International Journal of Machine Learning and Cybernetics (2020) 11:1939–1949
https://doi.org/10.1007/s13042-020-01084-9

ORIGINAL ARTICLE

# Character-level text classification via convolutional neural network and gated recurrent unit

**Bing Liu**[1,2,3] · **Yong Zhou**[1,2] · **Wei Sun**[4]

Received: 30 December 2018 / Accepted: 8 February 2020 / Published online: 4 March 2020


**Abstract**
Text categorization, or text classification, is one of key tasks for representing the semantic information of documents. Traditional deep leaning models for text categorization are generally time-consuming on large scale datasets due to slow convergence rate or heavily rely on the pre-trained word vectors. Motivated by fully convolutional networks in the field of image processing, we introduce fully convolutional layers to substantially reduce the number of parameters in the text classification model. A character-level model for short text classification, integrating convolutional neural network, bidirectional gated recurrent unit, highway network with the fully connected layers, is proposed to capture both the global and the local textual semantics at the fast convergence speed. Furthermore, In addition, error minimization extreme learning machine is incorporated into the proposed model to improve the classification accuracy further. Extensive experiments show that our approach achieves the state-of-the-art performance compared with the existing methods on the large scale text datasets.

**Keywords** Text categorization · Convolutional neural network · Gated recurrent unit · Highway network

## 1 Introduction

Text classification has been enormously applied to real-world problems, e.g., deceptive review identification [1, 2], sentiment analysis [3, 4], information retrieval [5], and email spam detection [6]. Many traditional techniques of text classification, such as topic modeling [7], are generally based on either the bag-of-words (BOW) or simple statistics of some ordered word combinations (such as *n*-grams) [8, 9]. However, the bag-of-words (BOW) ignores word order, such that different sentences might have the same representation. Although bag-of-*n*-grams considers the word order in short context, it is not applicable to text classification due to the sparse and high dimensional data representations. Traditional topic modeling methods, such as LDA (Latent Dirichlet Allocation), PLSA (Probabilitistic Latent Semantic Analysis) and NMF (Non-negative Matrix Factorization) [10, 11], are prone to serious issues with optimization, noise sensitivity, and instability for complex data relationships [12–14]. Different from topic modeling, some deep neural network models have been proposed to learn more effective vector representations of words, e.g., the pre-trained word vectors, which are mapped into a vector space such that semantically similar words have similar vector representations [15, 16].

By virtue of word embedding, a family of CNN text classification models was presented to explore the semantic representation of sentences. These methods were generally competitive to traditional models without any knowledge on the syntactic or semantic structures of a language [17, 18]. Kim [17] proposed a CNN structure for text classification, which utilizes pre-trained word embedding vectors as inputs. Then, a standard CNN model was applied to extract semantic features of sentences. In order to achieve better

✉ Yong Zhou
yzhou@cumt.edu.cn

✉ Wei Sun
sw3883204@163.com

1 School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, Jiangsu, People's Republic of China

2 Mine Digitization Engineering Research Center of Minstry of Education of the People's Republic of China, Xuzhou, People's Republic of China

3 Insititute of Electrics, Chinese Academy of Sciences, Beijing 100190, People's Republic of China

4 College of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, Jiangsu Province, People's Republic of China

performance, most of the researches constructed more complex models by increasing parameters or updating the architecture, such as using various word embedding techniques, increasing the number of layers, or introducing new pooling techniques [18–20]. However, these models generally converge very slowly. In addition, if embedding vectors of rare words are poorly estimated, it would likely have negative effects on the representations of words surrounding them and the performance of classification models. This is especially problematic in morphologically rich languages with long-tailed frequency distributions or domains with dynamic vocabularies (e.g. social media).

Fortunately, many researchers have demonstrated convolutional networks are useful in extracting information from raw signals [21–30], ranging from computer vision applications to speech recognition and others. There are convolutional networks approaches that use features extracted at word or word *n*-gram level form a distributed representation [23, 24], or utilize convolutional networks to extract character-level features toward different languages [21]. Consequently, these models have the ability of automatically learning abnormal character combinations, such as misspellings and emoticons.

Different from existing fast learning models based on convolutional neural networks [31, 32], we propose a character-level text classification model by utilizing both CNN and Bi-GRU to further improve the performance of the existing methods [33, 34]. Meanwhile, highway Network and fully convolutional layers are incorporated into the proposed model to speed up convergence rate. The main contributions of this work are summarized as follows:

(1) Other than existing models, the fully connected layers are replaced by fully convolutional layers in our model, which has significantly fewer parameters and is more applicable to large scale text classification tasks.
(2) By virtue of FCLs, an end-to-end character-level CNN-Highway-BiGRU network is constructed for handling raw text character sequences, and the argmax function is utilized to pre-train our end-to-end model, which can achieve satisfactory classification results with much faster convergence speed.
(3) By introducing error minimization extreme learning machine [35], our model can update output weights incrementally. Thus, compared with the exsiting method based on the softmax classifier, the proposed model has the ability of leveraging the extracted features to achieve better performance.

The remainder of the paper is organized as follows. We review related work in Sect. 2. Section 3 presents the details of our CNN-Highway-BiGRU network. Experimental results are shown in Sect. 4. Finally, Sect. 5 concludes the paper.

## 2 Related work

We mainly discuss some representative works for the two subtasks of text categorization, i.e., text feature extraction and classifier design.

Traditional methods of text feature representation have some limitations for classification. Specifically, some words occurring frequently across all documents tend to overshadow other words in the BOW model. TF-IDF, as a kind of term weight schemes, is commonly used to alleviate this problem by virtue of term frequency (TF) and inverse document frequency (IDF). In addition, the bag-of-*n*-grams model leverages the word order in short context and achieves better classification performance than BOW [7]. However, data sparsity, the curse of dimensionality and low utilization of semantic information are still challenging and intractable for these traditional methods [15, 16]. To this end, learning a low-dimensional vector representation of words via its local context, i.e., word embedding, has been developed and widely used in natural language processing (NLP) [36–40]. By transforming each short text unit (or sentence) into a matrix, CNN model can be naturally incorporated into text categorization. In all CNN-based methods, CNN-non-static, a single-layer and single-channel sentence model proposed by Kim is the simplest method and has satisfactory performance [17]. Compared with word embedding based methods, CNN based feature extraction methods are more efficient for raw signals. Santos confirmed that the accuracy of short text classification can be substantially improved if the English short text character sequence, as a processing unit, is input to learn the word and sentence-level features of the text, respectively [41]. Kanaris et al. [22] combined character-level *n*-grams with a linear classifier to obtain satisfactory performance of text classification. Zhang et al. [21] incorporated character-level features to convolutional networks for the classification tasks of different languages. Cho et al. [28] proposed a neural network language model to extract subword information based on a character-level convolutional neural network (CNN), whose output is used as an input to a recurrent neural network language model. Ling et al. [41] proposed a neural network using character level features to encode and decode individual characters in the translation process. Huang et al. [42] proposed a bidirectional LSTM model using character-level features to learn word embedding and character segmentation. Compared with traditional models, these approaches have superior performance in natural language processing. To speed up the convergence rate of deep CNN, Srivastava et al. [43] proposed the Highway Networks and combined this new structure with CNN and fully-connected networks.

