Cognitive Computation (2019) 11:317–327
https://doi.org/10.1007/s12559-018-9621-6

Hierarchical Neural Representation for Document Classification

Jianming Zheng[1] · Fei Cai[1] · Wanyu Chen[1] · Chong Feng[1] · Honghui Chen[1]

Received: 30 October 2017 / Accepted: 11 December 2018 / Published online: 16 January 2019

Abstract

Text representation, which converts text spans into real-valued vectors or matrices, is a crucial tool for machines to understand the semantics of text. Although most previous works employed classic methods based on statistics and neural networks, such methods might suffer from data sparsity and insensitivity to the text structure, respectively. To address the above drawbacks, we propose a general and structure-sensitive framework, i.e., the hierarchical architecture. Specifically, we incorporate the hierarchical architecture into three existing neural network models for document representation, thereby producing three new representation models for document classification, i.e., TextHFT, TextHRNN, and TextHCNN. Our comprehensive experimental results on two public datasets demonstrate the effectiveness of the hierarchical architecture. With a comparable (or substantially less) time expense, our proposals obtain significant improvements ranging from 4.65 to 35.08% in terms of accuracy against the baseline. We can conclude that the hierarchical architecture can enhance the classification performance. In addition, we find that the benefits provided by the hierarchical architecture can be strengthened as the document length increases.

Keywords Document representation · Neural networks · Hierarchical architecture · Document classification

Introduction

Text representation is a challenging task in natural language processing (NLP). This task converts text spans into real-valued vectors or matrices, which serve as a crucial tool for machines to understand the semantics of text. From the text generation framework (words from a phrase or a sentence and sentences from a document [27]), text representation can be divided into the following levels: word-level representation (e.g., word2vec [21] and GloVe [23]), sentence-level representation [17], and document-level representation [29]. In this paper, we focus on document-level representation, which has broad applications, e.g., sentiment classification [25], text retrieval [9], text ranking [26].

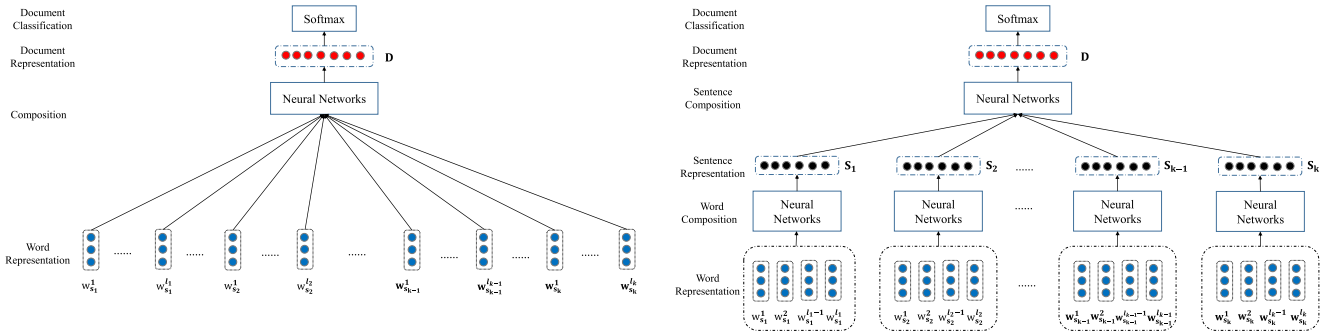
The most common approaches for document representation are bag-of-words (BoW) [11] and *n*-grams with term frequency-inverse document frequency (TF-IDF) [28]. However, such statistics-based methods might cause data sparsity and dimensional explosion when they are applied on a large-scale corpus. Recently, numerous approaches based on different neural network architectures have been proposed, e.g., FastText (based on a hidden layer) [12], TextCNN (based on convolutional neural networks) [14], TextRNN (based on recurrent neural networks) [19], and TextHAN (based on hierarchical attention networks) [27]. Such neural network–based models can generate low-dimensional vectors to represent a document, overcoming the aforementioned problems. In addition, compared to the statistics-based approaches, the neural network–based models perform better in capturing the semantic relationships between words [12].

However, the existing neural network–based document representation (see Fig. 1a) completely depends on a single neural network, ignoring the internal structure features of a document itself, e.g., the word-sentence and sentence-

Jianming Zheng and Wanyu Chen are co-first authors of this article.

✉ Fei Cai
caifei@nudt.edu.cn

[1] Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, China



(a) Existing architecture for document classification (b) Hierarchical architecture for document classification

Fig. 1 Comparison of existing and hierarchical architectures

document relationships. We argue that these features can be regarded as prior knowledge for generating a better document representation.

Hence, in this paper, inspired by Kia et al. [13], Li et al. [18], and Chen et al. [3], we propose a general and structure-sensitive framework (see Fig. 1b) for document representation by injecting the hierarchical architecture into neural networks. As shown in Fig. 1b, our proposals mainly consist of two parts, i.e., the word level and the sentence level. At the word level, each sentence is represented by utilizing a specific neural network architecture to aggregate the embeddings of words in the sentence. Similarly, at the sentence level, the document is represented by aggregating all the sentences generated from the former step. We implement our proposals on public large-scale benchmark datasets for document classification. Our experimental results indicate that the proposed hierarchical architecture does help to improve the performance after being incorporated into the existing neural network baselines, e.g., FastText [12], TextCNN [14], TextRNN [19], and TextHAN [27].

