ and the sentence as $s_i = \{w_{s_i}^1, w_{s_i}^2, \dots, w_{s_i}^{n_3}\}$, where $w_{s_i}^j$ represents the j th word in sentence s_i .

Next, as indicated in Figure 1, our proposed hierarchical neural architecture for document representation mainly consists of six processes.

Word Representation. We use the word embedding to convert each word $w_{s_i}^j$ into a real-valued vector $\mathbf{w}_{s_i}^j$ ($\mathbf{w}_{s_i}^j \in \mathbb{R}^D$), D is the dimension of word embeddings.

Word Combination. To integrate all words in a sentence s_i , the representation $\{\mathbf{w}_{s_i}^1, \mathbf{w}_{s_i}^2, \dots, \mathbf{w}_{s_i}^{n_3}\}$ obtained from the previous step is used as input to the neural network at the word level.

Sentence Representation. By word combination, we can get the output of the neural networks and regard it as the representation $\mathbf{s}_i$ of sentence s_i .

Sentence Combination. Similar to word combination, we use the representation of all sentences $\mathbf{s}_j$ ($\mathbf{s}_j \in \mathbb{R}^D$) with $j =$

$\{1, 2, \dots, n_2\}$ as the input of the neural networks at the sentence level.

Document Representation. Likewise, we can get the output of the neural networks at the sentence level as the representation $\mathbf{d}$ ($\mathbf{d} \in \mathbb{R}^D$) of document d .

Document Classification. The document representation $\mathbf{d}$ can be used as input for document classification. After the document representation module, we transform $\mathbf{d}$ into $\mathbf{d}^* = (x_1, x_2, \dots, x_c)$ by a fully connected layer, where c indicates the total number of document categories. Then, we employ a softmax function to produce the predictive distribution $\mathbf{q}$ over all document categories, where its element q_i indicates the probability of that document belonging to a specific category i :

$$q_i = \frac{\exp x_i}{\sum_{i=1}^c \exp x_i}. \quad (1)$$

3.2. Hierarchical Neural Representation. In this section, we present our proposed document representation models with hierarchical architecture (i.e., TextHFT, TextHRNN, and TextHCNN) and make a detailed analysis to their complexity. For simplicity, suppose that we have n_1 documents in the corpus, each document has the same length n_2 , and each sentence has the same length n_3 .

3.2.1. TextHFT. As shown in Figure 2, the key component of FastText integrates all word representations on a hidden layer. Different from the traditional FastText that directly averages all word embeddings of the document, TextHFT first averages all word embeddings of a sentence to get the sentence representation and then averages all sentence representations to get the document representation. Thus, we can get

FIGURE 2: The structure of FastText.

$$\mathbf{s}_i = \frac{1}{n_3} \sum_{j=1}^{n_3} \mathbf{w}_{s_i}^j \quad (\mathbf{s}_i, \mathbf{w}_{s_i}^j \in \mathbb{R}^D) \quad (2)$$

at the word level.

At the sentence level, we can then represent a document

$$\mathbf{d} = \frac{1}{n_2} \sum_{i=1}^{n_2} \mathbf{s}_i \quad (\mathbf{d} \in \mathbb{R}^D). \quad (3)$$

By doing so, each document can be represented by a vector.

From the above analysis, the complexity of this hierarchical architecture is mainly related to the sequence length. In particular, the complexity for FastText is $O(n)$, where $n = n_1 \times n_2 \times n_3$. However, in the TextHFT model, the complexity at the word level is $O(n_1 \times n_2 \times n_3)$, that is, $O(n)$, and that at the sentence level is $O(n_1 \times n_2)$, respectively. In total, the complexity for TextHFT is $O(n(1 + 1/n_3))$.

3.2.2. TextHRNN. To overcome the problem of gradient disappearance and of context scarcity, we implement the bidirectional Long Short-Term Memory RNN (Bi-LSTM) model to the text sequence (shown in Figure 3).