On the other hand, the softmax classifier has been replaced by other classifiers to improve the performance

of CNN-based text classification models. Some hybrid models, such as CNN-SVM model, were proven to outperform the traditional CNN model in sentiment analysis and face recognition [8–10]. However, when cross-validation is used in experiments, it is generally time-consuming to select the appropriate parameters. Extreme Learning Machine (ELM), proposed by Huang et al. [35], has been proven to be superior to SVM and has fewer parameters need to manually adjust. Furthermore, EM-ELM has the ability of automatically choosing the optimal number of hidden nodes and has the advantage of updating output weights incrementally.

Motivated by these studies, we propose a novel character-level CNN-Highway-BiGRU network for text categorization, which can achieve better classification performance with much faster convergence speed. Different from existing models, the fully connected layers are replaced by fully convolutional layers to effectively reduce the number of parameters in our model. In addition, the argmax classifier is used to pre-train our end-to-end model, which efficiently extracts the local and global features from raw text character sequence. By virtue of extracted deep features, EM-ELM is introduced to further enhance the performance of text classification model by automatically choosing the optimal number of hidden nodes and updating output weights incrementally. Consequently, the proposed model not only has faster convergence rate compared with the state-of-the-art methods, but has better classification accuracy for text datasets.

## 3 Character-level text categorization based on CNN-highway-BiGRU

In this section, we develop a character-level deep learning model for text classification. The architecture of our model is shown in Fig. 1. In the proposed model, the fully connected layers (FCLs) have been removed and replaced by fully convolutional layers. Instead of the softmax classifier, the argmax classifier is used to pre-train our end-to-end model. Then, the pre-trained model works as a deep feature extractor and normalized deep features are fed to the EM-ELM classifier.

At first, our model receives a sequence of characters (a sentence) as input, and then finds the corresponding One-hot vector for each character through the dictionary which containing $m$ characters. Due to the different sentence lengths in the dataset, the length of the longest sentence in the entire dataset is generally obtained as l0 (i.e., the number of characters), and then each sentence is filled to l0 in the preprocessing. For characters or spaces that do not appear in the dictionary, we assign a 0 vector to them. For English datasets, the dictionary contains the following 70 characters: abcdefghijklmnopqrstuvwxyz-,;.!?:''/\|_@#$%^&*~`+= <>()[]{}0123456789. After a lookup of character embedding and stacking them to form the input matrix, convolution operations are performed between the input matrix and multiple filter kernels. Then, a max-over-time pooling operation is applied to obtain a fixed-dimensional representation of the word, which is output to the highway network. The outputs of highway network are used as the inputs to a bidirectional gated recurrent unit RNN model, which aims to learn semantics of words and take the information of the context into consideration. After the entire network is completely trained, the FCLs are removed and the hidden representations of the bidirectional GRU are fed to EM-ELM to perform classification tasks.

### 3.1 Model description

Our method utilizes the CNN-non-static architecture which is a single-layer and single-channel CNN-based sentence model.

In the CNN-non-static, each word in one sentence is replaced with its vector representation. Let $V$ be the vocabulary of characters, $d$ be the dimensionality of character embedding, and $\mathbf{A} \in \mathbb{R}^{d\times|V|}$ be the matrix character embeddings. Suppose that word $w$ is made up of a sequence of characters $[c_1, \ldots c_l]$, where $l$ is the length of word $w$. Then the character-level representation of $w$ is given by the matrix $\boldsymbol{E}^w \in \mathbb{R}^{d\times l}$, where the $j$-th column corresponds to the character embedding for $c_j$.

A narrow convolution is used between $\boldsymbol{E}^w$ and a kernel $\mathbf{K} \in \mathbb{R}^{d\times\omega}$ of width $\omega$, after a bias $b$ is added, a feature map $\boldsymbol{f}^w \in \mathbb{R}^{l-\omega+1}$ is introduced, whose $i$-th element is defined by

$$\boldsymbol{f}^w[i] = \tanh\left(\boldsymbol{E}^w[*, i : i+\omega-1], \mathbf{K}\right) + b), \tag{1}$$

where $\boldsymbol{E}^w[*, i : i+\omega-1]$ is the $i$-to-$(i+\omega-1)$-th column of $\boldsymbol{E}^w$ and $\mathbf{M}, \mathbf{N} = \mathrm{Tr}\left(\mathbf{M}\mathbf{N}^T\right)$ is the Frobenius inner product. Finally, we take the max-over-time

$$y^w = \max_i \boldsymbol{f}^w[i] \tag{2}$$

as the feature corresponding to the filter $\mathbf{K}$, which can extract the highest value for a given filter. Each filter is essentially picking out a character $n$-gram, where the size of the $n$-gram corresponds to the filter width.

### 3.2 Highway network

In order to solve the problem of model training in deep learning, Srivastava et al. [43] proposed a network that can optimize deep learning model, termed as Highway network. Under the gating mechanism, Highway network can locally regulate the information flow. In a feedforward neural network consisting of $L$ layers, each layer can use non-linear transformation $\mathbf{G}$ with the parameter $\boldsymbol{W}_{\mathrm{G}}$ to