Our major contributions are summarized as follows:

(1) We have addressed the challenge of document representation for classification by incorporating the hierarchical architecture into neural networks.
(2) The computational complexities of our proposals are comparable to (or substantially less than) those of their corresponding baselines.
(3) Our proposals significantly outperform the corresponding state-of-the-art baselines with an improvement of approximately 8% in terms of accuracy.

The remainder of this paper is organized as follows. We describe the related work in the "Related Work" section. Our proposals are described in the "Methods" section. The "Experiments" section presents our experimental setup. In the "Results" section, we report and discuss our experimental results. Finally, we conclude in the "Conclusions" section.

Related Work

In this section, we will briefly summarize the approaches for document classification based on various document representation schemes, i.e., the traditional statistics-based representation (see "Statistical Representation–Based Document Classification") and the neural network–based representation (see "Neural Representation–Based Document Classification"). In addition, we will present the major differences between our proposals and previous works.

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 a vocabulary into a sparse vector with a single high value (i.e., 1) in its position and all the others with a low value (i.e., 0). The representation is employed in the BoW model [11] to reflect the word frequency information. However, the BoW model only symbolizes the words and cannot reflect the semantic relationships between words. Consequently, the bag-of-means model [28] was proposed to cluster the word embeddings learned by the word2vec model [21]. Furthermore, the bag-of-*n*-grams [28] was developed to take the *n*-grams order into account for document representation, which selected the most frequent n-grams (up to 5-grams) as the vocabulary in the BoW model. In addition, with some extra statistical information, e.g., TF-IDF, a better document representation can be achieved [3]. For instance, Al-Radaideh and Bataineh [1]

incorporated domain knowledge, statistical features and genetic algorithms to extract important points of documents. Henao et al. [7] levered the label-word information and proposed to view text classification as a label-word joint embedding problem.

Clearly, a progressive step has been made to document classification based on statistical representation. However, such traditional statistical representation approaches inevitably face the problems of data sparsity and dimensionality, leading to no applications on large-scale corpora. In addition, such approaches are simply built on shallow statistics, and deeper semantic information of a document has not been well developed. Instead, our proposal in this paper based on neural networks has the ability to learn the low-dimensional and distributed representation to overcome such problems.

Neural Representation–Based Document Classification

Since Bengio et al. [2] first employed the neural network architecture to train a language model, considerable attention has been devoted to proposing neural network–related models for document classification. For instance, the FastText model proposed by Joulin et al. [12] employs one hidden layer to integrate all input information and obtains satisfactory results. However, this model simply averages all word embeddings and discards the information of word order. To overcome the problem of insufficient training data that often appears in single-task supervised learning, Liu et al. [19] proposed the multitask learning framework with recurrent neural networks (RNNs) to jointly learn across multiple related tasks. Compared to RNNs, convolutional neural networks (CNNs) are easier to train and capture the global document information. For instance, Kim [14] adopted CNNs to classify documents, while Zhang et al. [28] at the character level also employed a CNN to represent documents. Furthermore, a combination of these neural network models can integrate the advantage of each single neural network. For instance, Lai et al. [15] proposed recurrent convolutional neural networks (RCNNs), which adopted the recurrent structure to grasp the context information and employed a max-pooling layer to identify the key components in a document. Besides, other document features have been injected in the document modeling [10]. For instance, He et al. [6] transformed the document-level knowledge to improve the performance of aspect-level sentiment classification.

Although the above neural network models are able to mine the hidden document features by training complex neural networks, such models built on one or multiple tasks lack interoperability. In other words, they cannot be simply transferred to other tasks and obtain satisfactory results. In addition, such approaches directly employ the neural network architecture to obtain the document representation without considering the structure features of the document, thus making the models less structure sensitive. Instead, our proposals with the hierarchical architecture concentrate more on the process of generating document representation, thereby improving the interpretability and structure sensitivity of models.

Methods

In this section, we will first formally describe our proposals, i.e., the general frameworks of the hierarchical architecture in the "General Framework" section. Then, in the "Hierarchical Neural Representation" section, we will detail three new proposed models incorporated with the hierarchical architecture, i.e., TextHFT, TextHRNN, and TextHCNN, which basically originate from the corresponding models, i.e., FastText, TextRNN, and TextCNN, respectively. In addition, a comprehensive computational complexity analysis will be conducted on all discussed models.

General Framework

First, we propose a general framework for document representation with the hierarchical neural architecture. Let us provide a brief illustration. For simplicity, suppose that we have n_1 documents in the corpus, where each document has the same length n_2 and each sentence has the same length n_3. Given a document d, we denote the document as $d = \{s_1, s_2, \cdots, s_{n_2}\}$ and the sentence as $s_i = \{w_{s_i}^1, w_{s_i}^2, \cdots, w_{s_i}^{n_3}\}$, where $w_{s_i}^j$ represents the jth word in sentence s_i. We summarize these notations in Table 1.

