Deep ConvRNN for Sentiment Parsing of Chinese Microblogging Texts

Jiajun Cheng, Sheng Zhang, Pei Li, Xin Zhang, Hui Wang
College of Information Systems and Management,
National University of Defense Technology,
Changsha, P. R. China
e-mail: {jiajun.cheng, zhangsheng, peili, zhangxin, huiwang} @nudt.edu.cn

***Abstract*—Recurrent Neural Networks (RNNs) are naturally applicable to sequential processing and have achieved outstanding performance in analyzing natural language. However, RNN-based sequence labeling methods may encounter some problems, such as word ambiguous and low-fidelity of word segmentation, in sentiment parsing of Chinese microblogging texts because it cannot well grasp local contextual information of words. Therefore, in this work, we propose a novel neural network architecture, named convRNN, for sentiment parsing of Chinese texts. The convRNN combines Convolutional Neural Network (CNN) and RNN to capture the local contextual feature and global sequence feature of words in a sentence. Experimental results demonstrate that extracting local contextual features of words with CNN improves the performance of RNN models. Furthermore, deep convRNNs achieve better performance than shallow models and outperform the RNN-based method substantially.**

Keywords- Microblog; sentiment parsing; sequence labeling; RNN; CNN

I. INTRODUCTION

Microblogging websites such as Twitter and Sina Weibo have attracted a lot of users to share their experiences and express their opinions on different topics, products, and services. Sentiment analysis of microblogging texts is of significance and has motivated many researchers to apply it to different practical applications, such as electronic opinion poll [1] and word-of-mouth tracking[2]. This paper focuses on a fine-grained sentiment analysis task—sentiment parsing of Chinese microblogging texts, which aims to jointly extract the targeted entities mentioned in a microblogging sentence as well as the sentiments expressed to these entities [3].

Generally, Sentiment parsing has been tackled as a sequence labeling problem and RNN performs well in this task [3]. RNN-based sequence labeling methods view a sentence as a word sequence and predict the label of each word according to the embedding of the word and the order of all the words in the sentence. In such process, local contextual features of words are not well extracted because the input of RNN is the word embedding with general meanings and RNN is not good at catching local contextual features. However, it is significant to consider local contextual information of words because of three reasons: (1) Many words express different meanings in different contexts. The meaning of a word often highly depends on the local contexts, i.e., the words near it. Hence, extracting local contextual feature makes the meanings of words more clear. (2) Some words are just parts of some phrases and do not have clear meanings as single words. Extracting local contextual feature helps to take the meaning of the phrase as the features. (3) The word segmentation result may not be good enough due to the terse, colloquial, and sometimes obscure writing style of Chinese microblogging texts. Wrong word segmentation may cause some problems in understanding the meaning of the sentence and thus increase the difficulties of sentiment parsing.

The issues raised above motivate the work presented in this paper. We improve the general RNN-based sequence labeling architecture by adding a local contextual feature extracting module. The new neural network architecture contains four kinds of layers, i.e., word embedding layer, local contextual feature extraction layers, global sequential feature extraction layers and labeling layer. Word embedding layer projects each word into a pre-trained vector. Local contextual feature extraction layers catch the local contextual information of each word with the vectors of the word and its neighbors. Global sequential feature extraction layers grasp the global information of the whole sentence and labeling layer predicts the label of each word according to its local and global contextual features. As long as CNN has shown its excellent capability of capturing local correlations of spatial or temporal structures in recent studies [4], we employ CNN in local contextual feature extraction layers to extract the local contextual feature of words.

In summary, this paper makes the following contributions:

- We propose a new neural network-based sequence labeling architecture for sentiment parsing of Chinese microblogging texts. The new architecture improves the general RNN-based sequence labeling architecture by considering the local contextual feature of words especially.
- We propose a new neural network model, called convRNN, to extract both the local and the global contextual feature of words by combining CNN and RNN for sequence labeling. The convRNN takes advantages of the capability of CNN in catching local contextual information and the ability of RNN in processing sequence problems.
- Experimental results demonstrate that convRNNs perform better than the corresponding RNN models and deep convRNN-based sentiment parsing method outperforms RNN-based methods substantially by nearly 4%.

