ISSN 2007-9737

Hybrid Attention Networks for Chinese Short Text Classification

Yujun Zhou[1,2,3], Jiaming Xu[1], Jie Cao[1,2,3], Bo Xu[1], Changliang Li[1], Bo Xu[1]

[1] Institute of Automation, Chinese Academy of Sciences, Beijing, P.R. China

[2] University of Chinese Academy of Sciences, Beijing, P.R. China

[3] Jiangsu Jinling Science and Technology Group Co., Ltd, Nanjing, P.R. China

{zhouyujun2014, jiaming.xu, caojie2014, boxu, changliang.li, xubo}@ia.ac.cn

Abstract. To improve the classification performance for Chinese short text with automatic semantic feature selection, in this paper we propose the Hybrid Attention Networks (HANs) which combines the word- and character-level selective attentions. The model firstly applies RNN and CNN to extract the semantic features of texts. Then it captures class-related attentive representation from word- and character-level features. Finally, all of the features are concatenated and fed into the output layer for classification. Experimental results on 32-class and 5-class datasets show that, our model outperforms multiple baselines by combining not only the word- and character-level features of the texts, but also class-related semantic features by attentive mechanism.

Keywords. Chinese short texts, text classification, attentive mechanism, convolutional neural network, recurrent neural network.

1 Introduction

Short text classification plays an import role in natural language processing. It has a wide range of applications, including SMS spam filtering, sentiment analysis and question answering etc. The task can be defined as follows: given a set of input texts T and a set of classes (or labels) L, the goal is to find some appropriate methods to assign a value from the set of L to each text in T. Many approaches have been proposed to classify short text, such as Naive Bayes (NB), Support Vector Machine (SVM), SVM with NB features (NBSVM), Latent Dirichlet Allocation (LDA). In recent years, Artificial Neural Networks (ANNs) also have shown promising results, including Convolutional Neural Networks (CNNs) [7, 8], Recursive Neural Networks (RecNNs) [15], Recurrent Neural Networks (RNNs) [12] and other variants [9].

However, above methods have been developed primarily for western language (e.g. English) datasets. In English text, there is a blank space between two words. And yet Chinese words are written next to each other without a delimiter between them, we should conduct word segmentation on the Chinese short text firstly. Furthermore, errors caused by Chinese Word Segmentation (CWS) will be propagated to the subsequent tasks and reduce their performance.

From the perspective of semantic analysis, words and characters are significant to short text classification. That is to say, several words or characters may determine the class of a short text. For example, in news titles the words "体育(sport)", "足球(football)", "篮球(basketball)" occur relatively frequently in sports, and the words "电脑(computer)", "编程语言(program language)" and "硬盘(hard disk)" are relatively unique for information technology. And similarly, in online

product comments the characters "好(good)", "坏(bad)" and "赞(awesome)" may show the sentiment directly. Inspired by the recent success of attention mechanism on many NLP tasks, such as neural machine translation [1] and document classification [17], we explore an end-to-end approach with attention-based neural networks to locate the key words and characters for Chinese short text classification.

The main contribution of this paper is a novel neural architecture, Hybrid Attention Networks (HANs), which is aimed to provide two insights into which characters or words contribute to the classification decision on Chinese short texts. Firstly, since Chinese short texts are composed of words or characters, we build representations based on RNN and CNN for each sentence with words and characters embeddings respectively, then concatenate them into a sentence representation. Secondly, it can be observed that there is typically a salient set of characters and words that signal each sentence class. Furthermore, the same word or character may be differentially important in different sentences.

Considering this sensitivity, our model introduces two attentive mechanisms with context, that is, character- and word-level attentions, which are designed to pay more or less attention to distinctive characters or words when constructing the character- and word-level representations of the sentence, and then concatenate these two-level attention models into an attentive representation of the sentence. Experimental results show that our model outperforms multiple baselines on one 32-class and two 5-class data sets.

The remainder of this paper is organized as follows: Section 2 briefly introduces the works related to this study. Section 3 describes the HANs model in detail. Experimental results and discussion are reported in Section 4. Section 5 concludes this paper.

2 Related Work

In general, a short text only contains several to dozens of words (e.g. the length of a mobile message is limited to 140 characters), it cannot provide enough word co-occurrences that are very important to those learning methods for normal documents. Due to data sparseness, traditional classification methods that depend on the word frequency or enough word co-occurrences, such as Naive Bayes (NB), Maximum Entropy (ME), K Nearest Neighbors (KNN) and Support Vector Machines (SVMs), usually fail to obtain satisfied performance. Hence some improved methods for short text classification appeared, such as sematic analysis, semi-supervised and ensemble models for short text classification [16].

Zelikovitz et al. [18] applied semantic analysis with Latent Semantic Indexing (LSI) to classify short text.