Next, as illustrated in Fig. 1b, our proposed *hierarchical architecture* for document representation mainly consists of six processes. We start from the word representation module, which converts each word into a specific low-dimensional vector. Then, for each sentence, the word

Table 1 Variable descriptions

Variable	Description
n_2	The number of sentences in document d
n_3	The number of words in sentence
s_i	The ith sentence in document d
$w_{s_i}^j$	The jth word in sentence s_i
$\mathbf{s_i}$	The representation of sentence s_i
$\mathbf{w_{s_i}^j}$	The representation of word $w_{s_i}^j$
$\mathbf{d}$	The representation of document d

combination module utilizes specific neural networks to aggregate the embeddings of all words in the sentence, which produces the sentence representation. Subsequently, the sentence composition module similarly aggregates the representations of all sentences in the document, which produces the document representation. Finally, the document classification task is implemented to verify the effectiveness of our proposed models. We briefly detail the six modules as pseudocode.

Algorithm 1 Hierarchical architecture for document classification.

Input: The embedding matrix for each word in a vocabulary, W_e; the sentence sequence in document d, $d = \{s_1, s_2, \cdots, s_{n_2}\}$; the word sequence in each sentence, e.g., $s_i = \{w_{s_i}^1, w_{s_i}^2, \cdots, w_{s_i}^{n_3}\}$.

Output: The class label of document d.

1: Initialize the network N_s at the sentence level and N_w at the word level. Initialize the matrix W and bias term b in softmax classifier.
2: $i = 0$
3: **while** $i < n_2$ **do**
4: $\quad j = 0$
5: $\quad$ **while** $j < n_3$ **do**
6: $\quad\quad$ Look up the embedding matrix W_e to obtain the embedding of word $w_{s_i}^j$: $\mathbf{w_{s_i}^j}$.
7: $\quad\quad j = j + 1$
8: $\quad$ **end while**
9: $\quad$ Obtain the representation $\mathbf{s_i}$ of sentence s_i with the input $\{\mathbf{w_{s_i}^1}, \mathbf{w_{s_i}^2}, \cdots, \mathbf{w_{s_i}^{n_3}}\}$ in the network N_w.
10: $\quad i + 1$
11: **end while**
12: Obtain the representation $\mathbf{d}$ of document d with the input $\{\mathbf{s_1}, \mathbf{s_2}, \cdots, \mathbf{s_{n_2}}\}$ in the network N_s.
13: Employ softmax classifier on document representation $\mathbf{d}$, i.e., $p = softmax(W\mathbf{d} + b)$.
14: **return** The position of the max value in p.

Algorithm 1 describes the entire process of the hierarchical architecture for document classification.

Hierarchical Neural Representation

In this section, we will present our proposed document representation models with the hierarchical architecture (i.e., TextHFT, TextHRNN, and TextHCNN) and perform a detailed analysis of their complexities.

TextHFT

The key component of FastText integrates all word representations on a hidden layer. In contrast to the existing FastText that directly averages all word embeddings of the document, TextHFT first averages all word embeddings of a sentence to obtain the sentence representation and then averages all sentence representations to obtain the document representation. Thus, we can obtain

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

at the word level.

At the sentence level, we can then represent a document as

$$\mathbf{d} = \frac{1}{n_2}\sum_{i=1}^{n_2}\mathbf{s}_i \ (\mathbf{d} \in \mathbb{R}^D). \tag{2}$$

Thus, TextHFT is completed.

From the above analysis, we find that the complexity of the hierarchical architecture is mainly related to the sequence length. In particular, the complexity of 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 $O(n_1 \times n_2 \times n_3)$, i.e., $O(n)$, and that at the sentence level is $O(n_1 \times n_2)$. In total, the complexity of TextHFT is $O(n(1 + 1/n_3))$.

TextHRNN

To overcome the problems of gradient disappearance and of context scarcity, we implement the bidirectional long short-term memory RNN (Bi-LSTM) model on the text sequence.

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

$$\overrightarrow{h_t^{s_i}} = \overrightarrow{\text{LSTM}}(\mathbf{w}_{s_i}^t, \overrightarrow{h_{t-1}^{s_i}})(\overrightarrow{h_t^{s_i}} \in \mathbb{R}^p)$$

and

$$\overleftarrow{h_t^{s_i}} = \overleftarrow{\text{LSTM}}(\mathbf{w}_{s_i}^t, \overleftarrow{h_{t+1}^{s_i}})(\overleftarrow{h_t^{s_i}} \in \mathbb{R}^p),$$

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

$$\mathbf{h}_t^{s_i} = (\overrightarrow{h_t^{s_i}}; \overleftarrow{h_t^{s_i}}), (\mathbf{h}_t^{s_i} \in \mathbb{R}^{2p}, t = 1, 2, \cdots, n_3). \tag{3}$$

Then, 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

$$\mathbf{s}_i = \mathbf{W}_{R,s}^T\mathbf{h}_{n_3}^{s_i} + \mathbf{b}_{R,s}, \tag{4}$$

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 employing Bi-LSTM to the sentence sequence $\{\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_{n_2}\}$, we will produce

$$\overrightarrow{h_t^d} = \overrightarrow{\text{LSTM}}(\mathbf{s}_t, \overrightarrow{h_{t-1}^{s_i}})(\overrightarrow{h_t^d} \in \mathbb{R}^{2p})$$

and

$$\overleftarrow{h_t^d} = \overleftarrow{\text{LSTM}}(\mathbf{s}_t, \overleftarrow{h_{t+1}^{s_i}})(\overleftarrow{h_t^d} \in \mathbb{R}^{2p}),$$

where $t = \{1, 2, \cdots, n_2\}$. We again concatenate each $\overrightarrow{h_t^d}$ with $\overleftarrow{h_t^d}$ to obtain a hidden output $\mathbf{h}_t^d$ as

$$\mathbf{h}_t^d = (\overrightarrow{h_t^d}; \overleftarrow{h_t^d}), (\mathbf{h}_t^d \in \mathbb{R}^{2p}). \tag{5}$$

