Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts

Xingyou Wang[1], **Weijie Jiang**[2], **Zhiyong Luo**[3],
[1]Beijing Language and Culture University, Beijing, China {ultimate010@gmail.com}
[2]Tsinghua University, Beijing, China {jwj14@mails.tsinghua.edu.cn}
[3]Beijing Language and Culture University, Beijing, China {luo_zy@blcu.edu.cn}

Abstract

Sentiment analysis of short texts is challenging because of the limited contextual information they usually contain. In recent years, deep learning models such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have been applied to text sentiment analysis with comparatively remarkable results. In this paper, we describe a jointed CNN and RNN architecture, taking advantage of the coarse-grained local features generated by CNN and long-distance dependencies learned via RNN for sentiment analysis of short texts. Experimental results show an obvious improvement upon the state-of-the-art on three benchmark corpora, MR, SST1 and SST2, with 82.28%, 51.50% and 89.95% accuracy, respectively. [1]

1 Introduction

The rapid development of the Internet, e-commerce and social networks brings about a large amount of user-generated short texts on the Internet, such as online reviews for products, services and blogs. Such short texts as online reviews are usually subjective and semantic oriented. To discriminate and classify the semantic orientation of such short texts properly is of great research and practical value.

Sentiment analysis of short texts is challenging because of the limited contextual information and the sparse semantic information they normally contain. The existing research on sentiment analysis of short texts basically include emotional knowledge-based methods and feature-based classification methods. The former mainly focuses on the extraction and the sentiment classification based on opinion-bearing words and opinion sentences (Hu and Liu, 2004; Kim and Hovy, 2005). The latter focuses on the sentiment classification based on features. Turney (2002) presented the unsupervised PMI-IR (Pointwise Mutual Information and Information Retrieval) algorithm to measure the similarity of words or phrases. Pang et al. (2002) and Cui et al. (2006) used n-grams and POS tags and applied them to NB, ME and SVM classifiers. Based on these studies, Kim and Hovy (2006) introduced the positional features and opinion-bearing word features. Mullen and Collier (2004) combined various features and used SVM for classification.

With the development of deep learning, typical deep learning models such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have achieved remarkable results in computer vision and speech recognition. Word embeddings, CNNs and RNNs have been applied to text sentiment analysis and gotten remarkable results. Kim (2014) applied CNN on top of pre-trained word vectors for sentence-level classification. Some studies utilized recursive neural networks to construct the sentence-level representation vector in sentiment analysis (Socher et al., 2011; Socher et al., 2012b; Socher et al., 2013b). Le and Mikolov (2014) presented the paragraph vector in sentiment analysis. Tai et al. (2015) put forward the tree-structured long short-term memory (LSTM) networks to improve the semantic representations. The improved performance of these algorithms mainly benefits from the following aspects: (1) The high-dimensional distributional vectors endow similar semantic-oriented words with high similarity. Word embeddings can better solve the semantic sparsity of short texts compared with the one-hot representation. (2) Similar to the translation, rotation and scale invariance of images in CNN, CNN is able to learn the local features from words or phrases in different places of texts. (3) RNN takes words in

[1] Code and data are available at `https://github.com/ultimate010/crnn`

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2428–2437, Osaka, Japan, December 11-17 2016.

a sentence in a sequential order and is able to learn the long-term dependencies of texts rather than local features.

In this paper, we present a jointed CNN and RNN architecture that takes the local features extracted by CNN as input to RNN for sentiment analysis of short texts. We develop an end-to-end and bottom-up algorithm to effectively model sentence representation. We take the word embeddings as the input of our CNN model in which windows of different length and various weight matrices are applied to generate a number of feature maps. After convolution and pooling operations, the encoded feature maps are taken as the input to the RNN model. The long-term dependencies learned by RNN can be viewed as the sentence-level representation. The sentence-level representation is taken to the fully connected network and the softmax output reveals the classification result. The deep learning algorithm we put forward differs from the existing methods in that: (1) We apply windows of different length and various weight matrices in convolutional operation. The max pooling operates on the adjacent features and moves from left to right instead of on the entire sentence. In this case, the feature maps generated in our CNN model retain the sequential information in the sentence context. (2) The deep learning architecture of our model takes advantage of the encoded local features extracted from the CNN model and the long-term dependencies captured by the RNN model. We experiment on three benchmarks for sentiment classification, MR, SST1 and SST2 and achieve the state-of-the-art results.

