Recurrent Convolutional Neural Networks for Text Classification

Siwei Lai, Liheng Xu, Kang Liu, Jun Zhao
National Laboratory of Pattern Recognition (NLPR)
Institute of Automation, Chinese Academy of Sciences, China
{swlai, lhxu, kliu, jzhao}@nlpr.ia.ac.cn

Abstract

Text classification is a foundational task in many NLP applications. Traditional text classifiers often rely on many human-designed features, such as dictionaries, knowledge bases and special tree kernels. In contrast to traditional methods, we introduce a recurrent convolutional neural network for text classification without human-designed features. In our model, we apply a recurrent structure to capture contextual information as far as possible when learning word representations, which may introduce considerably less noise compared to traditional window-based neural networks. We also employ a max-pooling layer that automatically judges which words play key roles in text classification to capture the key components in texts. We conduct experiments on four commonly used datasets. The experimental results show that the proposed method outperforms the state-of-the-art methods on several datasets, particularly on document-level datasets.

Introduction

Text classification is an essential component in many applications, such as web searching, information filtering, and sentiment analysis (Aggarwal and Zhai 2012). Therefore, it has attracted considerable attention from many researchers.

A key problem in text classification is feature representation, which is commonly based on the bag-of-words (BoW) model, where unigrams, bigrams, n-grams or some exquisitely designed patterns are typically extracted as features. Furthermore, several feature selection methods, such as frequency, MI (Cover and Thomas 2012), pLSA (Cai and Hofmann 2003), LDA (Hingmire et al. 2013), are applied to select more discriminative features. Nevertheless, traditional feature representation methods often ignore the contextual information or word order in texts and remain unsatisfactory for capturing the semantics of the words. For example, in the sentence "*A sunset stroll along the South Bank affords an array of stunning vantage points.*", when we analyze the word "Bank" (unigram), we may not know whether it means a financial institution or the land beside a river. In addition, the phrase "South Bank" (bigram), particularly considering the two uppercase letters, may mislead people who are not particularly knowledgeable about London to take it as a financial institution. After we obtain the greater context "stroll along the South Bank" (5-gram), we can easily distinguish the meaning. Although high-order n-grams and more complex features (such as tree kernels (Post and Bergsma 2013)) are designed to capture more contextual information and word orders, they still have the data sparsity problem, which heavily affects the classification accuracy.

Recently, the rapid development of pre-trained word embedding and deep neural networks has brought new inspiration to various NLP tasks. Word embedding is a distributed representation of words and greatly alleviates the data sparsity problem (Bengio et al. 2003). Mikolov, Yih, and Zweig (2013) shows that pre-trained word embeddings can capture meaningful syntactic and semantic regularities. With the help of word embedding, some composition-based methods are proposed to capture the semantic representation of texts.
cs224d Socher et al. (2011a; 2011b; 2013) proposed the **Recursive Neural Network** (RecursiveNN) that has been proven to be efficient in terms of constructing sentence representations. However, the RecursiveNN captures the semantics of a sentence via a tree structure. Its performance heavily depends on the performance of the textual tree construction. Moreover, constructing such a textual tree exhibits a time complexity of at least $O(n^2)$, where n is the length of the text. This would be too time-consuming when the model meets a long sentence or a document. Furthermore, the relationship between two sentences can hardly be represented by a tree structure. Therefore, RecursiveNN is unsuitable for modeling long sentences or documents.

Another model, which only exhibits a time complexity $O(n)$, is the **Recurrent Neural Network** (RecurrentNN). This model analyzes a text word by word and stores the semantics of all the previous text in a fixed-sized hidden layer (Elman 1990). The advantage of RecurrentNN is the ability to better capture the contextual information. This could be beneficial to capture semantics of long texts. However, the RecurrentNN is a biased model, where later words are more dominant than earlier words. Thus, it could reduce the effectiveness when it is used to capture the semantics of a whole document, because key components could appear anywhere in a document rather than at the end.

To tackle the bias problem, the **Convolutional Neural Network** (CNN), an unbiased model is introduced to NLP tasks, which can fairly determine discriminative phrases in a text with a max-pooling layer. Thus, the CNN may better capture the semantic of texts compared to recursive or recurrent neural networks. The time complexity of the CNN is also $O(n)$. However, previous studies on CNNs tends to use simple convolutional kernels such as a fixed window (Collobert et al. 2011; Kalchbrenner and Blunsom 2013). When using such kernels, it is difficult to determine the window size: small window sizes may result in the loss of some critical information, whereas large windows result in an enormous parameter space (which could be difficult to train). Therefore, it raises a question: can we learn more contextual information than conventional window-based neural networks and represent the semantic of texts more precisely for text classification.