In detail, at the word level, we first use Bi-LSTM to process the word sequence $\{\mathbf{w}_{s_i}^1, \mathbf{w}_{s_i}^2, \dots, \mathbf{w}_{s_i}^{n_3}\}$ with $i = \{1, 2, \dots, n_2\}$ to output

$$\begin{aligned} \vec{h}_t^{s_i} &= \overrightarrow{\text{LSTM}}(\mathbf{w}_{s_i}^t, \vec{h}_{t-1}^{s_i}) \quad (\vec{h}_t^{s_i} \in \mathbb{R}^p), \\ \overleftarrow{h}_t^{s_i} &= \overleftarrow{\text{LSTM}}(\mathbf{w}_{s_i}^t, \overleftarrow{h}_{t+1}^{s_i}) \quad (\overleftarrow{h}_t^{s_i} \in \mathbb{R}^p), \end{aligned} \quad (4)$$

where $t = 1, 2, \dots, n_3$ and p is the dimension of hidden output. $\vec{h}_t^{s_i}$ and $\overleftarrow{h}_t^{s_i}$ are the outputs of forward LSTM and backward LSTM, respectively. We then concatenate the $\vec{h}_t^{s_i}$ with $\overleftarrow{h}_t^{s_i}$ to obtain a hidden output $\mathbf{h}_t^{s_i}$ as

$$\mathbf{h}_t^{s_i} = \left(\vec{h}_t^{s_i}; \overleftarrow{h}_t^{s_i} \right), \quad (\mathbf{h}_t^{s_i} \in \mathbb{R}^{2p}, t = 1, 2, \dots, n_3). \quad (5)$$

After that, to encode the hidden output $\mathbf{h}_t^{s_i}$ into the sentence representation $\mathbf{s}$ with a fixed length, we add a fully connected layer as

$$\mathbf{s}_i = \mathbf{W}_{R,s}^T \mathbf{h}_{n_3}^{s_i} + \mathbf{b}_{R,s}, \quad (6)$$

where $\mathbf{W}_{R,s} \in \mathbb{R}^{2p \times D}$ is a weight matrix and $\mathbf{b}_{R,s} \in \mathbb{R}^D$ is a bias term.

Similarly, at the sentence level, by using Bi-LSTM to the sentence sequence $\{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_{n_2}\}$, we will produce

$$\begin{aligned} \vec{h}_t^d &= \overrightarrow{\text{LSTM}}(\mathbf{s}_t, \vec{h}_{t-1}^d) \quad (\vec{h}_t^d \in \mathbb{R}^{2p}), \\ \overleftarrow{h}_t^d &= \overleftarrow{\text{LSTM}}(\mathbf{s}_t, \overleftarrow{h}_{t+1}^d) \quad (\overleftarrow{h}_t^d \in \mathbb{R}^{2p}), \end{aligned} \quad (7)$$

where $t = \{1, 2, \dots, n_2\}$. Again, we concatenate each $\vec{h}_t^d$ with $\overleftarrow{h}_t^d$ to obtain a hidden output $\mathbf{h}_t^d$ as

$$\mathbf{h}_t^d = \left(\vec{h}_t^d; \overleftarrow{h}_t^d \right), \quad (\mathbf{h}_t^d \in \mathbb{R}^{2p}). \quad (8)$$

Then, the document representation $\mathbf{d}$ can be generated by

$$\mathbf{d} = \mathbf{W}_{R,d}^T \mathbf{h}_{n_2}^d + \mathbf{b}_{R,d}, \quad (9)$$

where $\mathbf{W}_{R,d}^T \in \mathbb{R}^{2p \times D}$ is a weight matrix and $\mathbf{b}_{R,d} \in \mathbb{R}^D$ is a bias term.