This work is organized as follows. In section 2, we discuss some background knowledge about word embeddings, sentence-level representation, convolutional neural network and recurrent neural network. In section 3, we describe our jointed CNN model and RNN model in detail. Section 4 presents our experiment results and some discussion. Finally, in section 5, we conclude and remark on our work.

2 Background

In this section, we discuss some background knowledge on word embeddings, sentence-level representation, convolutional neural network (CNN) and recurrent neural network (RNN).

2.1 Word Embeddings and Sentence-Level Representation

When applying deep learning methods to a text classification task, we normally need to transform words into high-dimensional distributional vectors that capture morphological, syntactic and semantic information about the words. Let d be the length of word embeddings and l be the length of a sentence. The sentence-level representation is encoded by an embedding matrix $C \in \mathbb{R}^{d \times l}$, where $C_i \in \mathbb{R}^d$ corresponds to the word embeddings of the i-th word in the sentence.

2.2 Convolution and Pooling

Convolution is widely used in sentence modeling (Kim, 2014; Kalchbrenner et al., 2014; dos Santos and Gatti, 2014). The structure of convolution varies slightly in different research fields, among which the structure used in natural language processing is shown in Figure 1.

Generally, let l and d be the length of sentence and word vector, respectively. Let $C \in \mathbb{R}^{d \times l}$ be the sentence matrix. A convolution operation involves a convolutional kernel $H \in \mathbb{R}^{d \times w}$ which is applied to a window of w words to produce a new feature. For instance, a feature c_i is generated from a window of words $C[*, i : i + w]$ by

$$c_i = \sigma(\sum(C[*, i : i + w] \circ H) + b) \quad (1)$$

Here $b \in \mathbb{R}$ is a bias term and σ is a non-linear function, normally tanh or ReLu. $\circ$ is the Hadamard product between two matrices. The convolutional kernel is applied to each possible window of words in the sentence to produce a feature map. $c = [c_1, c_2, \ldots, c_{l-w+1}]$, with $c \in \mathbb{R}^{l-w+1}$.

Next, we apply pairwise max pooling operation over the feature map to capture the most important feature. The pooling operation can be considered as feature selection in natural language processing. The max pooling operation is shown in Figure 2.

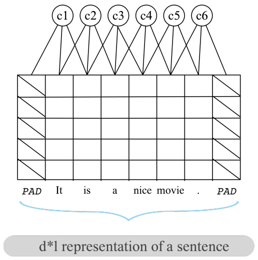

Figure 1: (1) A sentence matrix with padding. (2) Convolution with a window of three words.

Specifically, the output of convolution, the feature map $c = [c_1, c_2, \ldots, c_{l-w+1}]$ is the input of the pooling operation. Let the input be downscaled by 2, namely, the adjacent two features in the feature map be caculated as follows:

$$p_i = \max(c_{2\times i-1}, c_{2\times i}) \tag{2}$$

The output of the max pooling operation is $p = [p_1, p_2, \ldots, p_{\lfloor \frac{l-w+1}{2} \rfloor}]$, $p \in \mathbb{R}^{\lfloor \frac{l-w+1}{2} \rfloor}$.

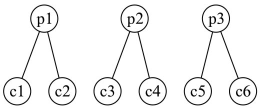

Figure 2: Pairwise max pooling operation on a scaling of 2

2.3 Recurrent neural network

Recurrent Neural Networks (RNNs) have shown great promise in machine translation tasks(Liu et al., 2014; Sutskever et al., 2014; Auli et al., 2013). Unlike feedforward neural networks, RNNs are able to handle a variable-length sequence input by having a recurrent hidden state whose activation at each time is dependent on that of the previous time. Figure 3 shows what a typical RNN looks like.