To address the limitation of the above models, we propose a Recurrent Convolutional Neural Network (RCNN) and apply it to the task of text classification. First, we apply a bi-directional recurrent structure, which may introduce considerably less noise compared to a traditional window-based neural network, to capture the contextual information to the greatest extent possible when learning word representations. Moreover, the model can reserve a larger range of the word ordering when learning representations of texts. Second, we employ a max-pooling layer that automatically judges which features play key roles in text classification, to capture the key component in the texts. By combining the recurrent structure and max-pooling layer, our model utilizes the advantage of both recurrent neural models and convolutional neural models. Furthermore, our model exhibits a time complexity of $O(n)$, which is linearly correlated with the length of the text length.

We compare our model with previous state-of-the-art approaches using four different types of tasks in English and Chinese. The classification taxonomy contains topic classification, sentiment classification and writing style classification. The experiments demonstrate that our model outperforms previous state-of-the-art approaches in three of the four commonly used datasets.

Related Work

Text Classification

Traditional text classification works mainly focus on three topics: feature engineering, feature selection and using different types of machine learning algorithms. For feature engineering, the most widely used feature is the bag-of-words feature. In addition, some more complex features have been designed, such as part-of-speech tags, noun phrases (Lewis 1992) and tree kernels (Post and Bergsma 2013). Feature selection aims at deleting noisy features and improving the classification performance. The most common feature selection method is removing the stop words (e.g., “the”). Advanced approaches use information gain, mutual information (Cover and Thomas 2012), or L1 regularization (Ng 2004) to select useful features. Machine learning algorithms often use classifiers such as logistic regression (LR), naive Bayes (NB), and support vector machine (SVM). However, these methods have the data sparsity problem.

Deep neural networks

Recently, deep neural networks (Hinton and Salakhutdinov 2006) and representation learning (Bengio, Courville, and Vincent 2013) have led to new ideas for solving the data sparsity problem, and many neural models for learning word representations have been proposed (Bengio et al. 2003; Mnih and Hinton 2007; Mikolov 2012; Collobert et al. 2011; Huang et al. 2012; Mikolov et al. 2013). The neural representation of a word is called word embedding and is a real-valued vector. The word embedding enables us to measure word relatedness by simply using the distance between two embedding vectors.

With the pre-trained word embeddings, neural networks demonstrate their great performance in many NLP tasks. Socher et al. (2011b) use semi-supervised recursive autoencoders to predict the sentiment of a sentence. Socher et al. (2011a) proposed a method for paraphrase detection also with recurrent neural network. Socher et al. (2013) introduced recursive neural tensor network to analyse sentiment of phrases and sentences. Mikolov (2012) uses recurrent neural network to build language models. Kalchbrenner and Blunsom (2013) proposed a novel recurrent network for dialogue act classification. Collobert et al. (2011) introduce convolutional neural network for semantic role labeling.

Model

We propose a deep neural model to capture the semantics of the text. Figure 1 shows the network structure of our model. The input of the network is a document D, which is a sequence of words $w_1, w_2 \ldots w_n$. The output of the network contains *class* elements. We use $p(k|D, \theta)$ to denote the probability of the document being class k, where θ is the parameters in the network.

Word Representation Learning

We combine a word and its context to present a word. The contexts help us to obtain a more precise word meaning. In our model, we use a recurrent structure, which is a bi-directional recurrent neural network, to capture the contexts.

We define $\boldsymbol{c}_l(w_i)$ as the left context of word w_i and $\boldsymbol{c}_r(w_i)$ as the right context of word w_i. Both $\boldsymbol{c}_l(w_i)$ and $\boldsymbol{c}_r(w_i)$ are dense vectors with $|\boldsymbol{c}|$ real value elements. The left-side context $\boldsymbol{c}_l(w_i)$ of word w_i is calculated using Equation (1), where $\boldsymbol{e}(w_{i-1})$ is the word embedding of word w_{i-1}, which is a dense vector with $|\boldsymbol{e}|$ real value elements. $\boldsymbol{c}_l(w_{i-1})$ is the left-side context of the previous word w_{i-1}. The left-side context for the first word in any document uses the same shared parameters $\boldsymbol{c}_l(w_1)$. $W^{(l)}$ is a matrix that transforms the hidden layer (context) into the next hidden layer. $W^{(sl)}$ is a matrix that is used to combine the semantic of the current word with the next word's left context. f is a non-linear activation function. The right-side context $\boldsymbol{c}_r(w_i)$ is calculated in a similar manner, as shown in Equation (2). The right-side contexts of the last word in a document share the parameters $\boldsymbol{c}_r(w_n)$.