From the above model description, in TextHRNN, we can find that the major computation consumption is focused on the Bi-LSTM layer and the fully connected layer. In particular, the Bi-LSTM layer is a process of the cross product of input matrixes, so the complexity is proportional to the square of the sequence length, that is, $O(n_3^2)$ at the word level and $O(n_2^2)$ at the sentence level, while, for the fully connected layer, we mainly focus on reshaping the input matrix, resulting in a complexity proportional to the sequence length, that is, $O(n_3)$ at the word level and $O(n_2)$ at the sentence level. Clearly, as $O(n_3^2) \gg O(n_3)$ and $O(n_2^2) \gg O(n_2)$, the consumption of the fully connected layer can be ignored. Therefore, the complexity of TextRNN and TextHRNN is $O(n_1(n_2n_3)^2)$ and $O(n_1(n_2n_3)^2(1/n_2 + 1/n_3^2))$, respectively.

3.2.3. TextHCNN. Similar to [13] (shown in Figure 4), at the word level, we first convolute the word sequence $\{\mathbf{w}_{s_i}^1, \mathbf{w}_{s_i}^2, \dots, \mathbf{w}_{s_i}^{n_3}\}$ with $i = \{1, 2, \dots, n_2\}$ using different filter operators $\mathbf{W}_h \in \mathbb{R}^{h \times D}$ with $(h = 1, \dots, m)$ to get the feature maps $\mathbf{c}_{s_i}^h \in \mathbb{R}^{n_3-h+1}$ as

$$\mathbf{c}_{s_i}^h = (c_{s_i,1}^h, c_{s_i,2}^h, \dots, c_{s_i,n_3-h+1}^h), \quad (10)$$

where m is the number of filter operators. In detail, the filter operator $\mathbf{W}_h \in \mathbb{R}^{h \times D}$ in the convolution layer is applied to a window of h words to produce a new feature $c_{s_i,j}^h$ at position j of $\mathbf{c}_{s_i}^h$. That is actually done by convoluting a window of h word embeddings $\mathbf{w}_{s_i}^{j:j+h-1}$ as

$$c_{s_i,j}^h = f(\mathbf{W}_h \cdot \mathbf{w}_{s_i}^{j:j+h-1} + b_h), \quad (11)$$

where the notation $(\cdot)$ means the dot product, f is a nonlinear function, and $b_h \in \mathbb{R}$ is a bias term.

FIGURE 3: The structure of Bi-LSTM.

FIGURE 4: The structure of CNN.

After that, we employ the max-over-time pooling operation [21] to different feature maps $\mathbf{c}_{s_i}^h$ to capture the most important feature $\tilde{\mathbf{c}}_{s_i}^h$:

$$\tilde{\mathbf{c}}_{s_i}^h = \max \{ \mathbf{c}_{s_i}^h \}. \quad (12)$$

Then, after concatenating all $\tilde{\mathbf{c}}_{s_i}^h$ with $h = \{1, 2, \dots, m\}$ as

$$\mathbf{c}_{s_i} = (\tilde{\mathbf{c}}_{s_i}^1, \tilde{\mathbf{c}}_{s_i}^2, \dots, \tilde{\mathbf{c}}_{s_i}^m), \quad (13)$$

the sentence representation $\mathbf{s}_i$ can be generated by

$$\mathbf{s}_i = \mathbf{W}_{C,s}^T (\mathbf{c}_{s_i}) + \mathbf{b}_{C,s}, \quad (14)$$

where $\mathbf{W}_{C,s} \in \mathbb{R}^{m \times D}$ is a weight matrix and $\mathbf{b}_{C,s} \in \mathbb{R}^D$ is a bias term.