978-1-5386-2030-4/17/$31.00 ©2017 IEEE

II. RELATED WORK

A. Sentiment Analysis

Most previous work on sentiment analysis focuses on sentiment classification of texts, which assign a sentiment score or a sentiment polarity to each input document using lexicon-based methods [5][6] or supervised learning-based methods [7][8]. However, sentiment classification does not consider the opinion target and thus insufficient for fine-grained sentiment analysis. Therefore, some studies use rule-based methods to extract target entities and corresponding sentiments jointly by analyzing the occurrence of expert-defined keywords and patterns in the text [10]. However, the keywords and patterns in rule-based approaches are often domain specific and expensive, if not difficult, to extend to new domains. Moreover, the terse, colloquial, and sometimes obscure writing style of microblogging texts brings new challenges to rule-based approaches.

B. Deep Learning in NLP

In recent years, deep learning has achieved success in a number of Natural Language Processing (NLP) tasks. Among the neural network models used in NLP, CNN and RNN are two most popular ones [11].

Owing to the capability of capturing local correlations of spatial or temporal structures, CNNs have been successfully applied to many NLP tasks. Some studies prove that CNN is an effective approach to grasp morphological information and apply it to generate word embedding from the characters[12][13]. In word-level processing, a lot of researchers employ CNN for text modeling and further exploit the text features in document-level and sentence-level NLP tasks, such as text classification [14], sentiment analysis [15] and machine translation [4]. However, because the dimension of features always decreases sharply through CNN layers, only a few work utilize CNNs for sequence labeling in NLP[16].

Benefiting from the recurrent network structure, RNNs have achieved outstanding performance in several NLP tasks. Mikolov [17] proposes an RNN-based language modeling approach and achieves better performance than feedforward neural network. Yazdani et. al. [18] apply RNN for semantic parsing. Li et. al. [19] employ RNN on biomedical named entity recognition and gain better results than CRF. Irsoy et al. [20] explore bidirectional and deep RNN models in opinion expression detection and achieve a better result than the state-of-the-art.

Some studies combine CNN and RNN for some NLP tasks. Most of them combine CNN and RNN for text modeling [11][23]. As to sequence labeling, Chiu and Nichols [12] employ character-level CNN to construct word vectors and utilize bidirectional LSTM for named entity extraction. Ma and Hovy [13] use CNN to generate word vectors as well and combine bidirectional LSTM and CRF for sequence labeling. Both studies take advantage of CNN to extract morphological information and exploit it to represent words in character-level. Different from the above methods, we use word-level CNN for sequence labeling task.

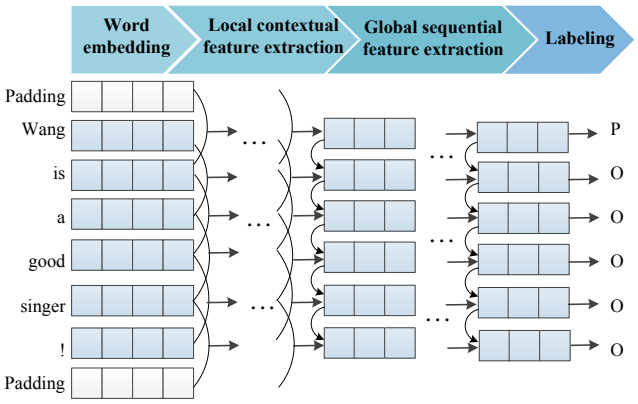

Figure 1. The neural network architecture of deep convRNN for sequence labeling of a sentence.

III. METHOD

In this section, we first give a brief introduction to our neural network architecture. After that, we describe the layers of the neural network one-by-one from the input layer to the output layer.

A. Architecture

Figure 1 displays the neural network architecture of deep convRNN for sequence labeling of a sentence. There are four types of layers. Word embedding layer represents the general meaning of each word with a fixed-length vector. Local contextual feature extraction layers aim to catch the contextual meaning of each word with the general meaning of the word and its local contexts, i.e., the words near it. Global sequential feature extraction layers analyze the semantic role of each word in a sentence with all the words in the sentence. Labeling layer predicts the label of each word with the feature extracted from the former layers.