$$\boldsymbol{c}_l(w_i) = f(W^{(l)}\boldsymbol{c}_l(w_{i-1}) + W^{(sl)}\boldsymbol{e}(w_{i-1})) \qquad (1)$$

$$\boldsymbol{c}_r(w_i) = f(W^{(r)}\boldsymbol{c}_r(w_{i+1}) + W^{(sr)}\boldsymbol{e}(w_{i+1})) \qquad (2)$$

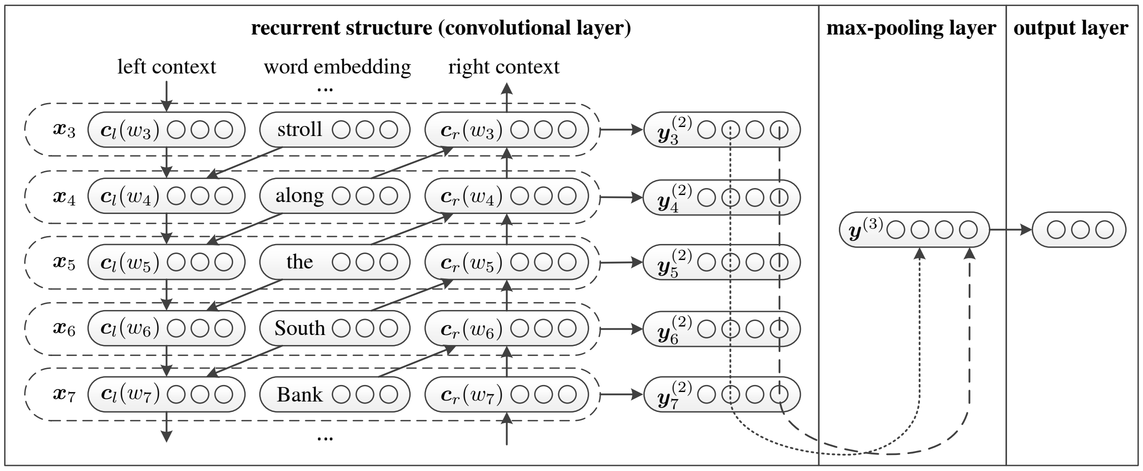
recurrent structure (convolutional layer)
max-pooling layer
output layer
left context
word embedding
right context
x_3 $c_l(w_3)$ stroll $c_r(w_3)$ $y_3^{(2)}$
x_4 $c_l(w_4)$ along $c_r(w_4)$ $y_4^{(2)}$
x_5 $c_l(w_5)$ the $c_r(w_5)$ $y_5^{(2)}$
x_6 $c_l(w_6)$ South $c_r(w_6)$ $y_6^{(2)}$
x_7 $c_l(w_7)$ Bank $c_r(w_7)$ $y_7^{(2)}$
$y^{(3)}$

Figure 1: The structure of the recurrent convolutional neural network. This figure is a partial example of the sentence "A sunset stroll along the South Bank affords an array of stunning vantage points", and the subscript denotes the position of the corresponding word in the original sentence.

various lengths?? x_i ?

As shown in Equations (1) and (2), the context vector captures the semantics of all left- and right-side contexts. For example, in Figure 1, $\boldsymbol{c}_l(w_7)$ encodes the semantics of the left-side context "stroll along the South" along with all previous texts in the sentence, and $\boldsymbol{c}_r(w_7)$ encodes the semantics of the right-side context "affords an ...". Then, we define the representation of word w_i in Equation (3), which is the concatenation of the left-side context vector $\boldsymbol{c}_l(w_i)$, the word embedding $\boldsymbol{e}(w_i)$ and the right-side context vector $\boldsymbol{c}_r(w_i)$. In this manner, using this contextual information, our model may be better able to disambiguate the meaning of the word w_i compared to conventional neural models that only use a fixed window (i.e., they only use partial information about texts).

$$\boldsymbol{x}_i = [\boldsymbol{c}_l(w_i); \boldsymbol{e}(w_i); \boldsymbol{c}_r(w_i)] \tag{3}$$

The recurrent structure can obtain all $\boldsymbol{c}_l$ in a forward scan of the text and $\boldsymbol{c}_r$ in a backward scan of the text. The time complexity is $O(n)$. After we obtain the representation $\boldsymbol{x}_i$ of the word w_i, we apply a linear transformation together with the $\tanh$ activation function to $\boldsymbol{x}_i$ and send the result to the next layer.

$$\boldsymbol{y}_i^{(2)} = \tanh\left(W^{(2)}\boldsymbol{x}_i + \boldsymbol{b}^{(2)}\right) \tag{4}$$

$\boldsymbol{y}_i^{(2)}$ is a latent semantic vector, in which each semantic factor will be analyzed to determine the most useful factor for representing the text.