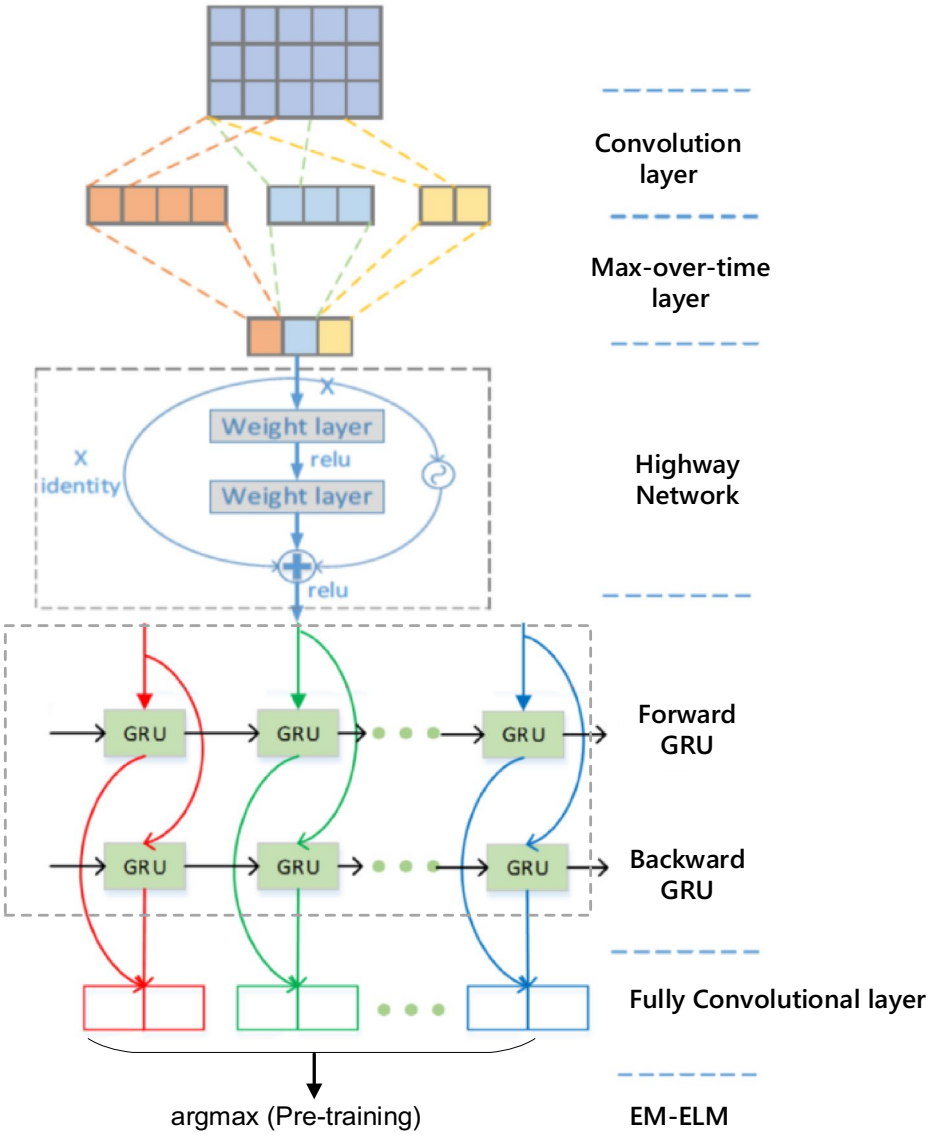


**Fig. 1** An illustration of the network architectures for pre-training and fine-tuning

generate the output $\mathbf{z}_i$ for the input $\boldsymbol{x}_i$, and the tensor $\mathbf{z}$ can be represented as

$$\mathbf{z} = \mathbf{G}(\boldsymbol{x}, \boldsymbol{W}_{\mathrm{G}}). \tag{3}$$

Highway networks introduce two non-linear transforms $\mathbf{T}$ and $\mathbf{C}$ into Eq. 4, so that the output $\mathbf{z}$ can be rewritten as

$$\mathbf{z} = \mathbf{G}(\boldsymbol{x}, \boldsymbol{W}_{\mathrm{G}}) \cdot \mathbf{T}(\boldsymbol{x}, \boldsymbol{W}_{\mathrm{T}}) + \boldsymbol{x} \cdot \mathbf{C}(\boldsymbol{x}, \boldsymbol{W}_{\mathrm{C}}), \tag{4}$$

where $\mathbf{T}$ is called the *transform* gate and $\mathbf{C}$ is called the *carry* gate, which express how much of the output is produced by transforming the input and carrying it, respectively. For simplicity, $\mathbf{C}$ is usually set as $\mathbf{1} - \mathbf{T}$. For every layer of highway network, we have

$$\mathbf{z} = \mathbf{G}(\boldsymbol{x}, \boldsymbol{W}_{\mathrm{G}}) \cdot \mathbf{T}(\boldsymbol{x}, \boldsymbol{W}_{\mathrm{T}}) + \boldsymbol{x} \cdot \left(\mathbf{1} - \mathbf{T}(\boldsymbol{x}, \boldsymbol{W}_{\mathrm{T}})\right), \tag{5}$$

where $\mathbf{G}$ is usually an affine transform followed by a non-linear activation function. The dimensionality of $\boldsymbol{x}, \mathbf{z}, \mathbf{G}(\boldsymbol{x}, \boldsymbol{W}_{\mathrm{G}})$ and $\mathbf{T}(\boldsymbol{x}, \boldsymbol{W}_{\mathrm{T}})$ must be the same to guarantee the validity of Eq. (5). Thus, based on the output of the transform gates, a highway layer can smoothly vary its behavior.

### 3.3 Gated recurrent unit

Recurrent neural network (RNN) can capture contextual information for text sequences. However, there are two major problems in traditional RNN model: vanishing gradient and exploding gradient. Gated recurrent unit (GRU), a variant

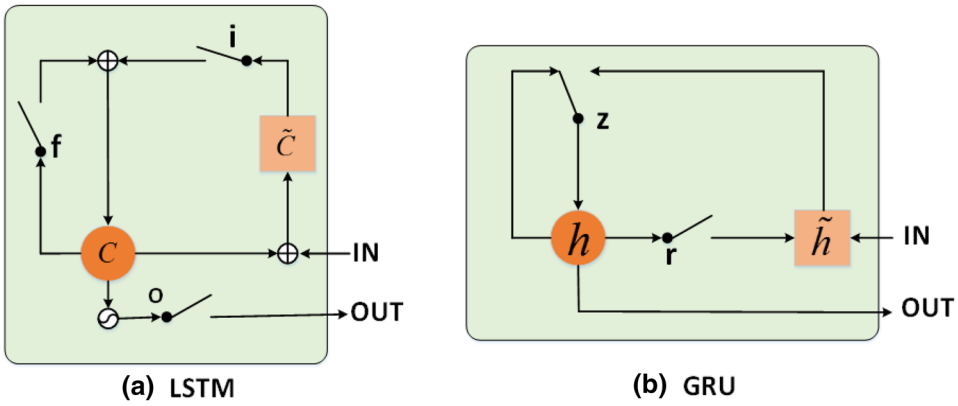


**Fig. 2** Architectures of LSTM and GRU. **a** LSTM, where $i$, $f$ and o are the input, forget and output gates, respectively. $C$ and $\tilde{C}$ denote the memory cell and the new memory cell content. **b** GRU, where $r$ and $z$ are the reset and update gates, and $h$ and $h$ are the activation and the candidate activation

of LSTM, is designed to avoid these problems [33, 34]. The architecture of LSTM unit and GRU unit are shown in Fig. 2 for comparison.