The diagram shows a RNN being unrolled into a full network. For example, if the input sequence is a four-word sentence, the network would be unrolled into a 4-layer neural network, one layer for each word. The formulas that govern the calculations in a RNN are as follows.

- x_t is the input at time step t. For example, x_i could be a one-hot vector or word embeddings corresponding to the i-th word of a sentence.
- h_t is the hidden state and the "memory" of the network at time step t. h_t is calculated according to the previous hidden state and the input at the current step:

$$h_t = \sigma(Ux_t + Wh_{t-1}) \tag{3}$$

Here h_0 is typically initialized to a zero vector in order to calculate the first hidden state.

- o_t is the output at time step t. For sentiment classification of short texts, it would be a vector of probabilities across all the sentiment categories. o_t is calculated as follows:

$$o_t = softmax(Vh_t) \tag{4}$$

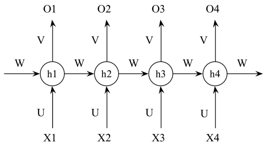

Figure 3: An unrolled recurrent neural network

Similar to a traditional neural network, we can use a twisted backpropagation algorithm Backpropagation Through Time (BPTT) to train a RNN (Mozer, 1989). Unfortunately, it is difficult to train RNN to capture long-term dependencies because the gradients tend to either vanish or explode (Bengio et al., 1994). Hochreiter and Schmidhuber (1997) proposed a long short-term memory (LSTM) unit and Cho et al. (2014) proposed a gated recurrent unit (GRU) to deal with the problem effectively.

2.4 LSTM and GRU

2.4.1 LSTM

The Long Short-Term Memory (LSTM) was first proposed by Hochreiter and Schmidhuber (1997) that can learn long-term dependencies. See Figure 4 for the graphical illustration.

Different from traditional recurrent unit, LSTM unit keeps the existing memory $c_t \in \mathbb{R}^n$ at time t. The input at time t is x_t, h_{t-1}, c_{t-1}, the output is h_t, c_t, they can be updated by the following equaions:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{5}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{6}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{7}$$

$$g_t = tanh(W_g x_t + U_g h_{t-1} + b_g) \tag{8}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{9}$$

$$h_t = o_t \odot tanh(c_t) \tag{10}$$

where $\sigma(\cdot)$ denotes the logistic sigmoid fuction. The opration $\odot$ denotes the element-wise vector product. At each time step t, there are an input gate i_t, a forget gate f_t, an output gate o_t, a memory cell c_t and a hidden unit h_t. h_0 and c_0 can be initialized to 0 and the parameters of the LSTM is W, U, b.

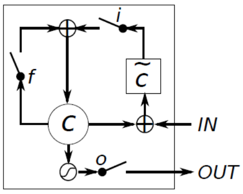

Figure 4: LSTM unit (Chung et al., 2014)

2.4.2 GRU

A gated recurrent unit (GRU) was initially proposed by Cho et al. (2014) to make each recurrent unit to adaptively capture dependencies of different time scales. See Figure 5 for the graphical illustration of GRU.

The parameters can be updated by the following equations

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{11}$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{12}$$

$$\hat{h}_t = \tanh(W x_t + U(r_t \odot h_{t-1})) \tag{13}$$

$$h_t = (1 - z_t) h_{t-1} + z_t \hat{h}_t \tag{14}$$

Where σ denotes the logistic sigmoid function, $\odot$ denotes the element-wise multiplication, r_t denotes the reset gate, z_t denotes the update gate and $\hat{h}_t$ denotes the candidate hidden layer.