Text Representation Learning

The convolutional neural network in our model is designed to represent the text. From the perspective of convolutional neural networks, the recurrent structure we previously mentioned is the convolutional layer.

When all of the representations of words are calculated, we apply a max-pooling layer.

$$\boldsymbol{y}^{(3)} = \max_{i=1}^{n} \boldsymbol{y}_i^{(2)} \tag{5}$$

The $\max$ function is an element-wise function. The k-th element of $\boldsymbol{y}^{(3)}$ is the maximum in the k-th elements of $\boldsymbol{y}_i^{(2)}$.

The pooling layer converts texts with various lengths into a fixed-length vector. With the pooling layer, we can capture the information throughout the entire text. There are other types of pooling layers, such as average pooling layers (Collobert et al. 2011). We do not use average pooling here because only a few words and their combination are useful for capturing the meaning of the document. The max-pooling layer attempts to find the most important latent semantic factors in the document. The pooling layer utilizes the output of the recurrent structure as the input. The time complexity of the pooling layer is $O(n)$. The overall model is a cascade of the recurrent structure and a max-pooling layer, therefore, the time complexity of our model is still $O(n)$.

The last part of our model is an output layer. Similar to traditional neural networks, it is defined as

$$\boldsymbol{y}^{(4)} = W^{(4)}\boldsymbol{y}^{(3)} + \boldsymbol{b}^{(4)} \tag{6}$$

Finally, the softmax function is applied to $\boldsymbol{y}^{(4)}$. It can convert the output numbers into probabilities.

$$p_i = \frac{\exp\left(\boldsymbol{y}_i^{(4)}\right)}{\sum_{k=1}^{n}\exp\left(\boldsymbol{y}_k^{(4)}\right)} \tag{7}$$

Training

Training Network parameters We define all of the parameters to be trained as θ.

$$\theta = \{E, \boldsymbol{b}^{(2)}, \boldsymbol{b}^{(4)}, \boldsymbol{c}_l(w_1), \boldsymbol{c}_r(w_n), W^{(2)}, \\ W^{(4)}, W^{(l)}, W^{(r)}, W^{(sl)}, W^{(sr)}\} \tag{8}$$

Specifically, the parameters are word embeddings $E \in \mathbb{R}^{|\boldsymbol{e}|\times|V|}$, the bias vectors $\boldsymbol{b}^{(2)} \in \mathbb{R}^H$, $\boldsymbol{b}^{(4)} \in \mathbb{R}^O$, the initial contexts $\boldsymbol{c}_l(w_1), \boldsymbol{c}_r(w_n) \in \mathbb{R}^{|\boldsymbol{c}|}$ and the transformation matrixes $W^{(2)} \in \mathbb{R}^{H\times(|\boldsymbol{e}|+2|\boldsymbol{c}|)}$, $W^{(4)} \in \mathbb{R}^{O\times H}$, $W^{(l)}, W^{(r)} \in$

$\mathbb{R}^{|\boldsymbol{c}|\times|\boldsymbol{c}|}, W^{(sl)}, W^{(sr)} \in \mathbb{R}^{|\boldsymbol{e}|\times|\boldsymbol{c}|}$, where $|V|$ is the number of words in the vocabulary, H is the hidden layer size, and O is the number of document types.

The training target of the network is used to maximize the log-likelihood with respect to θ:

$$\theta \mapsto \sum_{D\in\mathbb{D}} \log p(class_D|D,\theta) \tag{9}$$

where $\mathbb{D}$ is the training document set and $class_D$ is the correct class of document D.

We use stochastic gradient descent (Bottou 1991) to optimize the training target. In each step, we randomly select an example $(D, class_D)$ and make a gradient step.

$$\theta \leftarrow \theta + \alpha \frac{\partial \log p(class_D|D,\theta)}{\partial\theta} \tag{10}$$

where α is the learning rate.