As shown in Fig. 2, GRU ensembles forget gate and input gate into a single update gate. It also mixes cell states and hidden states, and some other changes. The final model is simpler than the standard LSTM unit. In addition, the experiments indicate that GRU can achieve competitive or higher result than LSTM in the NLP task. And the performance of the GRU is better at the convergence time and the required epoch. Based on the above-mention reasons, we choose GRU to capture semantics of character-level and sentence-level feature in the event extraction task. In the proposed model, the two layer GRU network is designed to encode the sentence. A forward GRU computes the state $\overrightarrow{\boldsymbol{h}_t}$ of the past (left) context of the sentence at character $\boldsymbol{c}_t$, while a backward GRU network reads the same sentence in reverse and outputs $\overleftarrow{\boldsymbol{h}_t}$ given the future (right) context. Afterwards, we concatenate the outputs $\overrightarrow{\boldsymbol{h}_t}$ and $\overleftarrow{\boldsymbol{h}_t}$ as the output of GRU network, $\boldsymbol{h}_t = \left[\overrightarrow{\boldsymbol{h}_t} : \overleftarrow{\boldsymbol{h}_t}\right]$. For the input sentence, we set the number of hidden layers as $m$, the result of GRU network can be expressed as follows:

$$\boldsymbol{H} = \left[h_1; h_2; \ldots; h_n\right], \tag{6}$$

where $n$ is the length of the input sentence. The RNN network result is $\boldsymbol{H} \in \mathbb{R}^{n\times(2\times m)}$ where each row of $\boldsymbol{H}$ represents the feature of one word generated by GRU.

### 3.4 Fully convolutional layers

Our model replaces fully connected layers with convolutional layers. In Eq. (7), "*" denotes the convolution operator. The first parameter $\boldsymbol{x}$ represents the input, which is the output of former layers in the convolutional neural network, and the second parameter $\boldsymbol{w}$ represents the weight vector of one convolution kernel. The time complexity of single convolutional layer is shown in Eq. (8), where $M$ represents the size of the output feature map, $K$ is the convolution kernel size, $Cin$ represents the number of input channels, and $Cout$ denotes the number of output channels. The spatial complexity of the model is shown in Eq. (9). As can be seen from the formula, the spatial complexity of the model is only related to the convolution kernel size $K$ and the channel numbers $Cin$ and $Cout$, regardless of the input size. Thus, the neurons are locally connected to the input data and share parameters. In contrast, each node of the fully connected layer is connected to all nodes of the upper layer, which suffers from a large amount of parameters.

$$\boldsymbol{s} = \boldsymbol{x}(t) * \boldsymbol{w}(t) \tag{7}$$

$$time = \mathcal{O}\left(M^2 \times K^2 \times Cin \times Cout\right) \tag{8}$$

$$space = \mathcal{O}\left(K^2 \times Cin \times Cout\right) \tag{9}$$

### 3.5 Error minimized extreme learning machine for classification

In our model, to reduce a large number of parameters of fully connected layers, the classifier, based on the argmax function, is used to pre-train our model for two-class or multiclass classification. Thus, the length of the last layer is determined by the number of classes. Then, error minimized extreme learning machine (EM-ELM) [35], which can add random hidden nodes to SLFNs one by one or group by group (with varying group size), is utilized to achieve better classification results by incrementally updating the output weights. The error minimized ELM (EM-ELM) algorithm is described as follows.

Compared with the standard ELM, which has to recalculate the output weights if the network architecture is updated, EM-ELM effectively reduces the computation complexity by updating the output weights incrementally. Furthermore, its convergence can still be guaranteed [35].

Algorithm 1: Error minimized extreme learning machine

Input: A set of training data $\{(\boldsymbol{x}_i, \boldsymbol{t}_i)\}_{i=1}^{N}$, the maximum number of hidden nodes $L_{max}$, the expected learning accuracy $\epsilon$, the initial number of hidden nodes $L_0$ and the maximum of hidden nodes $L_{max}$.

Output: output weights $\boldsymbol{\beta}$

*Phase 1—Initialization Phase*

1 Initialize the SLFN with a small group of randomly generated hidden nodes $\{(\boldsymbol{a}_i, b_i)\}_{i=1}^{L_0}$

2 Calculate the hidden-layer output matrix $\mathbf{H}_1$

$$\mathbf{H}_1 = \begin{bmatrix} G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_1) & \cdots & G(\boldsymbol{a}_{L_0}, b_{L_0}, \boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_N) & \cdots & G(\boldsymbol{a}_{L_0}, b_{L_0}, \boldsymbol{x}_N) \end{bmatrix}_{N \times L_0}$$

3 Compute the corresponding output error

$$\mathrm{E}(\mathbf{H}_1) = \left\| \mathbf{H}_1 \mathbf{H}_1^{\dagger} \mathbf{T} - \mathbf{T} \right\|.$$

*Phase 2—Recursively Growing Phase*

1 $k \leftarrow 0$

2 while $L_k < L_{max}$ and $\mathrm{E}(\mathbf{H}_k) > \epsilon$ do

Randomly add $\delta L_{k-1}$ hidden nodes to the existing SLFN

$L_k \leftarrow L_{k-1} + \delta L_{k-1}$

$$\delta \mathbf{H}_k \leftarrow \begin{bmatrix} G(\boldsymbol{a}_{L_{k-1}+1}, b_{L_{k-1}+1}, \boldsymbol{x}_1) & \cdots & G(a_{L_k}, b_{L_k}, \boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ G(\boldsymbol{a}_{L_{k-1}+1}, b_{L_{k-1}+1}, \boldsymbol{x}_N) & \cdots & G(a_{L_k}, b_{L_k}, \boldsymbol{x}_N) \end{bmatrix}_{N \times \delta L_{k-1}}$$

$\mathbf{H}_{k+1} \leftarrow [\mathbf{H}_k, \delta \mathbf{H}_k]$

$\mathbf{D}_k \leftarrow \left( (\mathbf{I} - \mathbf{H}_k \mathbf{H}_k^{\dagger}) \delta \mathbf{H}_k \right)^{\dagger}$

$\mathbf{U}_k \leftarrow \mathbf{H}_k^{\dagger} (\mathbf{I} - \delta \mathbf{H}_k \mathbf{D}_k)$

$\boldsymbol{\beta}_{k+1} \leftarrow \begin{bmatrix} \mathbf{U}_k \\ \mathbf{D}_k \end{bmatrix}^T \mathbf{T}$

$k \leftarrow k + 1$

3 return $\boldsymbol{\beta}$

## 4 Experiments

In this section, we evaluate the performance of our model on large scale datasets, including English and Chinese text datasets. The experiments were carried out using Ubuntu 14.04, Python 2.7 and TensorFlow 1.13.1 with Intel i7 4.0-GHZ CPU and 64G DDR4 Memory.