Chen et al. [3] put forward a solution to integrate the topics of multi-granularity, which can model the semantic of short texts more precisely.

Cabrera et al. [2] proposed a semi-supervised short text classification method, Distributional Term Representations (DTRs), which enriched document representations based on contextual information that overcame the small-length and high-sparseness short text to some extent.

Feng et al. [4] introduced an ensemble learning method, which directly measured the correlation between a short text instance and a domain instead of representing short texts as vectors of weights and outperformed the baselines based on Vector Space Model (VSM).

In recent years, many methods based on neural networks have been applied to classify text, in which distributed word representations (i.e. word embedding) [14] have been typically used as inputs to neural network models.

Socher et al. [15] introduced a Recursive Neural Network model using Matrix Vector (MV-RNN), which learned compositional vector representations for phrases and sentences with variable length.

Le and Mikolov [10] proposed paragraph vector to learn fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs and documents.

Kalchbrenner et al. [7] put forward Dynamic Convolutional Neural Network (DCNN) to model sentences of varying length, which used dynamic k-max pooling to explicitly capture short and long-range relations without relying on a parse tree.

Then Kim [8] proposed a simple CNN with two channels of word vectors which allow the using of dynamic-updated and static word embeddings simultaneously.

Liu et al. [12] introduced a Multi-Timescale Long Short-Term Memory (MT-LSTM) neural network to model short or long texts.

For Chinese text classification, Lai et al. [9] proposed a Recurrent Convolutional Neural Network (RCNN) and applied it to the task of text classification on Chinese long documents.

Zhang et al. [19] proposed the use of character-level Convolutional Networks (ConvNets) for text classification on Chinese news corpus.

Li et al. [11] developed two component-enhanced Chinese character embedding models and their bi-gram extensions for text classification on Chinese news titles.

More recently, there have been research efforts to incorporate attention mechanisms into CNNs or RNNs that are typically used in NLP applications. Bahdanau et al. [1] applied attention based model to machine translation, which allowed the decoder to watch different parts of the source sentence at each step of the output generation rather than to encode the full source sentence into a fixed-length vector, and explicitly find a soft alignment between the current position and the input source.

Since then, attention mechanisms are further used. Zhang et al. [20] proposed attention pooling based Convolutional Neural Network (CNN) to represent sentences, which used an intermediate sentence representation generated by the Bidirectional Long Short-Term Memory (BLSTM) as a reference for local representations produced by the convolutional layer to obtain attention weights.

Yang et al. [17] contributed to designing the Hierarchical Attention Network (HAN) for document classification, which has two levels of attention mechanisms applied at the word and sentence level. Of these our model is most closely related to the HAN model, which represents English documents with hierarchical attention mechanism. While we use the combination of character- and word-level attentions to represent Chinese short texts.

In this paper, we design a novel method to detect salient characters and words for separating the texts from different classes.

3 Model

Our model comprises four parts: embedding layer, attention layer, representation layer and classification layer. The overall architecture of the Hybrid Attention Networks (HANs) is shown in Fig. 1. We describe the model in details as follows.

3.1 Short Text Representation

In this work, we apply RNN and CNN to learn Chinese short text representation with the sequences of words and characters respectively.

3.1.1 LSTM-based Representation

The key idea behind RNN is to make use of sequential information. It can map input texts of arbitrary length into fixed size vectors by recursively outputting hidden state vectors h_t that being depended on the previous computations. And yet traditional RNN suffers from the problem of exploding or vanishing gradient, where the gradient vectors can grow or decay exponentially as they propagate to earlier time steps. Consequently, it is difficult to train vanilla RNN to capture long distance dependencies in a sequence. To address this problem, Hochreiter and Schmidhuber [5] proposed the Long Short- Term Memory (LSTM) that is a special type of RNN, which introduced a memory cell (c_t) and three gates (i.e. input gate i_t, forget gate f_t and output gate o_t) to control how the information flows through the network.

However, standard LSTM networks process sequences in temporal order, which cannot capture semantic dependency from future context when predicting the semantic meaning in the beginning or middle of an input text.

Bidirectional LSTM offers an effective way that can access both the preceding and succeeding contexts by involving two separate hidden layers, i.e. one is forward LSTM and the other is backward LSTM.