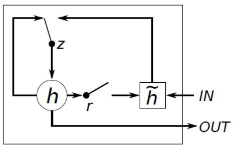

Figure 5: GRU unit (Chung et al., 2014)

3 Model

Convolutional neural networks (CNNs) are likely to extract local and deep features from natural language. It has been shown that CNN has gotten improved results in sentence classification (Kim, 2014). Recurrent neural networks (RNNs) are various kinds of time-recursive neural network that is able to learn the long-term dependencies in sequential data. Seeing that we can view the words in a sentence as a sequence from left to right, RNNs can be modeled in accordance with people's reading and understanding behavior of a sentence. Socher et al. (2012a) presented a convolutional-recursive deep model for 3D object classification that combined the convolutional and recursive neural networks together. The CNN layer learns the low-level translation invariant features which are inputs to multiple, fixed-tree RNNs (recursive neural networks) in order to compose higher order features. Kim et al. (2015) described a model that employed a convolutional neural network (CNN) and a highway network over characters, whose output is given to a long short-term memory (LSTM) recurrent neural network language model (RNN-LM). These two models both get better results than prior methods. However, the recursive neural networks need to build a tree structure that is usually based on the parser result of sentence. The recurrent neural network is particularly suited for modeling the sequential pattern. Inspired by those works and the fact that CNN can extract local features of input and RNN (recurrent neural network) can process sequence input and learn the long-term dependencies, we combine both of them in sentiment analysis of short texts. The model architecture is shown in Figure 6.

Our model consists of the following parts: word embeddings and sentence-level representation, convolutional and pooling layers, concatenation layer, RNN layer, fully connected layer with softmax output.

3.1 Word Embeddings and Sentence-Level Representation

Word embeddings play an important role in word representation. The commonly used are random initialization and unsupervised pre-training of word embeddings. In our experiment, we perform unsupervised learning of word-level embeddings using the *word2vec* method and also test random initialization. Let ν be the size of bag-of-words and d be the length of a word embedding, then the word embeddings of all the words in the vocabulary are encoded by column vectors in an embedding matrix $Q \in \mathbb{R}^{d \times \nu}$. A sentence can be represented by:

$$S = [w_1, w_2, \ldots, w_l] \tag{15}$$

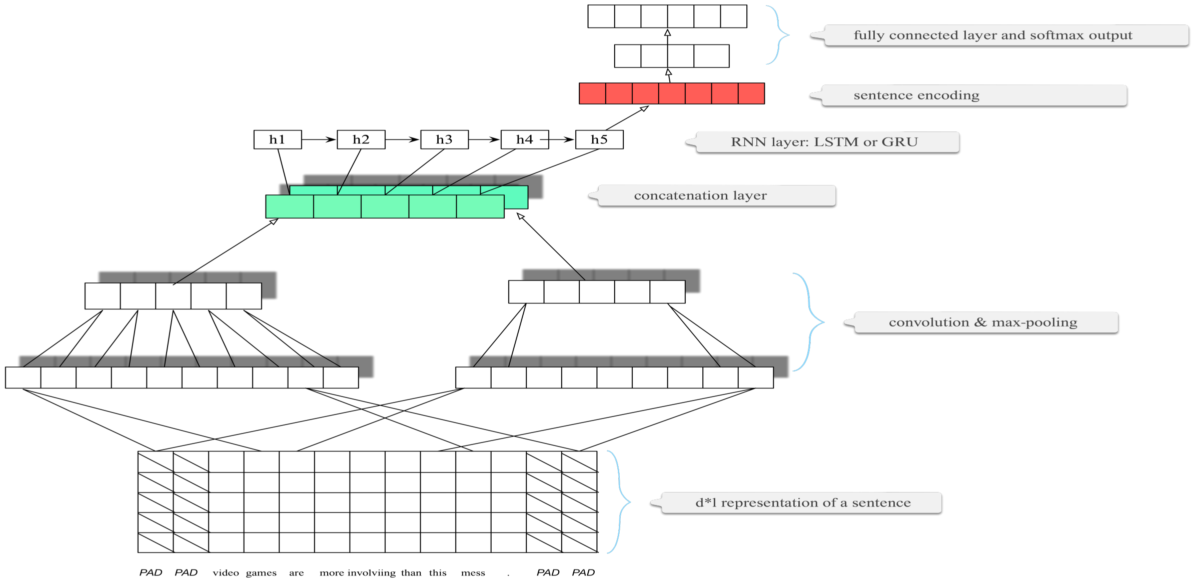
fully connected layer and softmax output
sentence encoding
RNN layer: LSTM or GRU
h1
h2
h3
h4
h5
concatenation layer
convolution & max-pooling
d*l representation of a sentence
PAD PAD video games are more involviing than this mess . PAD PAD