Similarly, at the sentence level, we convolute the sentence sequence $(\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_{n_2})$ in document d using different filter operator $\mathbf{W}_h$ with $(h = 1, \dots, m)$ to get the feature maps $\mathbf{c}_d^h$ as

$$\mathbf{c}_d^h = (c_{d,1}^h, c_{d,2}^h, \dots, c_{d,n_2-h+1}^h). \quad (15)$$

Again, we employ the max-over-time pooling operation to different feature maps $\mathbf{c}_d^h$ to get the corresponding important feature $\tilde{\mathbf{c}}_d^h$:

$$\tilde{\mathbf{c}}_d^h = \max \{ \mathbf{c}_d^h \}. \quad (16)$$

Finally, the document representation $\mathbf{d}$ can be produced as

$$\mathbf{d} = \mathbf{W}_{C,d}^T \mathbf{c}_d + \mathbf{b}_{C,d} \quad (17)$$

by concatenating all $\tilde{\mathbf{c}}_d^h$ as

$$\mathbf{c}_d = (\tilde{\mathbf{c}}_d^1, \tilde{\mathbf{c}}_d^2, \dots, \tilde{\mathbf{c}}_d^m), \quad (18)$$

TABLE 1: Complexity comparison.

Model	Complexity
FastText	$O(n)$
TextHFT	$O(n(1 + 1/n_3))$
TextRNN	$O(n_1 n_2^2 n_3^2)$
TextHRNN	$O(n_1 n_2^2 n_3^2 (1/n_2 + 1/n_3^2))$
TextCNN	$O(n)$
TextHCNN	$O(n(1 + 1/n_3))$

where $\mathbf{W}_{C,d} \in \mathbb{R}^{m \times D}$ is a weight matrix and $\mathbf{b}_{C,d} \in \mathbb{R}^D$ is a bias term.

From the above description, we can find that the main computation consumption is attributed to the convolution layer, the max-pooling layer, and the fully connected layer, which is only related to the sequence length, that is, n . Thus, similar to FastText, the complexity of TextCNN and TextHCNN is $O(n)$ and $O(n(1 + 1/n_3))$, respectively.

For clear description, we compare the complexity of discussed neural-network-based models in this paper with/without the hierarchical architecture in Table 1. Typically, as n_1, n_2 , and $n_3 \gg 1$, from Table 1, we would like to say that adding the hierarchical architecture to FastText and TextCNN makes a slight change to the complexity. However, compared to TextHRNN, a significant decrease of complexity is observed when injecting the hierarchical architecture to TextRNN.

4. Experiments

In this section, we first describe the datasets used in our experiments in Section 4.1. We then present the research questions in Section 4.2 that guide our experiments. Next we provide the details about our evaluation metrics and baselines in Section 4.3 and detail our experimental settings and parameters in Section 4.4.

4.1. Datasets. We implement our experiments on two large-scale public datasets that can be used for document representation and classification, that is, Yelp 2016 and Amazon Reviews (Electronics). The statistics of the datasets are summarized in Table 2. For each dataset, we randomly sample 80% of the data for training, 10% for validation and the remaining 10% for test.

- (i) *Yelp 2016* is obtained from the Yelp Dataset Challenge in 2016 (<https://www.yelp.com/dataset/challenge>), which has five levels of ratings from 1 to 5. In other words, we can classify the documents into five classes.
- (ii) *Amazon Reviews (Electronics)* are obtained from Amazon products data (<http://jmcauley.ucsd.edu/data/amazon/>). This dataset contains the product reviews and the metadata from Amazon from May 1996 to July 2014. Similarly, five levels of ratings from 1 to 5 are given to product reviews.

As shown in Table 2, the most notable differences between Yelp 2016 and Amazon Reviews (Electronics) lie in the

number of documents and the size of vocabulary, which could have an impact on the performance of text classification.

4.2. Research Questions. The research questions guiding our experiments are listed as follows. (1) Compared to the traditional text representation models, does the hierarchical architecture help to better represent documents? That is, can the neural-network models improve the document classification accuracy after being injected into the hierarchical architecture? (2) How does the number of sentences affect the classification performance of the proposed models with hierarchical architecture in terms of accuracy? (3) How does the document length affect the classification performance of the proposed models with hierarchical architecture in terms of accuracy?