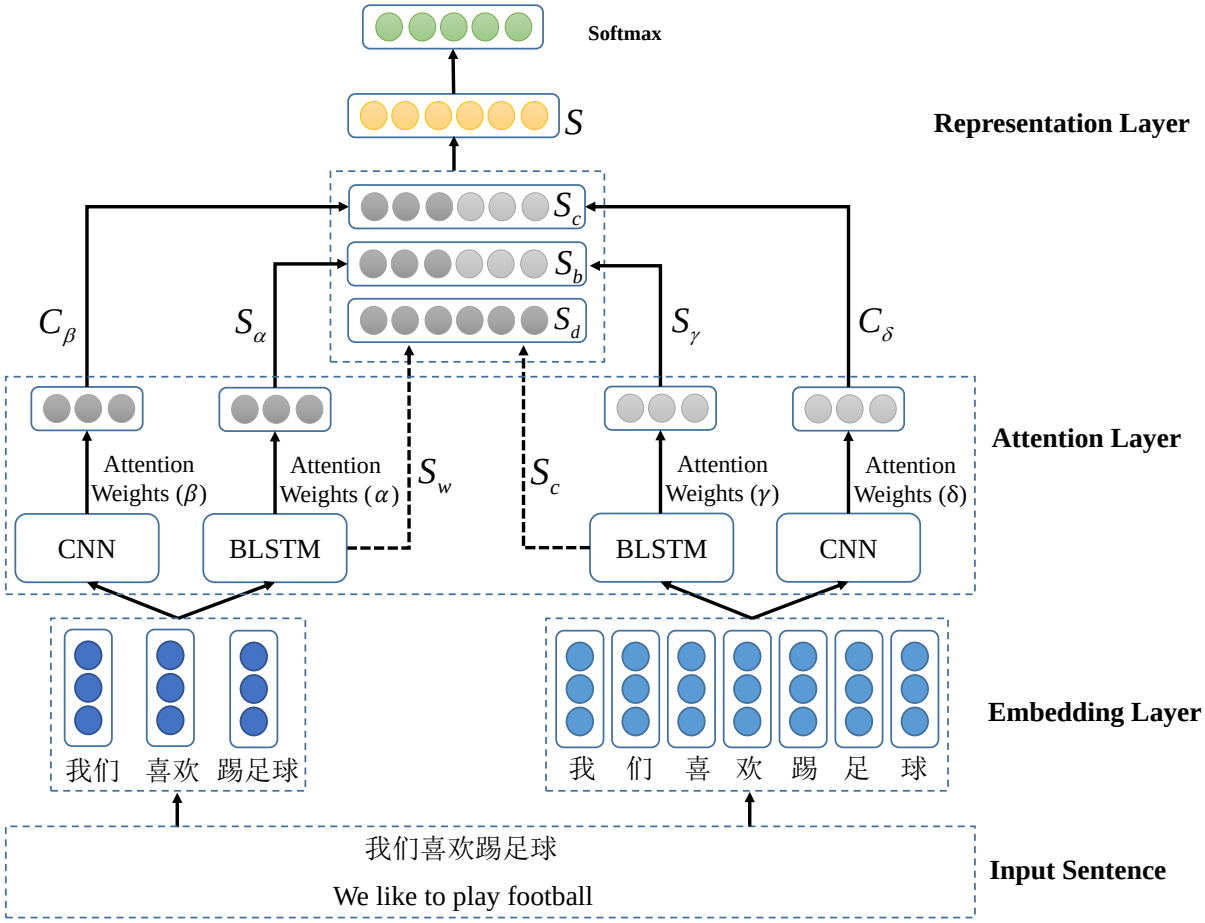

Fig. 1. The architecture of the hybrid attention networks, where α, β, γ, δ are the weights given by word- and character-level attentions, and *S*, *C* indicate the representation of the input short text respectively

Therefore the model is able to capture both past and future contextual information. In the model, an input short text sequence is fed into the forward LSTM layer and the reverse of the input sequence is fed into the backward LSTM layer.

At each time step *t*, the hidden state of the bidirectional LSTM is the concatenation of the forward and backward hidden states (i.e. $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$).

A given Chinese short text is a sequence of words or characters, hence we use LSTM/BLSTM to learn representation for each text with word and character embeddings respectively.

In Fig. 1, the BLSTM component can be substituted by LSTM.

3.1.2 CNN-based Representation

A standard CNN is usually made up of several convolutional and pooling layers. In this work, we design a CNN model with a single convolution layer and an attention-based pooling layer. Using a filter $W_c \in R^{h \times d}$, a convolution operation on *h* consecutive word or character vectors starting from i^{th} that is a concatenation vector $X_{i:i+h-1}$, which represents a slide window with *h* words or characters, outputs features for the input texts in the window by the following Eq. (1), where $X_i \in R^d$ is the word or character embedding, $b_c \in R$ is a bias, *d* is the dimension of embedding vector and *m* is the number of the filters. The operator $\cdot$ stands

for the dot product and *Relu*$(\cdot)$ is the element-wise rectified linear unit function:

$$C_i = Relu(W_c \cdot X_{i:i+h-1} + b_c) \in R^m. \quad (1)$$

We use *m* different filters to perform convolution operation, and denote the resulting feature set for each window as *C*. Given the number of words or characters in a Chinese short text is *l*. For the output length of the convolution layer is equal to the input length, we set the border mode of convolution as 'same'. Hence the *C* can be defined by Eq. (2). Then the features in *C* are fed into the attention-based pooling layer to obtain the short text representation:

$$C = [C_1, C_2, ..., C_l] \in R^{m \times l}. \quad (2)$$

3.2 Hybrid Attention Networks

The idea of attention is inspired by the observation that not all words or characters contribute equally to the representation of the sentence meaning. When reading one sentence, people usually can roughly form an intuition about which parts of the sentence (i.e. several words or characters) are more important. And we implement this idea using attention mechanism in our model from two levels, namely word- and character-level attentions.

3.2.1 Word-level Attention

We design a word-level attention mechanism to focus on the words which have closer semantic relationship to the sentence meaning. Word-level attention architecture is applied to the feature representations of LSTM/BLSTM and CNN. These representations are concatenated together as output of the attention layer.

Attention-based LSTM/BLSTM: Suppose the LSTM layer produces the output vectors $[h_1, h_2, ..., h_l]$. Then the new representation S_α of a Chinese short text is computed by an attention-weighted sum of these output vectors, which is defined as Eq. (3), where $\alpha_i \in R$, stands for the attention weight, just as the Eq. (4) and (5) demonstrate:

$$S_\alpha = \sum_{i=1}^{l} \alpha_i h_i, \quad (3)$$

$$u_i = \tanh(W_h h_i + b_h), \quad (4)$$

$$\alpha_i = softmax(W_\alpha u_i). \quad (5)$$

According to the formulas described above, in our model, we use attention mechanism to get the new representation $\overrightarrow{S_\alpha}$ and $\overleftarrow{S_\alpha}$ for the output of forward and backward LSTM respectively, and obtain the attentive representation S_α of the BLSTM layer by summing $\overrightarrow{S_\alpha}$ and $\overleftarrow{S_\alpha}$.

Fig. 2 describes the architecture of the attention-based BLSTM. By this means, our model can capture as many salient words from both directions as possible.

Attention-based CNN: Given the vectors $[C_1, C_2, ..., C_l]$ are the output of the convolution layer. Just as the attention-based LSTM/BLSTM, we first feed the convolution feature C_i into a $\tanh$ layer to get v_i as a hidden representation of C_i, and get the attention weight β_i that determines the informative convolution features through a softmax function. After that, we compute the pooling vector C_β by a weighted sum of the convolution output based on the attention weights. The attentive representation C_β of those output vectors is computed as follows:

$$v_i = \tanh(W_c C_i + b_c), \quad (6)$$

$$\beta_i = softmax(W_\beta v_i), \quad (7)$$

$$C_\beta = \sum_{i=1}^{l} \beta_i C_i. \quad (8)$$

We use the filters with variable convolution window sizes to form parallel CNNs, which can learn multiple local representations and complement each other to improve the model accuracy. Then we feed the attentive representations produced by all the distinct CNNs through a max function to obtain the final pooling feature vector. That is, $C_\beta = argmax(C_{\beta_k})$, where *k* is the length of convolution window.