We use one trick that is widely used when training neural networks with stochastic gradient descent in the training phase. We initialize all of the parameters in the neural network from a uniform distribution. The magnitude of the maximum or minimum equals the square root of the "fan-in"(Plaut and Hinton 1987). The number is the network node of the previous layer in our model. The learning rate for that layer is divided by "fan-in".

Pre-training Word Embedding Word embedding is a distributed representation of a word. Distributed representation is suitable for the input of neural networks. Traditional representations, such as one-hot representation, will lead to the curse of dimensionality (Bengio et al. 2003). Recent research (Hinton and Salakhutdinov 2006; Erhan et al. 2010) shows that neural networks can converge to a better local minima with a suitable unsupervised pre-training procedure.

In this work, we use the Skip-gram model to pre-train the word embedding. this model is the state-of-the-art in many NLP tasks (Baroni, Dinu, and Kruszewski 2014). The Skip-gram model trains the embeddings of words $w_1, w_2 \ldots w_T$ by maximizing the average log probability

$$\frac{1}{T}\sum_{t=1}^{T} \sum_{-c\leq j\leq c, j\neq 0} \log p(w_{t+j}|w_t) \tag{11}$$

$$p(w_b|w_a) = \frac{\exp\left(\boldsymbol{e}'(w_b)^T \boldsymbol{e}(w_a)\right)}{\sum_{k=1}^{|V|} \exp\left(\boldsymbol{e}'(w_k)^T \boldsymbol{e}(w_a)\right)} \tag{12}$$

where $|V|$ is the vocabulary of the unlabeled text. $\boldsymbol{e}'(w_i)$ is another embedding for w_i. We use the embedding $\boldsymbol{e}$ because some speed-up approaches (e.g., hierarchical softmax (Morin and Bengio 2005)) will be used here, and $\boldsymbol{e}'$ is not calculated in practice.

Experiments

Datasets

To demonstrate the effectiveness of the proposed method, we perform the experiments using the following four datasets: 20Newsgroups, Fudan Set, ACL Anthology Network, and Sentiment Treebank. Table 1 provides detailed information about each dataset.

Dataset	C	Train/Dev/Test	Len	Lang
20News	4	7520/836/5563	429	EN
Fudan	20	8823/981/9832	2981	CH
ACL	5	146257/28565/28157	25	EN
SST	5	8544/1101/2210	19	EN

Table 1: A summary of the datasets, including the number of classes, the number of train/dev/test set entries, the average text length and the language of the dataset.

20Newsgroups[1] This dataset contains messages from twenty newsgroups. We use the *bydate* version and select four major categories (comp, politics, rec, and religion) followed by Hingmire et al. (2013).

Fudan set[2] The Fudan University document classification set is a Chinese document classification set that consists of 20 classes, including art, education, and energy.

ACL Anthology Network[3] This dataset contains scientific documents published by the ACL and by related organizations. It is annotated by Post and Bergsma (2013) with the five most common native languages of the authors: English, Japanese, German, Chinese, and French.

Stanford Sentiment Treebank[4] The dataset contains movie reviews parsed and labeled by Socher et al. (2013). The labels are Very Negative, Negative, Neutral, Positive, and Very Positive.

Experiment Settings

We preprocess the dataset as follows. For English documents, we use the Stanford Tokenizer[5] to obtain the tokens. For Chinese documents, we use ICTCLAS[6] to segment the words. We do not remove any stop words or symbols in the texts. All four datasets have been previously separated into training and testing sets. The ACL and SST datasets have a pre-defined training, development and testing separation. For the other two datasets, we split 10% of the training set into a development set and keep the remaining 90% as the real training set. The evaluation metric of the 20Newsgroups is the Macro-F1 measure followed by the state-of-the-art work. The other three datasets use accuracy as the metric.

The hyper-parameter settings of the neural networks may depend on the dataset being used. We choose one set of commonly used hyper-parameters following previous studies (Collobert et al. 2011; Turian, Ratinov, and Bengio 2010). Moreover, we set the learning rate of the stochastic gradient descent α as 0.01, the hidden layer size as $H = 100$, the vector size of the word embedding as $|\boldsymbol{e}| = 50$ and the size of the context vector as $|\boldsymbol{c}| = 50$. We train word em-

[1]qwone.com/~jason/20Newsgroups/
[2]www.datatang.com/data/44139 and 43543
[3]old-site.clsp.jhu.edu/~sbergsma/Stylo/
[4]nlp.stanford.edu/sentiment/
[5]nlp.stanford.edu/software/tokenizer.shtml
[6]ictclas.nlpir.org