### 4.1 Datasets

In the experiments, for fair comparison we used both English and Chinese large-scale text datasets to test different models. English datasets include MR,[1] SST-2,[2] Tweet,[3] AG-News,[4] Yah,[5] DBPedia[6] and Yelp Review Full (Yelp.F).[7] Chinese datasets include the Sogou News dataset[8] and Chinese Movie Reviews dataset.[9] The detailed descriptions of these datasets are listed in Table 1.

[1] https://github.com/kuberkaul/SentimentAnalysis-MovieReviews.

[2] https://nlp.stanford.edu/sentiment/.

[3] https://trec.nist.gov/data/tweets/.

[4] http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html.

[5] https://webscope.sandbox.yahoo.com/.

[6] https://wiki.dbpedia.org/Datasets.

[7] https://s3.amazonaws.com/fast-ai-nlp/yelp_review_full_csv.tgz.

[8] https://www.sogou.com/labs/resource/c1s.php.

[9] https://github.com/Jacob-Zhou/LRMR_Core.

**Table 1** Statistics of English and Chinese datasets

| Datasets | Classes | Number of selected samples | Language | Size of the vocabulary |
|---|---|---|---|---|
| MR | 2 | 10,662 | English | 18,765 |
| SST-2 | 2 | 11,434 | English | 16,185 |
| Tweet | 10 | 25,552 | English | 33,438 |
| AG-News | 4 | 120,000 | English | 127,600 |
| Yah | 10 | 140,000 | English | 146,000 |
| DBPedia | 14 | 560,000 | English | 630,000 |
| Yelp.F | 5 | 650,000 | English | 700,000 |
| Sogou News | 5 | 450,000 | Chinese | 510,000 |
| Chinese Movie Reviews | 2 | 36,124 | Chinese | 90,958 |

### 4.2 Experimental settings

The kernel sizes were set as 1, 2, 3, 4, 5, 6, 8 respectively, and the numbers of channels were 50, 100, 150, 150, 200, 200, 200 correspondingly. For fair comparison, ReLU activation was used in all CNN-based models, and the dropout rate was set to 0.5 and mini-batch size 32. In GRU network, the number of hidden layer $m$ was set to 512 as in Ref [34]. The length of each fully convolutional layer was set to 512 and $1 \times 1$ kernels were used. We utilized the Adam optimizer instead of stochastic gradient decent (SGD) to pre-train our model, learning rate was set to 0.001 and the dropout rate was 0.5. For EM-ELM, we used the sigmoid activation function. Table 2 reports the maximum number of hidden nodes $L_{\max}$ and the expected learning error for each dataset. The SLFN was initialized by one hidden node and then new random hidden nodes are added one by one.

To verify the effectiveness and efficiency of the proposed model, we compared our model with the traditional classification methods and convolutional neural network classification models. The former include Naive Bayes, Multinomial Naive Bayes (MNB), KNN and Linear-SVM, and the latter include convolutional neural networks for text classification based on the CNN-rand, CNN-static, CNN-non-static and CNN-multichannel methods. We tested all algorithms by the 10-fold cross validation procedure, and all the algorithms were subjected to the same folds of the cross-validation process.

For these traditional classification methods, we first segmented words from sentences, and then removed special characters, such as the space character, and stopwords on Chinese datasets. For English datasets, we directly removed special characters and stopwords. Specifically, for each dataset, the bag-of-words model was constructed by selecting 30,000 most frequent words from the training subset. Then, the counts of each word were set as the term-frequency. The

**Table 2** The settings of the hyperparameters in EM-ELM on different datasets

| Datasets | $L_{max}$ | $e$ (%) |
|---|---|---|
| MR | 10,000 | 5 |
| SST-1 | 10,000 | 5 |
| Tweet | 20,000 | 5 |
| AG-News | 100,000 | 5 |
| Yah | 100,000 | 5 |
| DBPedia | 150,000 | 5 |
| Yelp.F | 150,000 | 5 |
| Sogou News | 100,000 | 5 |
| Chinese Movie Reviews | 20,000 | 5 |

inverse document frequency was set as the logarithm of the division between total number of samples and number of samples with the word in the training subset. To further reduce the dimensionality of the features, the Linear Discriminant Analysis (LDA) algorithm was performed to obtain low-dimensional vectors. The dimension of the embedding was set to 500 and the final features were normalized by dividing the largest feature value. Finally, NB, MNB, KNN and SVM were carried out on the generated low-dimensional features. For KNN, we set $k$ as 10 and used cosine similarity to obtain the $k$ nearest neighbors. For the sake of large amount of training data, we only performed linear SVM using sequential minimal optimization algorithm, where the penalty parameter $C$ equals to default value 1. For Multinomial Naive Bayes, we used the same parameters as those in [44].

In the CNN-rand model, all word vectors were initialized randomly and optimized in training. For the CNN-static model, the word embeddings were learnt from the training subset of each dataset with skip-gram [21], and the dimension of word embedding was set to 128 as in Ref. [44]. In CNN-non-static, these word vectors could be tuned. The CNN-multichannel model can be regarded as a combination of CNN-static and CNN-non-static. For Chinese datasets, we employed pypinyin package combined with jieba Chinese segmentation system to produce Pinyin—a phonetic romanization of Chinese, as in Ref. [34]. The proposed model can then be applied to Chinese datasets without change.