Answers to these two questions would provide valuable insights into the utility of hierarchical architecture in neural-network-based models for document representation and classification.

4.3. Models and Metrics. The typical neural-network-based models for document classification, for example, FastText [12], TextRNN [13], and TextCNN [14], are taken into account as baselines in this paper. Correspondingly, we inject the hierarchical architecture to these baselines, leading to TextHFT, TextHRNN, and TextHCNN, respectively.

For evaluation, we use accuracy and time consumption as the metrics, where accuracy is a standard metric to measure the overall document classification performance and time consumption reflects the relative time needed for model training. In detail, the metric accuracy can be computed as

$$\text{accuracy} = \frac{\sum_{i=1}^k \text{sgn}(\text{predict}(i), \text{ground_truth}(i))}{k}, \quad (19)$$

where k is the total number of test documents, $\text{sgn}(a, b)$ is a sign function ($\text{sgn}(a, b) = 1$ when a equals b ; otherwise, $\text{sgn}(a, b) = 0$), $\text{ground_truth}(i)$ indicates the ground truth of the class label for document i , and $\text{predict}(i)$ returns the predicted class label for document i by

$$\text{predict}(i) = \text{argmax}(\mathbf{q}), \quad (20)$$

where $\text{argmax}(\mathbf{q})$ returns the class label of the maximal element in $\mathbf{q}$ that is a predictive probability distribution of a document over all classes (see Section 3.1).

4.4. Experimental Setup. For data processing, in order to produce the hierarchical architecture, we split the documents into sentences and tokenize each sentence using the Stanford's CoreNLP [22]. Besides, we discard the words with single characters and other punctuation. We randomly generate the word embedding matrix $\mathbf{W}_e$ which will be updated according to a stochastic gradient descent process, where we set the embedding dimension to 200 [23]. For initializing the neural networks, we adopt the Xavier initialization approach to keep the scale of the gradients roughly the same in all layers [24]. We use the cross entropy function as the loss function and set the batch size to be 30 (i.e., 30 documents) [1]. Gradient clipping is adopted by scaling gradients when the

TABLE 2: Statistics of datasets (the vocabulary in datasets has gone through data cleaning, excluding single characters and punctuations as well as retaining only the lemmatized words).

Dataset	Yelp 2016	Amazon Reviews (Electronics)
# classes	5	5
# documents	4,153,150	1,689,188
# average sentences/document	8.11	6.88
# average words/sentence	17.02	7.65
# average words/document	138.02	136.97
# maximal sentences in document	166	416
# maximal words in document	1,431	7,488
# words in vocabulary	155,498	66,551

TABLE 3: Performance comparison of all models.

Model	Yelp 2016		Amazon Reviews (Electronics)	
	Accuracy	Time consumption	Accuracy	Time consumption
FastText	0.5431	1.50	0.5645	0.73
TextHFT	0.5927	1.52	0.6105	0.76
TextRNN	0.4433	263.85	0.5636	106.90
TextHRN	0.5988	91.75	0.5978	24.46
TextCNN	0.5939	101.94	0.5619	27.85
TextHCNN	0.6215	102.41	0.6032	28.64

norm exceeds a threshold of 5 [25]. In addition, we use the stochastic gradient descent approach to train all models with learning rate 0.001 [23]. In order to overcome the problem of overfitting, we set the number of batches to batch = 10000.

In addition, for *TextHFT*, in the hidden layer, we employ the mean layer to average all word embeddings. For *TextHRNN*, the number of neural cells is set to 80 (80 LSTM cells in one layer) and 3 layers are deployed. In order to accelerate deep networks training, we adopt the batch normalization [26] in the model training process. For *TextHCNN*, the window size of h words in filter $\mathbf{W}_h$ is designed to $h = \{1, 2, \dots, 7\}$ so as to fully take the word orders into consideration. Besides, we set the dropout rate in the dropout layer of our *TextHRNN* and *TextHCNN* models to 0.5 [27].