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

$$\mathbf{d} = \mathbf{W}_{R,d}^T \mathbf{h}_{\mathbf{n_2}}^{\mathbf{d}} + \mathbf{b}_{R,d}, \tag{6}$$

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

From the model description of TextHRNN, we find that the major computational cost 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 matrices; thus, the complexity is proportional to the square of the sequence length, i.e., $O(n_3^2)$ for each sentence at the word level and $O(n_2^2)$ for each document at the sentence level. For the fully connected layer, this process focuses on reshaping the input matrix, resulting in the complexity being proportional to the sequence length, i.e., $O(n_3)$ for each sentence at the word level and $O(n_2)$ for each document at the sentence level. Because $O(n_3^2) \gg O(n_3)$ and $O(n_2^2) \gg O(n_2)$, to simplify the problem description, the computational cost of the fully connected layer can be ignored. Therefore, the complexities of TextRNN and TextHRNN are $O(n_1 n_2^2 n_3^2)$ and $O(n_1 n_2^2 n_3^2 (1/n_2 + 1/{n_3}^2))$, respectively.

TextHCNN

At the word level, we first convolute the word sequence $\{\mathbf{w}_{s_i}^1, \mathbf{w}_{s_i}^2, \cdots, \mathbf{w}_{s_i}^{n_3}\}$ with $i = \{1, 2, \cdots, n_2\}$ using different filter operators $\mathbf{W}_h \in \mathbb{R}^{h \times D}$ with $(h = 1, \cdots, m)$ to obtain the feature maps $\mathbf{c}_{s_i}^h \in \mathbb{R}^{n_3 - h + 1}$ as

$$\mathbf{c}_{\mathbf{s_i}}^{\mathbf{h}} = (c_{s_i,1}^h, c_{s_i,2}^h, \cdots, c_{s_i,n_3-h+1}^h), \tag{7}$$

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

$$c_{s_i,j}^h = f(\mathbf{W}_h \circ \mathbf{w}_{s_i}^{j:j+h-1} + b_h), \tag{8}$$

where the notation $\circ$ means elementwise multiplication, f is a nonlinear function and $b_h \in \mathbb{R}$ is a bias term.

Subsequently, we employ the max-over-time pooling operation [4] to different feature maps $\mathbf{c}_{s_i}^h$ to capture the most important feature $\tilde{c}_{s_i}^h$:

$$\tilde{c}_{s_i}^h = \max\{\mathbf{c}_{s_i}^h\}. \tag{9}$$

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

$$\mathbf{c}_{s_i} = (\tilde{c}_{s_i}^1, \tilde{c}_{s_i}^2, \ldots, \tilde{c}_{s_i}^m). \tag{10}$$

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

$$\mathbf{s}_i = \mathbf{W}_{C,s}^T(\mathbf{c}_{s_i}) + \mathbf{b}_{C,s}, \tag{11}$$

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 $(\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_{n_2})$ in the document d using different filter operators $\mathbf{W}_h$ with $(h = 1, \cdots, m)$ to obtain the feature maps $\mathbf{c}_d^h$ as

$$\mathbf{c}_d^h = (c_{d,1}^h, c_{d,2}^h, \cdots, c_{d,n_2-h+1}^h). \tag{12}$$

We again employ the max-over-time pooling operation to different feature maps $\mathbf{c}_d^h$ to obtain the corresponding important feature $\tilde{c}_d^h$:

$$\tilde{c}_d^h = \max\{\mathbf{c}_d^h\}. \tag{13}$$

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

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

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

$$\mathbf{c}_d = (\tilde{c}_d^1, \tilde{c}_d^2, \ldots, \tilde{c}_d^m), \tag{14}$$

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. Thus, TextHCNN is completed.

From the above description, we find that the main computational cost is attributed to the convolutional layer, the max-pooling layer and the fully connected layer. Moreover, the respective complexities of these three layers are only related to the sequence length. Thus, similar to FastText, the complexities of TextCNN and TextHCNN are $O(n)$ and $O(n(1 + 1/n_3))$, respectively.

For a clear description, we compare the complexities of all the neural network models discussed in this paper with and without the hierarchical architecture in Table 2. Typically, since n_1, n_2 and $n_3 \gg 1$, then $1/n_2, 1/n_3$ and $1/n_3^2 \ll 1$. Hence, we can conclude that adding the hierarchical architecture to FastText and TextCNN causes a slight change in their complexities. However, compared to TextHRNN, a significant decrease in complexity is observed when injecting the hierarchical architecture into TextRNN.

Table 2 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	$O(n(1+1/n_3))$

Experiments

In this section, we will describe the datasets used in our experiments in the "Datasets" section. Then, we will present the research questions in the "Research Questions" section that guides our experiments. Next, we will provide the details about our evaluation metrics and baselines in the "Models and Metrics" section, and we detail our experimental settings and parameters in the "Experimental Setup" section.

Datasets

We implemented our experiments on two large-scale public datasets that could be used for document classification, i.e., Yelp 2016 and Amazon Reviews (Electronics). The statistics of these datasets are summarized in Table 3. We used the holdout cross-validation method to implement the experiments. That is, for each dataset, we randomly sampled 80% of the data for training, 10% for validation and the remaining 10% for testing.