## 4.3 Experimental results

### 4.3.1 Experiments on english datasets

We first compared our method with traditional methods and CNN based models on English datasets. In this experiment, the number of layers of highway networks was set to 3. The experiment results are listed in Table 3. As can be seen, the CNN-based models have the better classification accuracy than traditional methods. It is due to the fact that deep models have the advantage of extracting global and local features by virtue of multilayer neural network. Specifically, our method significantly outperforms both traditional methods and the existing CNN based models. It achieves all best results from 7 datasets. The performance of the proposed model is obviously superior to that of the CNN-non-static model, which shows that raw character information is useful to improve the performance of text classification. Our method is much better than the CNN-LSTM hybrid model, which validates the effectiveness of integrating CNN, highway network, GRU and fully convolutional layers into the united model. In addition, different from existing CNN-based methods, we leveraged the extracted features by means of EMELM. Consequently, the proposed model inherits the advantages of both traditional CNN-based deep neural networks and EMELM, which contributes to the performance improvement of text classification algorithms.

**Table 3** Performance comparison between different text categorization methods on English datasets (%)

| Models | MR | SST-2 | Tweet | AG-News | Yah | DBPedia | Yelp.F |
|---|---|---|---|---|---|---|---|
| Naive Bayes | 62.35 | 72.17 | 76.63 | 84.82 | 60.96 | 89.37 | 56.79 |
| MNB | 76.13 | 81.47 | 82.16 | 89.58 | 69.63 | 92.01 | 62.25 |
| KNN | 61.37 | 68.36 | 72.52 | 79.51 | 55.47 | 85.62 | 50.58 |
| Linear-SVM | 69.88 | 78.64 | 80.53 | 85.14 | 62.27 | 86.75 | 55.20 |
| CNN-rand | 74.83 | 83.36 | 85.16 | 87.52 | 71.65 | 90.92 | 61.76 |
| CNN-static | 73.75 | 81.16 | 82.86 | 89.86 | 71.78 | 93.41 | 61.53 |
| CNN-non-static | 74.52 | 84.04 | 85.39 | 90.82 | 72.95 | 92.17 | 60.21 |
| CNN-multi-channel | 75.23 | 83.15 | 84.13 | 90.26 | 71.47 | 92.98 | 61.26 |
| CNN-char-static | 75.57 | 83.21 | 84.78 | 86.74 | 70.28 | 92.35 | 60.83 |
| CNN-char-non-static | 76.05 | 82.78 | 84.54 | 88.93 | 71.43 | 92.36 | 60.74 |
| LSTM | 76.44 | 83.25 | 85.97 | 91.05 | 71.49 | 93.24 | 61.04 |
| CNN-LSTM | 78.46 | 85.47 | 87.03 | 92.32 | 72.36 | 94.13 | 62.18 |
| Our method | 83.92 | 91.63 | 92.25 | 94.40 | 75.89 | 98.97 | 67.42 |

**Table 4** Performance comparison between CNN-based methods on English datasets (%)

| Models | MR | SST-2 | Tweet | AG-News | Yah | DBPedia | Yelp F. |
|---|---|---|---|---|---|---|---|
| CNN-softmax | 74.35 | 83.09 | 84.29 | 90.44 | 70.78 | 92.01 | 60.84 |
| CNN-EMELM | 76.77 | 83.83 | 85.51 | 91.18 | 70.32 | 93.32 | 61.07 |
| CNN-LSTM-softmax | 77.03 | 84.53 | 87.14 | 92.32 | 70.95 | 94.50 | 62.59 |
| CNN- Highway-GRU-softmax | 78.34 | 87.96 | 90. 65 | 92.87 | 71.19 | 95.40 | 64.78 |
| Our method | 83.92 | 91.63 | 92.25 | 94.40 | 75.89 | 98.97 | 67.42 |

To further validate the effectiveness of our model, we tested different CNN-based text classification models using softmax and EM-ELM, respectively, and then reported the performance of classifiers in Table 4. For CNN-EMELM, we replaced the softmax classifier by the EM-ELM classifier based on the same network structure. Comparing CNN-softmax and CNN-EMELM, we can see that EMELM has the ability of improving the classification accuracy by using the same extracted features as CNN-softmax. In addition, it can be seen in Table 4 that our model is obviously superior to the counterparts based on softmax classifiers. The experimental results show that EM-ELM can enhance the performance further. As a result, these experimental results validate the effectiveness of the proposed model.

#### 4.3.2 Experiments on Chinese datasets

We further implemented different algorithms on Chinese datasets to validate the effectiveness and efficiency of the proposed model. The experiment results are listed in Table 5.

From Table 5, we can come to the same conclusion that the CNN-based models perform better than the traditional classification models on Chinese datasets. Specifically, it can be seen from Table 5 that the performance of CNN-rand model is similar to that of CNN-char-static model and CNN-char-non-static model, and is superior to that of Naive Bayes, MBN, KNN and Linear-SVM. The CNN-based models with highway networks outperform those without highway networks. The proposed model performs best among all models, which further validates the effectiveness of our model on Chinese datasets.

The accuracy and convergence curves on the Chinese movie reviews dataset were displayed in Figs. 3 and 4, respectively. From Figs. 3 and 4, we can see that our model has better performance than the standard CNN and highway network based CNN. It has superior classification accuracy with faster convergence speed in the training process.

Finally, we compared our methods with several widely used supervised text classification models, including the character level convolutional model (char-CNN) [21], Region embedding for text classification (Region.emb) [44], the character based convolution recurrent network (char-CRNN) [45], the bigram FastText (bigram-FastText) [46], the Discriminative LSTM (D-LSTM) [47] as well as the very deep convolutional network (VDCNN) [48]. The experimental results were reported in Table 6. As can be seen, our method achieves the best 4 results among all algorithms. For the Yah dataset, the classification accuracy of our method on the test dataset is very close to that of Region.emb. On AG, DBPedia, Yah and Yelp F, the performance of the proposed method is much better than that of other methods. Notably, all algorithms have unsatisfactory classification performance on Yah and Yelp F.

**Table 5** Performance comparison between different text categorization methods on Chinese datasets (%)

| Models | Sogou News | Chinese movie reviews |
|---|---|---|
| Naive Bayes | 82.35 | 78.29 |
| MBN | 89.87 | 83.14 |
| KNN | 66.74 | 73.58 |
| Linear-SVM | 84.26 | 79.39 |
| CNN-rand | 90.95 | 85.27 |
| CNN-static | 91.30 | 88.32 |
| CNN-non-static | 92.25 | 87.16 |
| CNN-multi-channel | 93.19 | 86.63 |
| CNN-char-static | 93.76 | 83.59 |
| CNN-char-non-static | 90.51 | 84.47 |
| CNN + highway + LSTM | 93.94 | 86.62 |
| Our method | 97.25 | 91.15 |

To analyze the stability of our method, we also reported results of several repeated runs on Yah and Yelp F in Tables 7 and 8, respectively. As can be seen, five independent runs are conducted on each dataset of Yah and Yelp F, where both standard deviations are within 0.051, and maximum performance variances are within 0.13% on accuracy, indicating that our method is still stable even if the accuracy is relatively low. Overall, our method is superior to the state-of-the-art algorithm on large scale datasets.