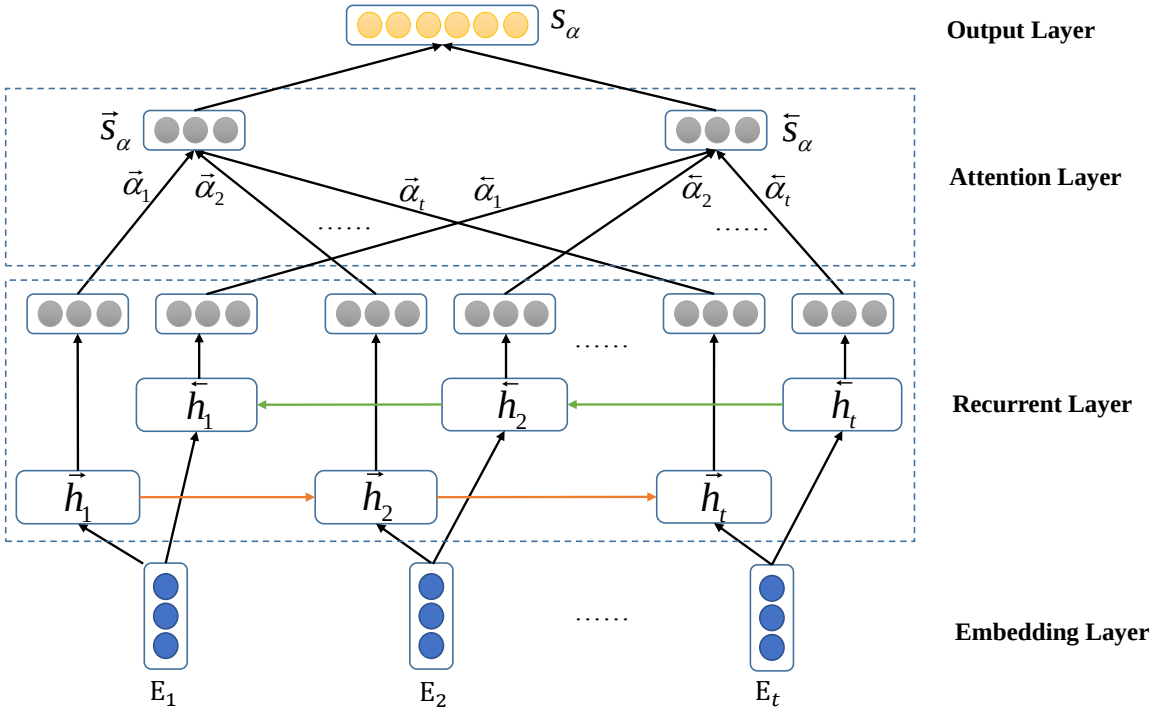

Fig. 2. The architecture of the attention-based BLSTM in our model

3.2.2 Character-level Attention

Word-level attention mechanism is introduced to extract salient words in a Chinese short text. Meanwhile, we use character-level attention to capture informative characters. For an input text, our model first encodes it with character embedding vectors, and then transmits the output to the LSTM/BLSTM and CNN layers simultaneously. Next, our model computes the attentive representations for the outputs of LSTM/BLSTM and CNN layers respectively, the process is the same as word-level attention. Finally, our model produces the new representation S_γ (i.e. the output of the attention-based LSTM/BLSTM) and C_δ (i.e. the output of the attention-based CNN).

3.2.3 The Hybrid of Attention

For BLSTM-based representation, we concatenate the outputs of BLSTM with word- and character-level attentions and get the attentive representation S_b. For CNN-based representation, we also concatenate the outputs of CNN with word- and character-level attentions and get the attentive representation S_c. The output vectors are defined by the following equations:

$$S_b = [S_\alpha, S_\gamma], \tag{9}$$

$$S_c = [C_\beta, C_\delta]. \tag{10}$$

Besides the hidden states, the LSTM/BLSTM layer also produces the last time step output. Hence we concatenate the two last outputs of LSTM/BLSTM with word and character embedding that are defined as S_w and S_c, and get the new output S_d defined by the Eq. (11):

$$S_d = [S_w, S_c]. \tag{11}$$

At last, our model outputs the final sentence vector S as follows:

$$S = [S_d \oplus S_b \oplus S_c]. \tag{12}$$

3.3 Classification

The output of representation layer is a comprehensive feature vector of the original sentence. It then be fed into the classification layer for supervised topic recognition. The classification layer is consist of a linear transformation layer and a softmax layer. That is, the linear layer converts sentence vector *S* to a real-valued vector whose dimension is the number of class, and the softmax layer is added to map each real value to a conditional probability, which is computed by Eq. (13):

$$P = softmax(W_s S + b_s). \tag{13}$$

For training data, each short text has its golden label P^g. We use the following cost function defined by Eq. (14) to minimize the categorical cross-entropy between gold classification probability $P^g(S)$ and predicated classification probability $P(S)$, where N_t is the number of training data, N_c is the class number, P_j^g has a 1-of-K schema whose dimension corresponding to the true class is 1 while all others are 0. During training phrase, all the parameters including weights and bias terms in the attention layer are determined and updated by the model. Word and character embeddings are also fine-tuned as well. Optimization is performed using Stochastic Gradient Descent (SGD):

$$Loss = -\sum_{i=1}^{N_t}\sum_{j=1}^{N_c} P_j^g(s_i) \cdot log(P_j(s_i)). \tag{14}$$

4 Experiments

We evaluate the effectiveness of our model for multi-class Chinese short text classification on two representative types of data sets, one is nearly formal short text (i.e. Chinese news titles) and the other is informal (i.e. Chinese Douban movie reviews and Weibo rumors). The detail of experimental setup, results and analysis is as follows.

4.1 Datasets

We conduct experiments on three datasets as follows:

Chinese news titles are obtained from [21]. This is a topic classification with 32 classes. There are 47,850 training samples and 15,950 testing samples.

Chinese Douban movie reviews are obtained from Douban[1]. We tag 84,634 samples with gold standard classification labels. 56,945 samples are used for training and 27,689 samples are used for testing. There are five levels of ratings from 1 to 5 (higher is better).