Figure 6: Model architecture for an example sentence

$$w_i \in [1, \nu], i \in [1, l]$$

Here is the sentence-level representation:

$$C_i = Q[w_i], C_i \in \mathbb{R}^d \tag{16}$$

$$C = [C_1, C_2, \ldots, C_l], C \in \mathbb{R}^{d \times l} \tag{17}$$

The column vector C_i corresponds to the word embedding of the i-th word in the sentence.

3.2 Convolution and Pooling

The convolutional layer applies a matrix-vector operation to each window of size w of successive windows in the sentence-level representation sequence. Let $H \in \mathbb{R}^{d \times w}$ be the weight matrix of the convolutional layer, we then add a bias item b to the result of the matrix-vector operation and get a feature mapping $c \in \mathbb{R}^{l-w+1}$. The i-th element of the feature map is:

$$c_i = \sigma(\sum(C[*, i : i + w] \circ H) + b) \tag{18}$$

where $C[*, i : i + w]$ is the i-th to the $i + w$-th column vectors of the sentence-level representation.

The same weight matrix is used to extract local features for each window of the given sentence. Using the matrix over all word windows of the sentence, we extract the n-grams feature vector of size $l - w + 1$. We also apply various kinds of weight matrices and multiple filter lengths to get various and sufficient features.

We apply the max and average pooling operations and find that the former performs better with less computational complexity. Thus, we apply the max pooling operation to the output of convolutional layer which transform the feature map of size $l - w + 1$ to $\lfloor \frac{l-w+1}{2} \rfloor$,

$$p = [p_1, p_2, \ldots, p_{\lfloor \frac{l-w+1}{2} \rfloor}] \tag{19}$$

We apply m kinds of matrix weight to get m feature maps.

$$P = [p^1, p^2, \ldots, p^m] \tag{20}$$

where $P \in \mathbb{R}^{\lfloor \frac{l-w+1}{2} \rfloor \times m}$.

We borrow the experience from the literature (Zhang and Wallace, 2015; Kim, 2014) and choose window size 4 and 5 to get matrix P_4, P_5. After trunking longer result, we concatenate P_4 and P_5 together to get Z, where

$$Z = \oplus(P_4, P_5), Z \in \mathbb{R}^{\lfloor \frac{l-w+1}{2} \rfloor \times (m \times 2)} \tag{21}$$

where $\oplus$ is the concatenation operator, Z can be viewed as the convolutional coding of a sentence.

3.3 Recurrent Neural Network

The features generated from convolution and pooling operation can be viewed as advanced features like n-grams. Since recurrent neural network (RNN) can process sequential input and learn the long-term dependencies, we take these features as the input of the recurrent neural network. We apply LSTM and GRU that are mentioned in previous chapter and both get good results. The output of RNN $T \in \mathbb{R}^n$ is deemed as the encoding of the whole sentence.

3.4 Fully Connected Network with Softmax Output

The features generated from RNN form the penultimate layer and are passed to a fully connected softmax layer whose output is the probability distribution over all the categories. The softmax operation over the scores of all the categories is calculated as follows:

$$\hat{P}_i = \frac{exp(o_i)}{\sum_{j=1}^{C} exp(o_j)} \tag{22}$$

We take cross entropy as the loss function that measures the discrepancy between the real sentiment distribution $\hat{P}^t(C)$ and the model output distribution $\hat{P}(C)$ of sentences in the corpora.

$$loss = -\sum_{s \in T} \sum_{i=1}^{V} \hat{P}_i^t(C) log(\hat{P}_i(C)) \tag{23}$$

Here T is the training corpora, V is the number of the sentiment categories. $\hat{P}^t(C)$ is the V-dimension one-hot coding vector where the elements corresponding to the sentence's real sentiment category is 1 and other elements 0. The entire model is trained end-to-end with stochastic gradient descent.