5. Results and Discussions

In Section 5.1, we examine the performance of our proposed models with hierarchical architecture on public datasets. Then, Section 5.3 zooms in on the effect on document classification by varying the document length.

5.1. Performance Comparison. To answer RQ1, in Table 3, we present the experimental results of all discussed neural-network-based models in this paper for document classification on Yelp 2016 and Amazon Reviews (Electronics), respectively.

Clearly, as shown in Table 3, on the Yelp 2016 dataset, our models with hierarchical architecture, that is, TextHFT, TextHRNN, and TextHCNN, obviously outperform the corresponding neural-network-based models, that is, FastText, TextRNN, and TextCNN, in terms of accuracy. In particular, TextHFT presents a modest improvement of 9.13% against

FastText. TextHRNN shows a significant improvement of 35.08% against TextRNN. TextHCNN shows an improvement of 4.65% against TextCNN. It means that, compared to FastText and TextCNN, TextRNN receives the greatest benefits from hierarchical architecture as a substantial improvement in terms of accuracy is observed by comparing TextHRNN against TextRNN. For complexity, compared to FastText, TextHFT presents competitive time consumption. Similar findings can be observed by comparing TextHCNN against TextCNN in terms of time consumption. One particularly interesting point is that TextHRNN shows a straight decrease in terms of time consumption when comparing to TextRNN, accounting for one-third of the relative time consumption of TextRNN. It could be explained by the fact that a sequential network, for example, RNN, favors a sequential input, which is optimized by the hierarchical architecture. These findings are consistent with the theoretical analysis on complexity in Section 3.2.

Similar findings can be observed on the Amazon Reviews (Electronics) dataset. In terms of accuracy, our proposals with hierarchical architecture, that is, TextHFT, TextHRNN, and TextHCNN, result in an improvement of 8.15%, 6.07%, and 7.35%, against the corresponding FastText, TextRNN, and TextCNN, respectively. In terms of time consumption, again, no obvious differences are observed when comparing TextHFT against FastText and TextHCNN against TextCNN. A slightly different finding is that TextHRNN shows one-fourth of time consumption of TextRNN. Furthermore, we compare the results on different datasets produced by the same model. No obvious difference in terms of accuracy can be found. However, in terms of time consumption, we find that a dramatic drop is observed when comparing the result on the Amazon Reviews (Electronics) dataset against

FIGURE 5: Classification accuracy on documents with various sentence numbers.

that of the Yelp 2016 dataset. It can be attributed to the fact that the Amazon Reviews (Electronics) dataset has a larger vocabulary than the Yelp 2016 dataset and a larger vocabulary will lead to a higher complexity.

The outcomes of the main comparisons of our proposals against the baselines on the Yelp 2016 dataset and the Amazon Reviews (Electronics) dataset demonstrate that the hierarchical architecture does help to represent the document when being injected into neural-network-based models, which results in a better performance in terms of accuracy for document classification at a comparable (or substantially less) expense in terms of time consumption.

5.2. Impact of the Number of Sentences. To answer RQ2, we manually group the documents according to the number of sentences, for example, (0, 5], (5, 10], ..., (25, 30], and (30, +∞), and then examine the performance of our proposals as well as the baselines on groups of documents with various numbers of sentences. We plot the result in Figures 5(a) and 5(b) for Yelp 2016 and Amazon Reviews (Electronics), respectively. As shown in Figure 5(a), as the number of sentences increases, the performance of all discussed models decreases. It indicates that the number of sentences is an important factor that influences the classification performance. However, when the number of sentences increases, the margins between the original models and their corresponding hierarchical proposals are enlarged. The similar findings can be found on Amazon Reviews (Electronics). However, the gaps between our hierarchical proposals and their corresponding original models go up when the number of sentences increases. The aforementioned findings may be explained by the fact that the hierarchical architecture can alleviate the impact on the classification accuracy brought by the number of sentences. In other words, compared to the short documents, the long documents may benefit more from the hierarchical architecture. This leads us to investigate RQ3.