Weibo rumors are obtained from [13]. There are 9,079 Weibo rumors. After deleting these records with 'x' label, we remain 5 classes, and 6,031 samples for training and 2,159 samples for testing.

The above three datasets all have no validation samples. We randomly choose 10 percent of the training samples for validation. The statistics of the data sets are summarized in Table 1.

4.2 Baselines

Most neural methods used to text classification are variants of convolutional or recurrent networks. We select five neural network baselines including CNN, RNN, RCNN, FastText [6] and C-LSTMs/BLSTMs[21].

CNN: We report the CNN baselines from [21], which implemented the model[8] and conducted experiments with word and character embeddings respectively (i.e. CNN-char and CNN-word).

RNN: We also report the RNN baselines from [21], which implemented a single layer recurrent neural network with LSTM and bidirectional LSTM, and conducted experiments with word and character embeddings respectively (i.e. LSTM-char, LSTM-word, BLSTM-char and BLSTM-word).

RCNN: Lai et al. [9] developed a two-layer neural model, the first layer applied a bi-directional recurrent structure to learn text representation with word embedding, and the second layer used a max-pooling mechanism to choose the salient features in the texts. We implement the

[1] https://movie.douban.com

Table 1. Dataset overview. $|W|$ is the word vocabulary size of each data set. $|C|$ is the character vocabulary size of each data set. #L denotes the length of short text, including average and maximum per sentence

Data Set	Class	$\|W\|$	$\|C\|$	Training	Test	Avg. #L	Max #L
Chinese news titles	32	62,167	4,721	47,850	15,950	18	64
Douban movie reviews	5	85,770	6,027	56,945	27,689	89	100
Weibo rumors	5	19,324	3,805	6,031	2,159	128	201

RCNN baseline with LSTM-RNNs instead of vanilla RNNs, and experiment with word and character embeddings respectively (i.e. RCNN-char and RCNN-word).

FastText: Joulin et al. [6] proposed a fast text classification method based on word embedding, which averaged all of the word representations in each text into a vector representation that was in turn fed to a hierarchical softmax layer to classify the texts. We implement the baseline to classify Chinese short texts, and also experiment with word and character embeddings respectively (i.e. FastText-char and FastText-word).

C-LSTMs/BLSTMs: Zhou et al. [21] introduced a compositional recurrent neural networks with LSTM or BLSTM, which used word and character embeddings to learn a Chinese short text representation respectively, and then concatenated the two vectors to construct a sentence representation for classification. The results were reported in [21].

4.3 Experimental Setup

Note that we directly use the word and character embeddings from [21]. We apply Jieba[2] to conduct Chinese word segmentation on the input text, and initialize the lookup tables of input texts with the 100-dimensional word and character embeddings respectively. The hyperparameters of our model are tuned on the validation set and early stopping is used with 10 epoches. Dropout rate of 0.4 is applied to obtain better performance. We use stochastic gradient descent to train all models with learning rate of 0.01 and momentum of 0.9. Table 2 shows the hyper parameter settings.

[2]https://github.com/fxsjy/jieba

4.4 Results

Table 3 shows a detailed comparison of our model with the baselines. From Rows 1 to 6, the results show that the performances of CNN-word and RNN-word are better than CNN-char and RNN-char respectively. These results indicate that word embedding is more effective than character embedding and RNN with LSTM or BLSTM is also more effective than the CNN model. The results of Rows 11 and 12 prove that the combination of word and character embeddings can improve performance further. The non-attentive networks C-LSTMs/BLSTMs already perform better than other baselines. If we add the attentive mechanism, then the performance on all data sets improves further better than all baseline methods.

In Fig. 1, our model concatenates the outputs of the word- and character-level LSTM/BLSTM layers, which are equivalent to the C-LSTMs/BLSTMs model. From Rows 11 and 12, it is observed that C-LSTMs/BLSTMs model achieves differentiated results on three datasets respectively. For each data set, we choose the LSTM or BLSTM model that gets the best performance, which is the basis of our model. That is to say, for non-attentive representation, our model concatenates the word- and character-level outputs of the LSTM on the datasets of News Title, and concatenates the outputs of BLSTM on Douban Movie Review and Weibo Rumor.