Model	20News	Fudan	ACL	SST
BoW + LR	92.81	92.08	46.67	40.86
Bigram + LR	93.12	92.97	47.00	36.24
BoW + SVM	92.43	93.02	45.24	40.70
Bigram + SVM	92.32	93.03	46.14	36.61
Average Embedding	89.39	86.89	41.32	32.70
ClassifyLDA-EM (Hingmire et al. 2013)	93.60	-	-	-
Labeled-LDA (Li, Sun, and Zhang 2008)	-	90.80	-	-
CFG (Post and Bergsma 2013)	-	-	39.20	-
C&J (Post and Bergsma 2013)	-	-	**49.20**	-
RecursiveNN (Socher et al. 2011b)	-	-	-	43.20
RNTN (Socher et al. 2013)	-	-	-	45.70
Paragraph-Vector (Le and Mikolov 2014)	-	-	-	**48.70**
CNN	94.79	94.04	47.47	46.35
RCNN	**96.49**	**95.20**	**49.19**	47.21

Table 2: Test set results for the datasets. The top, middle, and bottom parts are the baselines, the state-of-the-art results and the results of our model, respectively. The state-of-the-art results are reported by the corresponding essays.

beddings using the default parameter in word2vec[7] with the Skip-gram algorithm. We use Wikipedia dumps in both English and Chinese to train the word embedding.

Comparison of Methods

We compare our method with widely used text classification methods and the state-of-the-art approaches for each dataset.

Bag of Words/Bigrams + LR/SVM Wang and Manning (2012) proposed several strong baselines for text classification. These baselines mainly use machine learning algorithms with unigram and bigrams as features. We use logistic regression (LR) and SVM[8], respectively. The weight of each feature is the term frequency.

Average Embedding + LR This baseline uses the weighted average of the word embeddings and subsequently applies a softmax layer. The weight for each word is its tf-idf value. Huang et al. (2012) also used this strategy as the global context in their task. Klementiev, Titov, and Bhattarai (2012) used this in crosslingual document classification.

LDA LDA-based approaches achieve good performance in terms of capturing the semantics of texts in several classification tasks. We select two methods as the methods for comparison: ClassifyLDA-EM (Hingmire et al. 2013) and Labeled-LDA (Li, Sun, and Zhang 2008).

Tree Kernels Post and Bergsma (2013) used various tree kernels as features. It is the state-of-the-art work in the ACL native language classification task. We list two major methods for comparison: the context-free grammar (CFG) produced by the Berkeley parser (Petrov et al. 2006) and the reranking feature set of Charniak and Johnson (2005) (C&J).

[7]code.google.com/p/word2vec

[8]www.csie.ntu.edu.tw/~cjlin/liblinear

RecursiveNN We select two recursive-based methods for comparison with the proposed approach: the Recursive Neural Network (RecursiveNN) (Socher et al. 2011a) and its improved version, the Recursive Neural Tensor Networks (RNTNs) (Socher et al. 2013).

CNN We also select a convolutional neural network (Collobert et al. 2011) for comparison. Its convolution kernel simply concatenates the word embeddings in a pre-defined window. Formally,

$$x_i = [\boldsymbol{e}(w_{i-\lfloor win/2 \rfloor}); \ldots; \boldsymbol{e}(w_i); \ldots; \boldsymbol{e}(w_{i+\lfloor win/2 \rfloor})].$$

Results and Discussion

The experimental results are shown in Table 2.

- When we compare neural network approaches (RecursiveNN, CNN, and RCNN) to the widely used traditional methods (e.g., BoW+LR), the experimental results show that the neural network approaches outperform the traditional methods for all four datasets. It proves that neural network based approach can effective compose the semantic representation of texts. Neural networks can capture more contextual information of features compared with traditional methods based on BoW model, and may suffer from the data sparsity problem less.
- When comparing CNNs and RCNNs to RecursiveNNs using the SST dataset, we can see that the convolution-based approaches achieve better results. This illustrates that the convolution-based framework is more suitable for constructing the semantic representation of texts compared with previous neural networks. We believe the main reason is that CNN can select more discriminative features through the max-pooling layer and capture contextual information through convolutional layer. By contrast, RecursiveNN can only capture contextual information using semantic composition under the constructed textual tree, which heavily depends on the performance of tree construction. Moreover, compared to the recursive-based

approaches, which require $O(n^2)$ time to construct the representations of sentences, our model exhibits a lower time complexity of $O(n)$. In practice, the training time of the RNTN as reported in Socher et al. (2013) is approximately 3-5 hours. Training the RCNN on the SST dataset only takes several minutes using a single-thread machine.