5.3. Impact of the Document Length. To answer RQ3, we manually group the documents according to their

length, for example, (0, 100], (100, 200], ..., (900, 1000], and (1000, +∞), and then examine the performance of our proposals as well as the baselines on groups of documents with various lengths. We plot the results in Figures 6(a) and 6(b) for Yelp 2016 and Amazon Reviews (Electronics), respectively.

Clearly, on the Yelp 2016 dataset, in general, the hierarchical neural models, that is, TextHFT, TextHRNN and TextHCNN, invariantly outperform the corresponding models, that is, FastText, TextRNN, and TextCNN, at all lengths. In particular, for FastText and TextHFT, as the document length increases, the accuracy slightly declines until (600–700]. After that, a fluctuation is observed. Generally, from Figure 6(a), the relative improvement of TextHFT against FastText stays stable when the document length increases. For TextRNN and TextHRNN, a similar phenomenon can be found that the accuracy declines when the document length goes up. However, the relative improvement of TextHRNN against TextRNN is enlarged as the accuracy gap between TextHRNN and TextRNN becomes larger when the document length goes up. Differently, the accuracy of TextCNN and TextHCNN increases first until (200–300] and then goes down monotonously. The relative improvement of TextHCNN against TextCNN similarly goes up when the document length increases.

On the Amazon Reviews (Electronics) dataset, interestingly, all discussed models present their peak performance at the point of document length (100–200] and then show a decrease in terms of accuracy. However, the relative improvements of our proposals with hierarchical architecture against their corresponding neural-network models, that is, TextHFT versus FastText, TextHRNN versus TextRNN, and TextHCNN versus TextCNN, share the same rhythm with that observed on the Yelp 2016 dataset when the document length increases, that is, the relative improvement of TextHFT against FastText stays stable and that of TextHRNN against TextRNN (TextHCNN against TextCNN) goes up when the document length increases. It indicates that, compared to the short documents, the long documents benefit more from the

FIGURE 6: Classification accuracy on documents with various length.

hierarchical architecture, which is similar to the analysis in Section 5.2.

Interestingly, when the document length increases, the accuracy of models on Yelp 2016 does not vary as much as that on Amazon Reviews (Electronics). This difference may originate from the difference of # average words/sentence (see Table 2). Since the # average words/sentence of Yelp 2016 is nearly 3 times that of Amazon Reviews (Electronics), it means that if the document length increases at the same interval, the increased number of sentences on Yelp is far less than that on Amazon Review (Electronics). In addition, according to the findings in Section 5.2, when the number of sentences increases, the accuracy of all discussed models will decline. Thus, it may lead to the consequence that the accuracy does not vary too much as the document length increases over the Yelp 2016 dataset, while the benefit diminishes a lot over the Amazon Reviews dataset.

6. Conclusion

In this paper, we propose a general framework for document representation with a hierarchical neural architecture, which takes the text generation frame into consideration to better improve the interoperability for different tasks. In detail, we incorporate the hierarchical neural architecture into three traditional neural-network methods, that is, FastText, TextRNN, and TextCNN, leading to the new proposals, that is, TextHFT, TextHRNN, and TextHCNN, respectively.

Our experimental results on two public datasets, that is, Yelp 2016 and Amazon Reviews (Electronics), demonstrate that our new proposals significantly outperform the corresponding neural-network-based models without hierarchical architecture for document classification. In detail, our newly proposed models present a significant improvement ranging from 4.65% to 35.08% in terms of accuracy at a comparable (or substantially less) expense in terms of time consumption. In addition, we conclude that the long documents benefit

more from the hierarchical architecture than the short ones as the improvement in terms of accuracy on long documents is higher than that on short documents.