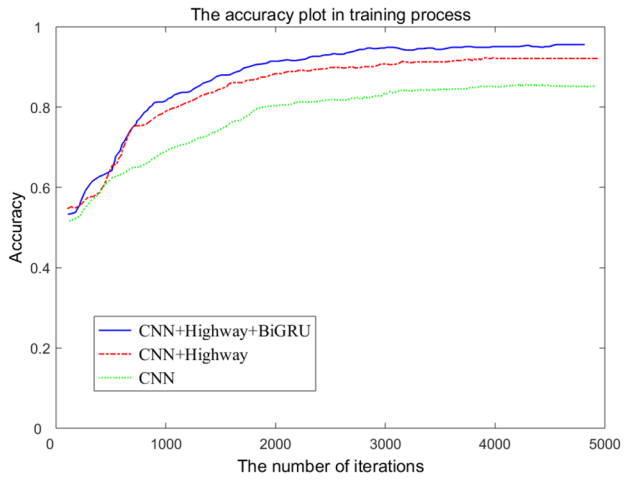


**Fig. 3** The accuracy curves over iterations on the Chinese Movie Reviews dataset

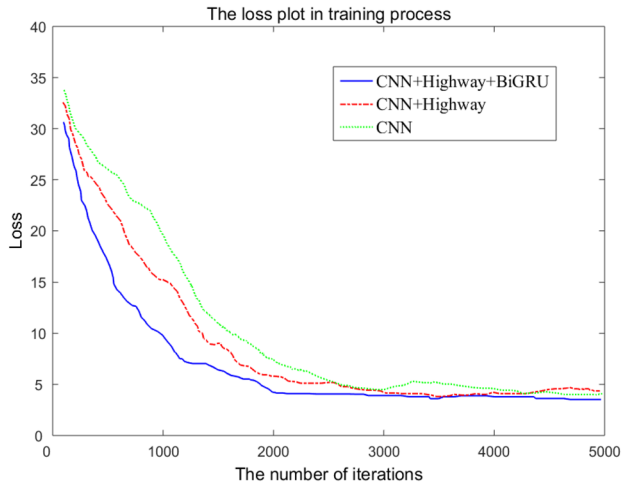


**Fig. 4** The convergence curves over iterations on the Chinese Movie Reviews dataset

**Table 6** Performance comparison with the state-of-the-art methods on several datasets (%)

| Models | AG | DBPedia | Sogou | Yah | Yelp F. |
|---|---|---|---|---|---|
| Char-CNN [21] | 88.56 | 98.30 | 95.10 | 71.20 | 62.00 |
| Char-CRNN [45] | 92.32 | 98.30 | 95.10 | 71.20 | 62.00 |
| bigram-FastText [46] | 93.26 | 98.60 | 96.80 | 72.30 | 63.90 |
| Region.emb [44] | 93.71 | 98.90 | **97.60** | 73.70 | 64.90 |
| D-LSTM [47] | 92.94 | 98.70 | 94.90 | 73.70 | 59.60 |
| VDCNN [48] | 92.15 | 98.70 | 96.80 | 73.40 | 64.70 |
| Our method | **94.40** | **98.97** | 97.25 | **75.89** | **67.42** |

Bold value indicates that the best results of classification among all algorithms

**Table 7** Performance variance through several repeated runs on Yah

| Tries Num. | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Accuracy (%) | 75.88 | 75.84 | 75.93 | 75.87 | 75.95 |
| Mean ± SD | 75.89 ± 0.045 | | | | |
| Maximum performance variance | ≤ %0.11 | | | | |

**Table 8** Performance variance through several repeated runs on Yelp F

| Tries Num. | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Accuracy (%) | 67.40 | 67.44 | 67.36 | 67.46 | 6.49 |
| Mean ± SD | 67.42 ± 0.051 | | | | |
| Maximum performance variance | ≤ %0.13 | | | | |

## 5 Conclusion

In this paper, we proposed a character-level text categorization model based on convolutional neural network, Highway network and gated recurrent unit, which has the ability of efficiently extracting both the global and the local textual semantics. In addition, the fully convolutional layers are introduced to substantially reduce the large amount of parameters arisen from the original fully convolutional layers. Thus, the convergence rate of the model can be significantly sped up. Furthermore, combined with error minimization extreme learning machine, the extracted features are leveraged to improve the classification accuracy. Experimental results validate that the proposed method can achieve satisfactory classification performance with faster learning speed.

**Acknowledgements** This work is supported by "the Fundamental Research Funds for the Central Universities" (No. 2017XKQY082).

## References

1. Zhang W, Tang X, Yoshida T (2015) TESC: an approach to text classification using semi-supervised clustering. Knowl-Based Syst 75:152–160
2. Zhang W, Du Y, Yoshida T, Wang Q (2018) DRI-RCNN: an approach to deceptive review identification using recurrent convolutional neural network. Inf Process Manag 54(4):576–592
3. Poria S, Cambria E, Bajpai R, Hussain A (2017) A review of affective computing: from unimodal analysis to multimodal fusion. Inf Fusion 37:98–125
4. Cambria E (2016) Affective computing and sentiment analysis. IEEE Intell Syst 31(2):102–107