On the basis, we further investigate the effect of our model on three datasets by augmenting different attentive approaches. In Table 3, from Row 13 to 14, it shows that the results of HANs-LSTM are better than HANs-CNN. While combined with the attention-based CNN and LSTM layer, HANs-LSTM+CNN obtains further

Table 2. Experimental ranges and choices of hyper parameters in our model

Parameter	Choice	Experiment Range
Word embedding dimension	100	50, 100, 300
Character embedding dimension	100	50, 100, 300
LSTM/CNN hidden layer size	100	64, 100, 128, 256
CNN number of filters	100	64, 100, 128, 256
CNN filter height	2-5	2-7
Dropout rate	0.4	0.4, 0.5
Epoch size	10	10, 15, 20
Mini-batch size	8	8, 16, 32, 64

Table 3. Results in percent of weighted-average on Chinese news title, Douban movie review and Weibo Rumor. Best result per column is bold. HANs- stands for different attention layers in our model. **HANs-CNN** denotes the model with only the attention-based CNN. **HANs-LSTM** denotes the model with only the attention-based LSTM. **HANs-BLSTM** denotes the model with only the attention-based BLSTM. **HANs-LSTM+CNN** denotes the model with the combination of the attention-based LSTM and CNN. **HANs-BLSTM+CNN** denotes the model with the combination of the attention-based BLSTM and CNN

Methods	F_1 (Precision, Recall)		
	News Title	Movie Review	Weibo Rumor
CNN-char [21]	74.0 (74.2, 74.2)	9.2 (5.7, 23.9)	15.0 (28.6, 23.0)
CNN-word [21]	76.8 (77.8, 76.8)	35.4 (43.2, 35.5)	77.8 (79.2, 78.1)
LSTM-char [21]	77.3 (77.8, 77.4)	11.7 (26.9, 24.5)	17.6 (14.4, 26.4)
LSTM-word [21]	80.0 (80.2, 80.1)	37.3 (44.6, 35.8)	78.3 (78.9, 78.4)
BLSTM-char [21]	77.4 (77.7, 77.4)	35.6 (39.5, 36.5)	78.6 (79.7, 78.6)
BLSTM-word [21]	80.0 (80.4, 79.9)	39.5 (42.6, 38.6)	82.8 (85.4, 83.0)
RCNN-char [9]	76.7 (76.9, 76.9)	36.4 (38.0, 37.1)	80.4 (81.3, 80.5)
RCNN-word [9]	78.8 (79.5, 78.8)	40.1 (44.3, 39.5)	82.5 (84.3, 82.3)
FastText-char [6]	77.3 (77.4, 77.4)	34.0 (40.1, 33.1)	79.3 (80.0, 79.3)
FastText-word [6]	78.5 (78.8, 78.5)	38.3 (42.6, 37.3)	80.2 (80.8, 80.1)
C-LSTMs [21]	82.1 (82.2, 82.1)	38.4 (45.1, 37.3)	81.0 (82.5, 81.3)
C-BLSTMs [21]	81.9 (82.1, 82.1)	39.5 (44.6, 38.2)	86.0 (86.8, 86.0)
HANs-CNN	81.9 (82.2, 81.8)	37.8 (44.5, 37.2)	81.4 (81.7, 81.5)
HANs-LSTM	82.3 (82.9, 82.4)	41.3 (46.3, 42.0)	86.4 (87.3, 86.3)
HANs-LSTM+CNN	82.5 (82.8, 82.5)	40.8 (45.3, 39.8)	83.8 (84.0, 83.8)
HANs-BLSTM	82.7 (83.0, 82.6)	**41.6** (44.0, 42.4)	**87.9** (88.2, 87.9)
HANs-BLSTM+CNN	**83.1** (83.2, 83.2)	41.0 (45.7, 40.0)	84.1 (85.1, 84.1)

better performance on two datasets. Row 16 implies that HANs-BLSTM can continue to improve performance. And we observe that HANs-BLSTM+CNN achieves best results on News Title, while lowers the performances on Movie Review and Weibo Rumor.

4.5 Analysis

These experimental results agree well with the findings of the effectiveness of attentive mechanism. For Chinese short text, we explore the attentive neural networks from two levels, that is, word- and character-level. Experimental results prove that our model is effective for the representation of Chinese short text and obtains better classification performance.

The experimental results from Rows 14 to 17 reveal that HANs-BLSTM is better than HANs-LSTM and HANs-BLSTM+CNN is also better than HANs-LSTM+CNN. These comparisons indicates that the BLSTM-based attention is more effective than the LSTM-based attention. The reason is that the BLSTM-based attention can capture more semantic information from both directions than the LSTM-based attention. In Fig. 2, our model firstly computes the attentive representation for the forward and backward LSTM layer respectively, then sum the two outputs to generate the representation of the input short text. Therefore the addition of the attentive representation from both directions picks out informative characters and words.

For Movie Review and Weibo Rumor, Rows 16 and 17 imply that adding CNN-based attention does not help and may even lower the performances. It may be the reason that several discrete words or characters determine the semantic of each Chinese short text in above two datasets, and the distances among these salient words or characters are long while CNN can only capture the semantic in the range of convolution window.