- **Yelp 2016** is obtained from the Yelp Dataset Challenge in 2016,[1] which has five levels of ratings from 1 to 5. In other words, we can classify these documents into five classes.
- **Amazon Reviews (Electronics)** is obtained from Amazon product data.[2] This dataset contains the product reviews and 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 3, the most notable differences between Yelp 2016 and Amazon Reviews (Electronics) are in the number of documents and the size of the vocabulary, which could have an impact on the document classification performance. In Fig. 2, we also plot the length distributions of documents in Yelp 2016 and Amazon Reviews (Electronics). We find that the majority of documents in both datasets have less than 300 words.

[1] https://www.yelp.com/dataset/challenge

[2] http://jmcauley.ucsd.edu/data/amazon/

Research Questions

The research questions guiding our experiments are the following.

RQ1 Compared to the existing neural network representation models, does the hierarchical architecture help to better represent documents? In other words, can the neural network models improve classification performance after being injected with the hierarchical architecture?

RQ2 How does the document length affect the classification performance of all discussed models?

The answers to these two questions would provide valuable insights into the utility of the hierarchical architecture in the neural network models for document classification.

Models and Metrics

The typical neural-network models for document classification, e.g., FastText [12], TextRNN [19], TextCNN [14], and TextHAN [27] are used as baselines in this paper. Correspondingly, we inject the hierarchical architecture into these baselines, leading to TextHFT, TextHRNN, and TextHCNN, respectively. In this paper, we repeat all discussed models on Yelp 2016 and Amazon Reviews (Electronics).

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

$$accuracy = \frac{\sum_{i=1}^{k} Sgn(predict(i), ground\_truth(i))}{k}, \tag{15}$$

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

$$predict(i) = \arg\max p(\mathcal{Y}|\mathbf{d}), \tag{16}$$

where

$$p(\mathcal{Y}|\mathbf{d}) = \mathbf{softmax}(\mathbf{W}\mathbf{d} + \mathbf{b}). \tag{17}$$

Here, $\mathbf{d}$ is the document representation (see Algorithm 1), and W and b are the reshape matrix and the bias, respectively. For the metric *time consumption*, we regard the respective time consumptions of FastText on Yelp 2016

Table 3 Statistics of datasets (the vocabulary in the datasets has gone through data cleaning, excluding single characters and punctuation and retaining only the lemmatized words)

Dataset	Yelp 2016	Amazon Reviews (Electronics)
No. of classes	5	5
No. of documents	4,153,150	1,689,188
No. of average sentences/document	8.11	6.88
No. of average words/sentence	17.02	7.65
No. of average words/document	138.02	136.97
No. of maximum sentences in document	166	416
No. of maximum words in document	1431	7488
No. of words in vocabulary	155,498	66,551

and Amazon Reviews (Electronics) as one unit, and the time consumptions of other models are the relative time consumption against that of FastText in their corresponding dataset.

Experimental Setup

For data processing, to produce the hierarchical architecture, we split the documents into sentences and tokenized each sentence using Stanford's CoreNLP [20]. Moreover, we discarded the words with single characters and other punctuation. We randomly generated the word embedding matrix W_e, which was updated according to a stochastic gradient descent process, where we set the embedding dimension to 200 [16]. For initializing the neural networks, we adopted the Xavier initialization approach to maintain the scale of the gradients approximately the same in all layers [5]. We used the cross entropy function as the loss function and set the batch size to 30 (i.e., 30 documents) [27]. Gradient clipping was adopted by scaling gradients when the norm could exceed a threshold of 5 [22]. In addition, we used the stochastic gradient descent approach to train all models with a learning rate of 0.001 [16]. To overcome the problem of overfitting, we set the number of batches as $batch = 10000$.

In addition, for **TextHFT**, in the hidden layer, we employed the mean layer to average all word embeddings. For **TextHRNN**, the number of neural cells was set to 80 (80 LSTM cells in one layer), and 3 layers were deployed. To accelerate training of the deep networks, we adopted batch normalization [8] in the model training process. For **TextHCNN**, the window size of h words in filter $\mathbf{W_h}$ was designed as $h = \{1, 2, \cdots, 7\}$ to fully take the word orders into consideration. Moreover, we set the dropout rate in the dropout layer of our **TextHRNN** and **TextHCNN** models to 0.5 [24].