- In all datasets expect ACL and SST dataset, RCNN outperforms the state-of-the-art methods. In the ACL dataset, RCNN has a competitive result with the best baseline. We reduce the error rate by 33% for the 20News dataset and by 19% for the Fudan set with the best baselines. The results prove the effectiveness of the proposed method.
- We compare our RCNN to well-designed feature sets in the ACL dataset. The experimental results show that the RCNN outperforms the CFG feature set and obtains a result that is competitive with the C&J feature set. We believe that the RCNN can capture long-distance patterns, which are also introduced by tree kernels. Despite the competitive results, the RCNN does not require handcrafted feature sets, which means that it might be useful in low-resource languages.
- We also compare the RCNN to the CNN and find that the RCNN outperforms the CNN in all cases. We believe that the reason is the recurrent structure in the RCNN captures contextual information better than window-based structure in CNNs. This results demonstrate the effectiveness of the proposed method. To illustrate this point more clearly, we propose a detailed analysis in the next subsection.

Contextual Information In this subsection, we investigate the ability of the recurrent structure in our model for capturing contextual information in further detail. The difference between CNNs and RCNNs is that they use different structure for capturing contextual information. CNNs use a fixed window of words as contextual information, whereas RCNNs use the recurrent structure to capture a wide range of contextual information. The performance of a CNN is influenced by the window size. A small window may result in a loss of some long-distance patterns, whereas large windows will lead to data sparsity. Furthermore, a large number of parameters are more difficult to train.

We consider all odd window sizes from 1 to 19 to train and test the CNN model. For example, when the window size is one, the CNN only uses the word embedding $[\boldsymbol{e}(w_i)]$ to represent the word. When the window size is three, the CNN uses $[\boldsymbol{e}(w_{i-1}); \boldsymbol{e}(w_i); \boldsymbol{e}(w_{i+1})]$ to represent word w_i. The test scores for these various window sizes are shown in Figure 2. Because of space limitations, we only show the classification results for the 20Newsgroups dataset. In this figure, we can observe that the RCNN outperforms the CNN for all window sizes. It illustrate that the RCNN could capture contextual information with a recurrent structure that does not rely on the window size. The RCNN outperforms window-based CNNs because the recurrent structure can preserve longer contextual information and introduces less noise.

Learned Keywords To investigate how our model constructs the representations of texts, we list the most important words in the test set in Table 3. The most important words are the information most frequently selected in the max-pooling layer. Because the word representation in our model is a word together with its context, the context may contain the entire text. We only present the center word and its neighboring trigram. For comparison, we also list the most positive/negative trigram phrases extracted by the RNTN (Socher et al. 2013).

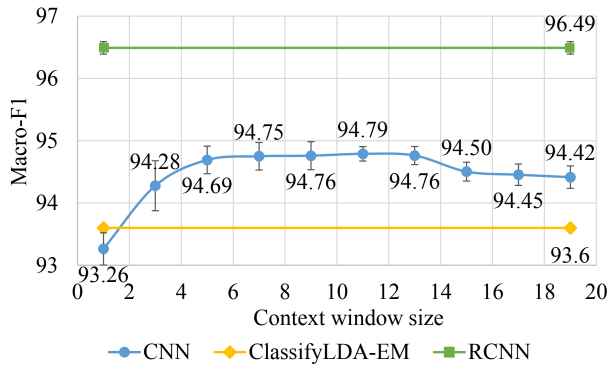

Figure 2: Macro-F1 curve for how context window size influences the performance of the 20Newsgroups classification

	RCNN
P	well *worth* the; a *wonderful* movie; even *stinging* at; and *invigorating* film; and *ingenious* entertainment; and *enjoy* .; 's *sweetest* movie
N	A *dreadful* live-action; Extremely *boring* .; is *n't* a; 's *painful* .; Extremely *dumb* .; an *awfully* derivative; 's *weaker* than; incredibly *dull* .; very *bad* sign;
	RNTN
P	an amazing performance; most visually stunning; wonderful all-ages triumph; a wonderful movie
N	for worst movie; A lousy movie; a complete failure; most painfully marginal; very bad sign

Table 3: Comparison of positive and negative features extracted by the RCNN and the RNTN

In contrast to the most positive and most negative phrases in RNTN, our model does not rely on a syntactic parser, therefore, the presented n-grams are not typically "phrases". The results demonstrate that the most important words for positive sentiment are words such as "worth", "sweetest", and "wonderful", and those for negative sentiment are words such as "awfully", "bad", and "boring".