5. Gong Y, Ke Q, Isard M, Lazebnik S (2014) A multi-view embedding space for modeling internet images, tags, and their semantics. Int J Comput Vision 106(2):210–233
6. Carreras X, Marquez L (2001) Boosting trees for anti-spam email filtering. In: RANLP, pp 58–64
7. Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: International conference on machine learning, pp 1188–1196
8. Poria S, Cambria E, Howard N, Huang G-B, Hussain A (2016) Fusing audio, visual and textual clues for sentiment analysis from multimodal content. Neurocomputing 174:50–59
9. Cambria E, White B (2014) Jumping NLP curves: a review of natural language processing research. IEEE Comput Intell Mag 9(2):48–57
10. Pavlinek M, Podgorelec V (2017) Text classification method based on self-training and LDA topic models. Expert Syst Appl 80:83–93
11. Fu R, Qin B, Liu T (2015) Open-categorical text classification based on multi-LDA models. Soft Comput 19(1):29–38
12. Agrawal A, Fu W, Menzies T (2018) What is wrong with topic modeling? And how to fix it using search-based software engineering. Inform Software Tech 98:74–88
13. Panichella A, Dit B, Oliveto R, Di Penta M, Poshyvanyk D, De Lucia A (2013) How to effectively use topic models for software engineering tasks? An approach based on genetic algorithms. In: Proceedings of the 2013 international conference on software engineering, IEEE Press, pp 522–531
14. Tai KS, Socher R, Manning CD (2015) Improved semantic representations from tree-structured long short-term memory networks. In: Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing, pp 1556–1566
15. Cambria E, Fu J, Bisio F, Poria S (2015) AffectiveSpace 2: enabling affective intuition for concept-level sentiment analysis. In: AAAI, Austin, pp 508–514
16. Chunting Z, Chonglin S, Zhiyuan L et al (2015) A C-LSTM neural network for text classification. Comput Sci 1(4):39–44
17. Kim Y (2014) Convolutional neural networks for sentence classification. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1746–1751
18. dos Santos C, Gatti M (2014) Deep convolutional neural networks for sentiment analysis of short texts. In: Proceedings of COLING 2014, the 25th international conference on computational linguistics: technical papers, pp 69–78, Dublin, Ireland
19. Bengio Y, Schwenk H, Senécal J-S, Morin F, Gauvain J-L (2016) Neural probabilistic language models. In: Innovations in machine learning. Springer, pp 137–186
20. Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P (2011) Natural language processing (almost) from scratch. J Mach Learn Res 12:2493–2537
21. Zhang X, Zhao JB, Lecun Y (2015) Character-level convolutional networks for text classification. Adv Neural Inf Process Syst 28:1–9
22. Kanaris I, Kanaris K, Houvardas I, Stamatatos E (2007) Words versus character n-grams for anti-spam filtering. Int J Artif Intell Tools 16(06):1047–1067
23. Santos CD, Zadrozny B (2014) Learning character-level representations for part-of-speech tagging. In: Proceedings of the 31st international conference on machine learning (ICML-14), pp 1818–1826
24. Shen Y, He X, Gao J, Deng L, Mesnil G (2014) A latent semantic model with convolutional-pooling structure for information retrieval. In: Proceedings of the 23rd ACM international conference on conference on information and knowledge management, pp 101–110
25. Poria S, Cambria E, Gelbukh A (2015) Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In: EMNLP, pp 2539–2544
26. Xu B, Ye D, Xing Z, Xia X, Chen G, Li S (2016) Predicting semantically linkable knowledge in developer online forums via convolutional neural network. In: Proceedings of the 31st IEEE/ACM international conference on automated software engineering (ASE 2016). New York, NY, USA, pp 51–62
27. Chaturvedi I, Ong Y-S, Tsang I, Welsch R, Cambria E (2016) Learning word dependencies in text by means of a deep recurrent belief network. Knowl-Based Syst 108:144–154
28. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using rnn encoder-decoder for statistical machine translation. In: Proceedings of empirical methods in natural language processing, pp 1724–1734
29. Poria S, Cambria E, Gelbukh A (2016) Aspect extraction for opinion mining with a deep convolutional neural network. Knowl-Based Syst 108:42–49
30. Majumder N, Poria S, Gelbukh A, Cambria E (2017) Deep learning based document modeling for personality detection from text. IEEE Intell Syst 32(2):74–79
31. Jaderberg M, Vedaldi A, Zisserman A (2014) Speeding up convolutional neural networks with low rank expansions. In: Proceedings of the British machine vision conference
32. Lebedev V, Ganin Y, Rakhuba M, Oseledets I, Lempitsky V (2015) Speeding-up convolutional neural networks using finetuned cp-decomposition. In: 3rd international conference on learning representations
33. Tang D, Qin B, Liu T (2015) Document modeling with gated recurrent neural network for sentiment classification. In: Proceedings of the 2015 conference on empirical methods in natural language processing, pp 1422–1432
34. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F Schwenk H, Bengio Y (2015) Learning phrase representations using rnn encoder-decoder for statistical machine translation. In: Proceedings of the 2014 conference on empirical methods in natural language processing, pp 1724–1734
35. Feng G, Huang GB, Lin Q et al (2009) Error minimized extreme learning machine with growth of hidden nodes and incremental learning. IEEE Trans Neural Netw 20(8):1352–1357
36. Zhang M-L, Zhou Z-H (2006) Multilabel neural networks with applications to functional genomics and text categorization. IEEE Trans Knowl Data Eng 18(10):1338–1351
37. Nam J, Kim J, Mencía EL, Gurevych I, Fürnkranz J (2014) Largescale multi-label text classification—revisiting neural networks. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, pp 437–452
38. Benites F, Sapozhnikova E (2015) Haram: a hierarchical aram neural network for large-scale text classification. In: 2015 IEEE international conference on data mining workshop (ICDMW). IEEE, pp 847–854
39. Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp 807–814
40. Dai AM, Le QV (2015) Semi-supervised sequence learning. In: Advances in neural information processing systems, pp 3079–3087
41. Ling W, Luís T, Marujo L et al (2015) Finding function in form: compositional character models for open vocabulary word representation. Comput Sci 10:1899–1907
42. Huang Z, Xu W, Yu K (2015) Bidirectional LSTM-CRF models for sequence Tagging. arXiv:1508.01991
43. Srivastava RK, Greff K, Schmidhuber J (2015) Training very deep networks. Comput Sci 10:1–5

44. Qiao C, Huang B, Niu G, Li D, Dong D, He W, Yu D, Wu H (2018) A new method of region embedding for text classification. In: International conference on learning representations
45. Xiao Y, Cho K (2016) Efficient character-level document classification by combining convolution and recurrent layers. arXiv :1602.00367
46. Joulin A, Grave E, Bojanowski P, Mikolov T (2017) Bag of tricks for efficient text classification. In: Proceedings of the 15th conference of the european chapter of the association for computational linguistics, vol 2, pp 427–431
47. Yogatama D, Dyer C, Ling W, Blunsom P (2017) Generative and discriminative text classification with recurrent neural networks. arXiv:1703.01898
48. Conneau A, Schwenk H, Barrault L, Lecun Y (2016) Very deep convolutional networks for natural language processing. arXiv :1606.01781