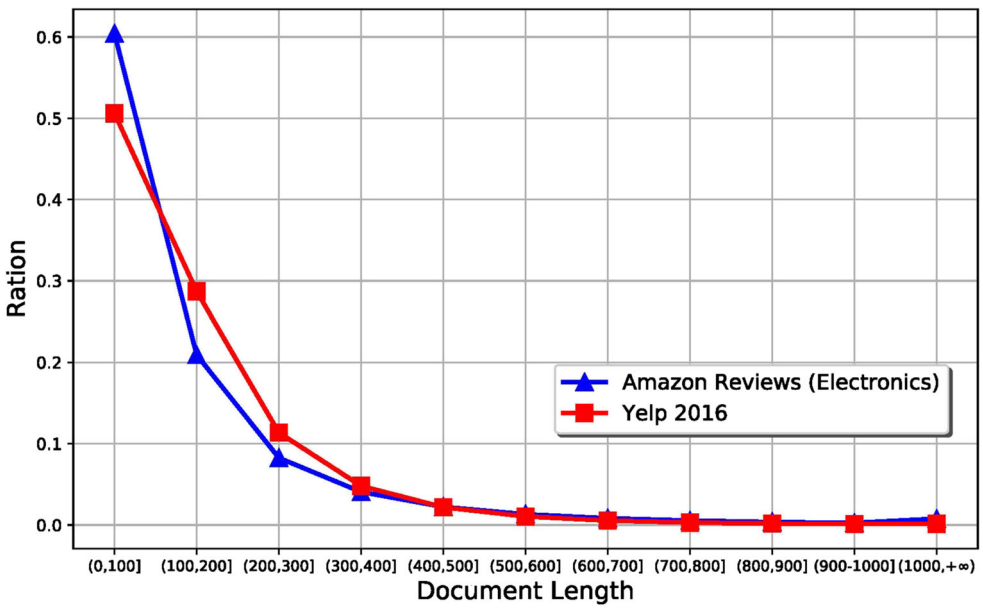


Fig. 2 Distributions of document length

Results

In the "Performance Comparison" section, we will examine the performance of our proposed models with the hierarchical architecture on public datasets. Then, the "Impact of the Document Length" section will focus on the effect on document classification by varying the document length.

Performance Comparison

To answer **RQ1**, in Table 4, we present the experimental results of all discussed models on Yelp 2016 and Amazon Reviews (Electronics).

On Yelp 2016, our models with the hierarchical architecture, i.e., TextHFT, TextHRNN, and TextHCNN, clearly outperform the corresponding baselines, i.e., FastText, TextRNN, and TextCNN, respectively, 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. This result indicates that, compared to FastText and TextCNN, TextRNN benefits the most from the hierarchical architecture. Regarding complexity, TextHFT presents a competitive time consumption compared to FastText (1.00 vs. 1.01). Similar findings can be observed by comparing TextHCNN against TextCNN in terms of time consumption (67.96 vs. 68.27). One particularly interesting result is that TextHRNN shows a linear decrease in terms of time consumption when compared to that of TextRNN, which is approximately one third of the time consumption of TextRNN.

This result could be explained as sequential neural networks, e.g., RNNs, favor sequential inputs, which can be optimized by the hierarchical architecture. These findings are consistent with the theoretical analysis of complexity in the "Hierarchical Neural Representation" section.

Similar findings can be observed on the Amazon Reviews (Electronics) dataset. In terms of accuracy, our proposals with the hierarchical architecture, i.e., TextHFT, TextHRNN, and TextHCNN, result in improvements of 8.15%, 16.60%, and 7.35% compared to the corresponding FastText, TextRNN, and TextCNN, respectively. In terms of time consumption, again, no clear differences are observed when comparing TextHFT against FastText and TextHCNN against TextCNN. However, a slightly different finding is that TextHRNN shows nearly one fourth of the time consumption of TextRNN.

Furthermore, we compare the results on different datasets produced by the same model. No clear differences in terms of accuracy can be found. However, in terms of time consumption, we observe a dramatic decrease when comparing the result on Amazon Reviews (Electronics) against that on Yelp 2016. This decrease can be attributed to the fact that Amazon Reviews (Electronics) has a larger vocabulary than that of Yelp 2016, and a larger vocabulary will lead to a higher complexity.

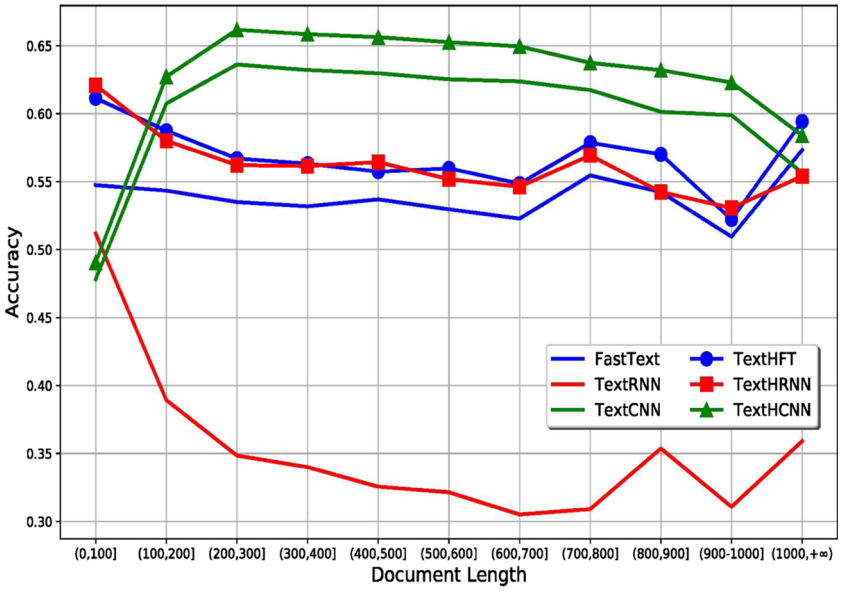
The aforementioned findings demonstrate that the hierarchical architecture does help to represent the document when being injected into neural network–based models. With a comparable (or substantially less) time expense, our proposals with the hierarchical architecture can achieve better performance in terms of accuracy for document classification.