In all of the baselines, C-LSTMs/BLSTMs model almost achieves the best result relatively. However, comparison with the best results of all baselines, our model HANs-BLSTM (+CNN) further improves the F_1-score by 1.0%, 1.5% and 1.9% on three datasets respectively.

5 Conclusion

In this paper, we propose a Hybrid Attention Networks (HANs) for Chinese short text classification, which aggregates important characters and words into sentence vectors respectively and merges them into one representative vector. Experimental results demonstrate that our model outperforms all of the baselines on two typical types of Chinese short text (i.e. regular and user-generated short text). Also it shows that our model is effective to select informative semantic units from a Chinese short text. In the future, we intend to explore our model in the representation of longer document.

Acknowledgements

This work is supported by the National Natural Science Foundation (No. 61602479), National High Technology Research and Development Program of China (No. 2015AA015402) and National Key Technology R&D Program of China under No. 2015BAH53F02.

References

1. **Bahdanau, D., Cho, K., & Bengio, Y. (2014).** Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*.
2. **Cabrera, J. M., Escalante, H. J., & Montes-y Gómez, M. (2013).** Distributional Term Representations for Short-Text Categorization. *CICLing 2013*, pp. 335–346.
3. **Chen, M., Jin, X., & Shen, D. (2011).** Short Text Classification Improved by Learning Multi-Granularity Topics. *IJCAI*, pp. 1776–1781.
4. **Feng, X., Shen, Y., Liu, C., Liang, W., & Zhang, S. (2013).** Chinese Short Text Classification Based on Domain Knowledge. *IJCNLP 2013*, pp. 859–863.
5. **Hochreiter, S. & Schmidhuber, J. (1997).** Long short-term memory. *Neural computation*, Vol. 9, No. 8, pp. 1735–1780.
6. **Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016).** Bag of Tricks for Efficient Text Classification. *CoRR*, Vol. abs/1607.01759.

7. **Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014).** A Convolutional Neural Network for Modelling Sentences. *Proceedings of the 52nd ACL*, pp. 655–665.
8. **Kim, Y. (2014).** Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on EMNLP*, pp. 1746–1751.
9. **Lai, S., Xu, L., Liu, K., & Zhao, J. (2015).** Recurrent Convolutional Neural Networks for Text Classification. *AAAI 2015*, pp. 2267–2273.
10. **Le, Q. V. & Mikolov, T. (2014).** Distributed Representations of Sentences and Documents. *ICML 2014*, pp. 1188–1196.
11. **Li, Y., Li, W., Sun, F., & Li, S. (2015).** Component-Enhanced Chinese Character Embeddings. *Proceedings of the 2015 Conference on EMNLP*, pp. 829–834.
12. **Liu, P., Qiu, X., Chen, X., Wu, S., & Huang, X. (2015).** Multi-Timescale Long Short-Term Memory Neural Network for Modelling Sentences and Documents. *Proceedings of the 2015 Conference on EMNLP*, pp. 2326–2335.
13. **Liu, Z. Y., Zhang, L., Cunchao, T. U., & Sun, M. S. (2015).** Statistical and semantic analysis of rumors in chinese social media. *Scientia Sinica*, Vol. 45, No. 12.
14. **Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013).** Efficient Estimation of Word Representations in Vector Space. *CoRR*, Vol. abs/1301.3781.
15. **Socher, R., Huval, B., Manning, D. C., & Ng, Y. A. (2012).** Semantic Compositionality through Recursive Matrix-Vector Spaces. *Proceedings of the 2012 EMNLP*, Association for Computational Linguistics, Jeju Island, Korea, pp. 1201–1211.
16. **Song, G., Ye, Y., Du, X., Huang, X., & Bie, S. (2014).** Short Text Classification: A Survey. *Journal of Multimedia*, Vol. 9, No. 5, pp. 635–643.
17. **Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016).** Hierarchical attention networks for document classification. *NAACL HLT 2016*.
18. **Zelikovitz, S. & Marquez, F. (2005).** Transductive Learning For Short-Text Classification Problems Using Latent Semantic Indexing. *IJPRAI*, Vol. 19, No. 2, pp. 143–163.
19. **Zhang, X., Zhao, J., & LeCun, Y. (2015).** Character-level Convolutional Networks for Text Classification. *NIPS 2015*, pp. 649–657.
20. **Zhang, Y., Er, M. J., Wang, N., & Pratama, M. (2016).** Attention Pooling-based Convolutional Neural Network for Sentence Modelling. *Information Sciences*, Vol. 373, pp. 388–403.
21. **Zhou, Y., Xu, B., Xu, J., Yang, L., Li, C., & Xu, B. (2016).** Compositional recurrent neural networks for chinese short text classification. *2016 IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 137–144.

Article received on 19/12/2016; accepted on 28/02/2017.
Corresponding author is Yujun Zhou.