4 Experimental Setup and Results

We conduct experiments to empirically evaluate our method by applying it to three benchmarks as follows.

- MR: Movie reviews with one sentence per review. Classification involves detecting positive/negative reviews (Pang and Lee, 2005).[2]

- SST1: Stanford Sentiment Treebank - an extension of MR but provided five kinds of labels, very negative, negative, neutral, positive and very positive (Socher et al., 2013a). [3]

- SST2: Same as SST1 but with neutral reviews removed and binary labels.

The experiment runs on Tesla K40c GPU. Summary statistics of the datasets are in Table 1.

[2] https://www.cs.cornell.edu/people/pabo/movie-review-data/

[3] http://nlp.stanford.edu/sentiment/

Data	c	l	$\lvert V_{train}\rvert$	$\lvert V_{val}\rvert$	$\lvert V_{test}\rvert$
MR	2	20	8655	961	1046
SST1	5	18	151525	0	2200
SST2	2	19	76836	0	1811

Table 1: Summary statistics for the datasets after tokenization. c: Number of target classes. l: Average sentence length. $|V_{train}|$: Training set size. $|V_{val}|$: Validation set size. $|V_{test}|$: Test set size. The training set of SST1 and SST2 includes phrases extracted from sentences and sentences themselves, and test set only includes sentences.

4.1 Model Variations

We experiment with several variants of the model.

- CNN-GRU-word2vec: A model with pre-trained vectors from word2vec, max pooling and GRU recurrent unit.
- CNN-LSTM-word2vec: A model with pre-trained vectors from word2vec, max pooling and LSTM recurrent unit.
- AGV-GRU-word2vec: A model with pre-trained vectors from word2vec, average pooling and GRU recurrent unit.
- CNN-GRU-rand: A model with randomly initialized vectors, max pooling and GRU recurrent unit.
- CNN-LSTM-rand: A model with randomly initialized vectors, max pooling and LSTM recurrent unit.

4.2 Results and Dicussion

Results of our models against other methods are listed in tabel 2. Specially, our models with pre-trained vectors from word2vec[4] and max pooling perform best among all the models, of which the one with the GRU recurrent unit performs better on MR and SST2 while the one with LSTM performs better on SST1. The classification accuracy is raised by 0.7% on MR and 1.8% on SST2, when implementing CNN-GRU-word2vec model compared with the existing models. At the same time, the CNN-LSTM-word2vec model raises the classification accuracy by 0.1%. Furthermore, LSTM reveals good performance on SST1 while GRU performs better on MR and SST2.

In the meantime, we find that our models with pre-trained vectors all perform better than the others with randomly initialized vectors on all three corpora. Thus, we infer that the pre-trained vectors on large-scale corpora can solve the semantic sparsity problem to some degree.

Compared with the existing methods and experiment results, we find that our jointed architecture of CNN and RNN model performs better than the CNN and RNN models alone in sentiment classification of short texts. We take advantage of both the CNN model and the RNN model thus get higher classification accuracy than the existing models. CNN extracts the local features of input and RNN processes sequence input while learning the long-term dependencies and get sentence-level feature representation. The experiments substantiate the validity of our idea.

5 Conclusion

In this work we present a deep neural network architecture that takes advantage of the construction of convolutional neural network (CNN) and recurrent neural network (RNN) and joint them together for sentimental analysis of short texts. In particular, our pooling operation on adjacent words is able to retain the local features and their sequential relations in a sentence. Besides, RNN can learn the long-term dependencies and the positional relation of features as well as the global features of the whole sentence.