Impact of the Document Length

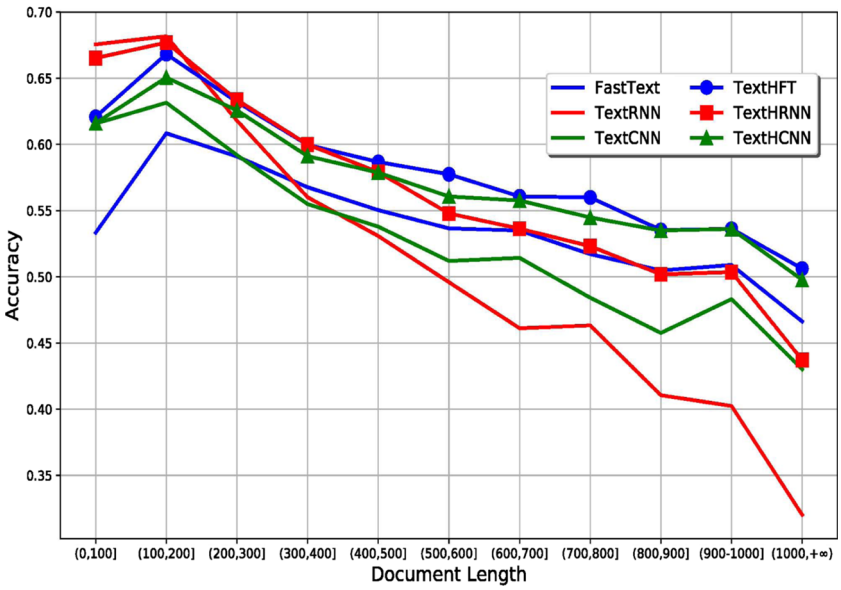
To answer **RQ2**, we manually group the documents according to their lengths, e.g., (0, 100], (100, 200], $\cdots$, (900, 1000] and (1000, $+\infty$), and then we examine the performances of our proposals and of the baselines in groups of documents with different lengths. We plot the results in Fig. 3a and b for Yelp 2016 and Amazon Reviews (Electronics), respectively.

Table 4 Performance comparison of all models (in the same neural network architecture, e.g., FastText and TextHFT, the higher accuracy and the lower time consumption are boldfaced and italicized, respectively)

	Yelp 2016		Amazon Reviews (Electronics)	
Model	Accuracy	Time consumption	Accuracy	Time consumption
FastText	0.5431	*1.00*	0.5645	*1.00*
TextRNN	0.4433	175.90	0.5127	146.44
TextCNN	0.5939	*67.96*	0.5619	*38.15*
TextHAN	0.5575	70.19	0.5493	39.30
TextHFT	**0.5927**	1.01	**0.6105**	1.04
TextHRNN	**0.5988**	*61.17*	**0.5978**	*33.51*
TextHCNN	**0.6215**	68.27	**0.6032**	39.23



(a) Yelp 2016



(b) Amazon Reviews (Electronics)

Fig. 3 Classification accuracy in documents with various lengths

Clearly, on Yelp 2016, our proposals with the hierarchical architecture, i.e., TextHFT, TextHRNN, and TextHCNN, generally outperform the corresponding models, i.e., FastText, TextRNN, and TextCNN, in each length group. Among them, as the document length increases, the accuracies of FastText and TextHFT slightly decrease until (600–700]. Subsequently, a fluctuation is observed. Generally, from Fig. 3a, the relative improvement of TextHFT against FastText remains stable as the document length increases. For TextRNN and TextHRNN, the accuracy decreases as the document length increases. However, the relative improvement of TextHRNN against TextRNN becomes larger as the document length increases. In contrast, the accuracies of TextCNN and TextHCNN increase until (200–300] and then monotonically decreases. The relative improvement of TextHCNN against TextCNN similarly increases as the document length increases.

On Amazon Reviews (Electronics), however, all discussed models first increase to their peak in the section of document length (100–200] and then show a decrease in accuracy. However, the relative improvements of our proposals against their corresponding neural network models,

i.e., TextHFT vs. FastText, TextHRNN vs. TextRNN, and TextHCNN vs. TextCNN, share the same pace with those observed on Yelp 2016 as the document length increases. That is, the relative improvement of TextHFT against FastText remains stable and those of TextHRNN against TextRNN (TextHCNN against TextCNN) increase when the document length increases.

Moreover, as the document length increases, the accuracies of all discussed models on Yelp 2016 do not vary as much as those on Amazon Reviews (Electronics). This difference may originate from the difference of no. of average words/sentence (see Table 3). The no. of average words/sentence of Yelp 2016 is nearly three times of 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 Reviews (Electronics). Additionally, the longer the sequence length is, the lower is the accuracy performance of all discussed models. Hence, compared with the decrease on Amazon Reviews (Electronics), the smaller increased sentence sequence on Yelp 2016 may present a slower decrease.

From the above observations, we can conclude that for the task of document classification based on neural network models, e.g., FastText, RNN, and CNN, the effect of the hierarchical architecture can be magnified as the document length increases. That is, the benefits resulting from the hierarchical architecture can be strengthened as the document length increases.

Conclusions

In this paper, we propose a general and structure-sensitive framework, i.e., the hierarchical architecture, which regards the text generation frame as prior knowledge to improve the interoperability for different tasks. In detail, we incorporate the hierarchical neural architecture into three existing neural network methods, i.e., FastText, TextRNN, and TextCNN, leading to our proposals, i.e., TextHFT, TextHRNN, and TextHCNN, respectively.