Disclosure

Jianming Zheng and Yupu Guo are co-first authors of this article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

- [1] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 1480–1489, Human Language Technologies, June 2016.
- [2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of International Conference on Learning Representations*, 2013.
- [3] J. Pennington, R. Socher, and C. D. Manning, "GloVe: global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1532–1543, 2014.
- [4] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of Advances in Neural Information Processing Systems 26*, pp. 3111–3119, 2013.
- [5] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning, ICML 2014*, pp. 2931–2939, chn, June 2014.
- [6] Z. Zhao, T. Liu, X. Hou, B. Li, and X. Du, "Distributed text representation with weighting scheme guidance for sentiment

- analysis,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 9931, pp. 41–52, 2016.
- [7] D. Tang, “Sentiment-specific representation learning for document-level sentiment analysis,” in *Proceedings of the 8th ACM International Conference on Web Search and Data Mining, WSDM 2015*, pp. 447–451, China, February 2015.
 - [8] C. L. Isbell, “Sparse multi-level representations for retrieval,” *Journal of Computing Information Science in Engineering*, vol. 8, no. 3, pp. 603–616, 1998.
 - [9] M. Wang, M. Liu, S. Feng, D. Wang, and Y. Zhang, “A novel calibrated label ranking based method for multiple emotions detection in chinese microblogs,” in *Natural Language Processing and Chinese Computing*, pp. 238–250, Heidelberg, Germany, 2014.
 - [10] T. Joachims, “Text categorization with support vector machines: learning with many relevant features,” in *Proceedings of the of European Conference on Machine Learning*, pp. 137–142, 1398.
 - [11] X. Zhang, J. Zhao, and Y. Lecun, “Character-level convolutional networks for text classification,” in *Proceedings of the 29th Annual Conference on Neural Information Processing Systems*, pp. 649–657, 2015.
 - [12] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, “Bag of tricks for efficient text classification,” in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017*, pp. 427–431, 2017.
 - [13] Y. Kim, “Convolutional neural networks for sentence classification,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1746–1751, 2014.
 - [14] P. Liu, X. Qiu, and V. Huang, “Recurrent neural network for text classification with multi-task learning,” in *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pp. 2873–2879, 2016.
 - [15] S. Lai, L. Xu, K. Liu, and Z. Jun, “Recurrent convolutional neural networks for text classification,” in *Proceedings of the In Proceedings of Association for the Advancement of Artificial Intelligence*, pp. 2267–2273, 2015.
 - [16] R. Xu, T. Chen, Y. Xia, Q. Lu, B. Liu, and X. Wang, “Word embedding composition for data imbalances in sentiment and emotion classification,” *Cognitive Computation*, vol. 7, no. 2, pp. 226–240, 2015.
 - [17] Y.-W. Chen, Q. Zhou, W. Luo, and J.-X. Du, “Classification of chinese texts based on recognition of semantic topics,” *Cognitive Computation*, vol. 8, no. 1, pp. 114–124, 2016.
 - [18] D. D. Lewis, “Evaluation of phrasal and clustered representations on a text categorization task,” in *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 37–50, 1992.
 - [19] M. Post and S. Bergsma, “Explicit and implicit syntactic features for text classification,” in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pp. 866–872, 2013.
 - [20] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model,” *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.
 - [21] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural language processing (almost) from scratch,” *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.
 - [22] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, “The stanford corenlp natural language processing toolkit,” in *Proceedings of the Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55–60, 2014.
 - [23] S. Lai, K. Liu, S. He, and J. Zhao, “How to generate a good word embedding,” *IEEE Intelligent Systems*, vol. 31, no. 6, pp. 5–14, 2016.
 - [24] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.
 - [25] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” *Computer Science*, vol. 52, no. 3, 2012.
 - [26] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proceedings of the 32nd International Conference on Machine Learning (ICML ’15)*, pp. 448–456, 2015.
 - [27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.