[4] Pre-trained vectors `https://code.google.com/archive/p/word2vec/`

Group	Model	MR	SST1	SST2
Other	NB(Socher et al., 2013b)	–	41.0	81.8
	SVM(Socher et al., 2013b)	–	40.7	79.4
CNN	1-layer convolution(Kalchbrenner et al., 2014)	–	37.4	77.1
	Deep CNN(Kalchbrenner et al., 2014)	–	48.5	86.8
	Non-static(Kim, 2014)	81.5	48.0	87.2
	Multichannel(Kim, 2014)	81.1	47.4	88.1
Recursive	Basic(Socher et al., 2013b)	–	43.2	82.4
	Matrix-vector (Socher et al., 2013b)	–	44.4	82.9
	Tensor (Socher et al., 2013b)	–	45.7	85.4
	Tree LSTM1 (Zhu et al., 2015)	–	48.0	-
	Tree LSTM2 (Tai et al., 2015)	–	51.0	88.0
	Tree LSTM3 (Le and Zuidema, 2015)	–	49.9	88.0
	Tree bi-LSTM (Li et al., 2015)	0.79	–	–
Recurrent	LSTM(Tai et al., 2015)	–	46.4	84.9
	bi-LSTM(Tai et al., 2015)	–	49.1	87.5
Vector	Word vector avg(Socher et al., 2013b)	–	32.7	80.1
	Paragraph vector(Le and Mikolov, 2014)	–	48.7	87.8
TBCNNs	c-TBCNN(Mou et al., 2015)	–	50.4	86.8
	d-TBCNN(Mou et al., 2015)	–	51.4	87.9
CNN-RNN	CNN-GRU-word2vec	**82.28**	50.68	**89.95**
	CNN-LSTM-word2vec	81.52	**51.50**	89.56
	AVG-GRU-word2vec	81.44	50.36	89.61
	CNN-GRU-rand	76.34	48.27	86.64
	CNN-LSTM-rand	77.04	49.50	86.80

Table 2: Results of our jointed architecture of CNN and RNN against other methods.

Our models perform well on three benchmark datasets and achieve higher classification accuracy than the existing models.

Our jointed neural network architecture can be applied to sentence modeling as well as other natural language processing tasks. For future work, we will extend our models to long texts classification tasks.

Acknowledgments

We would like to thank Zhiyong Luo for his meticulous guidance and Jinsong Zhang for providing servers for us to run our code. We would also like to thank Ju Lin for his insightful comments.

References

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *EMNLP*, volume 3, page 0.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Hang Cui, Vibhu Mittal, and Mayur Datar. 2006. Comparative experiments on sentiment classification for online product reviews. In *AAAI*, volume 6, pages 1265–1270.

Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Soo-Min Kim and Eduard Hovy. 2005. Automatic detection of opinion bearing words and sentences. In *Companion Volume to the Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, pages 61–66.

Soo-Min Kim and Eduard Hovy. 2006. Automatic identification of pro and con reasons in online reviews. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 483–490. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. *arXiv preprint arXiv:1503.02510*.

Jiwei Li, Dan Jurafsky, and Eudard Hovy. 2015. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185*.

Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. 2014. A recursive recurrent neural network for statistical machine translation. In *ACL (1)*, pages 1491–1500.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. *Unpublished manuscript: http://arxiv. org/abs/1504. 01106v5. Version*, 5.

Michael C Mozer. 1989. A focused back-propagation algorithm for temporal pattern recognition. *Complex systems*, 3(4):349–381.

Tony Mullen and Nigel Collier. 2004. Sentiment analysis using support vector machines with diverse information sources. In *EMNLP*, volume 4, pages 412–418.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Richard Socher, Brody Huval, Bharath Bath, Christopher D Manning, and Andrew Y Ng. 2012a. Convolutional-recursive deep learning for 3d object classification. In *Advances in Neural Information Processing Systems*, pages 665–673.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012b. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng, and Christopher Potts. 2013a. Parsing With Compositional Vector Grammars. In *EMNLP*.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over tree structures. *arXiv preprint arXiv:1503.04881*.