Conclusion

We introduced recurrent convolutional neural networks to text classification. Our model captures contextual information with the recurrent structure and constructs the representation of text using a convolutional neural network. The experiment demonstrates that our model outperforms CNN and RecursiveNN using four different text classification datasets.

Acknowledgments

The authors would like to thank the anonymous reviewers for the constructive comments. This work was sponsored by the National Basic Research Program of China (No. 2014CB340503) and the National Natural Science Foundation of China (No. 61202329, 61272332).

References

Aggarwal, C. C., and Zhai, C. 2012. A survey of text classification algorithms. In *Mining text data*. Springer. 163–222.

Baroni, M.; Dinu, G.; and Kruszewski, G. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL*, 238–247.

Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003. A Neural Probabilistic Language Model. *JMLR* 3:1137–1155.

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE TPAMI* 35(8):1798–1828.

Bottou, L. 1991. Stochastic gradient learning in neural networks. In *Proceedings of Neuro-Nımes*, volume 91.

Cai, L., and Hofmann, T. 2003. Text categorization by boosting automatically extracted concepts. In *SIGIR*, 182–189.

Charniak, E., and Johnson, M. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL*, 173–180.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *JMLR* 12:2493–2537.

Cover, T. M., and Thomas, J. A. 2012. *Elements of information theory*. John Wiley & Sons.

Elman, J. L. 1990. Finding structure in time. *Cognitive science* 14(2):179–211.

Erhan, D.; Bengio, Y.; Courville, A.; Manzagol, P.-A.; Vincent, P.; and Bengio, S. 2010. Why does unsupervised pre-training help deep learning? *JMLR* 11:625–660.

Hingmire, S.; Chougule, S.; Palshikar, G. K.; and Chakraborti, S. 2013. Document classification by topic labeling. In *SIGIR*, 877–880.

Hinton, G. E., and Salakhutdinov, R. R. 2006. Reducing the dimensionality of data with neural networks. *Science* 313(5786):504–507.

Huang, E. H.; Socher, R.; Manning, C. D.; and Ng, A. Y. 2012. Improving word representations via global context and multiple word prototypes. In *ACL*, 873–882.

Kalchbrenner, N., and Blunsom, P. 2013. Recurrent convolutional neural networks for discourse compositionality. In *Workshop on CVSC*, 119–126.

Klementiev, A.; Titov, I.; and Bhattarai, B. 2012. Inducing crosslingual distributed representations of words. In *Coling*, 1459–1474.

Le, Q. V., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *ICML*.

Lewis, D. D. 1992. An evaluation of phrasal and clustered representations on a text categorization task. In *SIGIR*, 37–50.

Li, W.; Sun, L.; and Zhang, D. 2008. Text classification based on labeled-lda model. *Chinese Journal of Computers* 31(4):620–627.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, 3111–3119.

Mikolov, T.; Yih, W.-t.; and Zweig, G. 2013. Linguistic regularities in continuous space word representations. In *NAACL-HLT*, 746–751.

Mikolov, T. 2012. *Statistical language models based on neural networks*. Ph.D. Dissertation, Brno University of Technology.

Mnih, A., and Hinton, G. 2007. Three new graphical models for statistical language modelling. In *ICML*, 641–648.

Morin, F., and Bengio, Y. 2005. Hierarchical probabilistic neural network language model. In *AISTATS*, 246–252.

Ng, A. Y. 2004. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *ICML*, 78.

Petrov, S.; Barrett, L.; Thibaux, R.; and Klein, D. 2006. Learning accurate, compact, and interpretable tree annotation. In *Coling-ACL*, 433–440.

Plaut, D. C., and Hinton, G. E. 1987. Learning sets of filters using back-propagation. *Computer Speech & Language* 2(1):35–61.

Post, M., and Bergsma, S. 2013. Explicit and implicit syntactic features for text classification. In *ACL*, 866–872.

Socher, R.; Huang, E. H.; Pennington, J.; Ng, A. Y.; and Manning, C. D. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*, volume 24, 801–809.

Socher, R.; Pennington, J.; Huang, E. H.; Ng, A. Y.; and Manning, C. D. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*, 151–161.

Socher, R.; Perelygin, A.; Wu, J. Y.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 1631–1642.

Turian, J.; Ratinov, L.; and Bengio, Y. 2010. Word representations: a simple and general method for semi-supervised learning. In *ACL*, 384–394.

Wang, S., and Manning, C. D. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*, 90–94.