Our experimental results on two public datasets, i.e., Yelp 2016 and Amazon Reviews (Electronics), demonstrate that our proposals with the hierarchical architecture significantly outperform the corresponding neural network models without the hierarchical architecture for document classification. Specifically, with a comparable (or substantially less) time expense, our newly proposed models present significant improvements ranging from 4.65 to 35.08% in terms of accuracy. In addition, we conclude that the benefits resulting from the hierarchical architecture can be strengthened as the document length increases.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

Informed Consent Informed consent was not required as no human or animals were involved.

Human and Animal Rights This article does not contain any studies with human participants performed by any of the authors.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

1. Al-Radaideh QA, Bataineh DQ. A hybrid approach for arabic text summarization using domain knowledge and genetic algorithms. Cogn Comput. 2018;10(4):651–69.
2. Bengio Y, Ducharme R, Vincent P, Janvin C. A neural probabilistic language models. J Mach Learn Res. 2003;3(6):1137–55.
3. Chen YW, Zhou Q, Luo W, Du JX. Classification of chinese texts based on recognition of semantic topics. Cogn Comput. 2016;8(1):114–24.
4. Collobert R, Weston J, Karlen M, Kavukcuoglu K, Kuksa P. Natural language processing (almost) from scratch. J Mach Learn Res. 2011;12(1):2493–537.
5. Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the 13th International Conference on Artificial Intelligence and Statistics; 2010. p. 249–256.
6. He R, Lee WS, Ng HT, Dahlmeier D. Exploiting document knowledge for aspect-level sentiment classification. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics; 2018. p. 579–585.
7. Henao R, Li C, Carin L, Shen D, Wang G, Wang W, Zhang Y, Zhang X. Joint embedding of words and labels for text classification. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics; 2018. p. 2321–2331.
8. Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on Machine Learning; 2015. p. 448–456.
9. Isbell CL. Sparse multi-level representations for retrieval. J Comput Inf Sci Eng. 1998;8(3):603–16.
10. Jianming Z, Fei C, Taihua S, Honghui C. Self-interaction attention mechanism-based text representation for document classification. Appl Sci. 2018;8(4):613.
11. Joachims T. Text categorization with suport vector machines: Learning with many relevant features. In: Proceedings of European Conference on Machine Learning; 1998. p. 137–142.
12. Joulin A, Grave E, Bojanowski P, Mikolov T. Bag of tricks for efficient text classification. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics; 2017. p. 427–431.
13. Kia D, Soujanya P, Amir H, Erik C, Hawalah AYA, Alexander G, Qiang Z. Multilingual sentiment analysis: state of the art and independent comparison of techniques. Cogn Comput. 2016;8(4):1–4.

14. Kim Y. Convolutional neural networks for sentence classification. In: Proceedings of Conference on Empirical Methods in Natural Language Processing; 2014. p. 1746–1751.
15. Lai S, Xu L, Liu K, Jun Z. Recurrent convolutional neural networks for text classification. In: Proceedings of Association for the Advancement of Artificial Intelligence; 2015. p. 2267–2273.
16. Lai S, Liu K, He S, Zhao J. How to generate a good word embedding. IEEE Intell Syst. 2016;31(6):5–14.
17. Le Q, Mikolov T. Distributed representations of sentences and documents. In: Proceedings of the 31st International Conference on Machine Learning; 2014.
18. Li Y, Pan Q, Yang T, Wang S, Tang J, Cambria E. Learning word representations for sentiment analysis. Cogn Comput. 2017;9(6):843–51.
19. Liu P, Qiu X, Huang X. Recurrent neural network for text classification with multi-task learning. In: Proceedings of International Joint Conference on Artificial Intelligence; 2016. p. 2873–2879.
20. Manning CD, Surdeanu M, Bauer J, Finkel J, Bethard SJ, Mcclosky D. The Stanford CoreNLP natural language processing toolkit. In: Meeting of the association for computational linguistics: system demonstrations; 2014. p. 55–60.
21. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. In: Proceedings of International Conference on Learning Representations; 2013.
22. Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks. Computer Science. 2012;52(3):III–1310.
23. Pennington J, Socher R, Manning C. Glove: Global vectors for word representation. In: Proceedings of Conference on Empirical Methods in Natural Language Processing; 2014. p. 1532–1543.
24. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15(1):1929–58.
25. Tang D. Sentiment-specific representation learning for document-level sentiment analysis. In: Proceedings of 8th ACM International Conference on Web Search and Data Mining; 2015. p. 447–452.
26. Wang M, Liu M, Feng S, Wang D, Zhang Y. A novel calibrated label ranking based method for multiple emotions detection in chinese microblogs. Berlin: Natural Language Processing And Chinese Computing; 2014.
27. Yang Z, Yang D, Dyer C, He X, Smola A, Hovy E. Hierarchical attention networks for document classification. In: Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; 2017. p. 1480–1489.
28. Zhang X, Zhao J, Lecun Y. Character-level convolutional networks for text classification. In: Proceedings of Advances in Neural Information Processing Systems vol. 28; 2015. p. 649–657.
29. Zhao Z, Liu T, Hou X, Li B, Du X. Distributed text representation with weighting scheme guidance for sentiment analysis. In: Proceedings of Asia-Pacific Web Conference; 2016. p. 41–52.