B. Word Embedding

Word embedding layer represents each word with a vector and thus words can be calculated in high-level layers. Specifically, suppose D is the dictionary containing all tokens in the dataset, word embedding $E \in R^{d \times |D|}$ contains a special vector $e_t \in R^d$ for each token t in dictionary D. In the task-specific learning process, word embedding E is initialized with E_0, which is pre-trained on large unsupervised corpus, and taken as parameters in the training process. Because E_0 is trained on large unsupervised language corpus, it represents the general meanings of words.

C. Local Contextual Feature Extraction with CNN

Word embedding layer provides each word a unique and fixed-length vector to represent its general meaning. However, the meaning of a word in a sentence is often affected by its contexts, especially local contexts. Therefore, it is important to consider contextual information of a word when analyzing the meaning and corresponding label of it. Local contextual feature extraction layers aim to make the

meaning of each word more clear with its local contexts. As long as many studies have shown that CNN is an effective approach to extract local features [11][14], we apply CNN layers to extract local contextual features of words after the word embedding layer.

Generally, CNN contains two main steps—convolution and pooling. Convolution step extracts steady local features with same weights in different locations by sliding a fixed-size-widow. Pooling step selects important and invariance features over a region by computing the max or mean value of the features in this region. In this paper, we process a sequence labeling task to select a label for each token in a sentence. Therefore, we need to keep the number of hidden units in each hidden layer same with the length of the input sentence. Considering that pooling step get one value for a region and thus will reduce the dimension sharply, we use convolution step only to extract local features of each word in sentences. In addition, we add some paddings at the beginning and the end of sentences before each convolution operation to keep the number of hidden units in each hidden layer steady.

Specifically, for sentence $S = \{t_1, t_2, ..., t_s\}$, each token t_j has been represented to be a *d*-dimensional vector $e_{t_j} \in R^d$ in word embedding layer. In convolution operation, let m be the length of filters, n be the number of filters, $W_{ci} \in R^{m \times d}$ be the filter weight of the ith filter, and $c_j \in R^n$ be the feature vector of t_j after a convolutional operation. To avoid bias in convolutional operation m is set to be odd. Then the convolution operation is computed by

$$c_{ji} = f(W_{ci} \circ [e_{t_{j-m/2+1/2}}, ..., e_{t_j}, ..., e_{t_{j+m/2-1/2}}] + b_i), \quad (1)$$

where $\circ$ denotes element-wise multiplication, $b_i \in R$ is a bias term and f is nonlinear active function (e.g., *tanh* and *sigmoid* function). When $(j - m/2 + 1/2)$ is less than 1 or $(j + m/2 - 1/2)$ is larger than s, the corresponding vector is the embedding of padding.

D. Global Sequential Feature Extraction with RNN

In sentiment parsing task, the label of a word in a sentence is determined by the role of the word in the sentence as well as the meaning of it. Therefore, it is significant to extract global sequential feature before labeling each word in a sentence. In view of the good performance of RNNs in processing sequence problems [13] [20], we use RNN and its variant models as the sequence labeling model to extract global sequential features for sentiment parsing.

Specifically, given an input $X = \{x_1, x_2, ..., x_s\}$, simple RNN (SRNN) employs a recurrent hidden layer $H = \{h_1, h_2, ..., h_s\}$ to extract features of a token from the token and its former tokens. Each hidden unit h_j is calculated with the current input x_j and last hidden unit h_{j-1}:

$$h_j = f(Wx_j + Uh_{j-1} + b_h), \quad (2)$$

where W and U are weight matrices, b_h is a bias vector, f is nonlinear function, $j \in \{1, 2, ..., s\}$ and $h_0 = \mathbf{0}$. The output of a simple RNN is computed by a linear function:

$$y_j = Vh_j + b_y, \quad (3)$$

where V is a weight matrix and b_y is a bias vector.

Simple RNN often suffers the gradient vanishing problem, which means that the influence of a given input on the hidden layer decays exponentially as it cycles around the recurrent of the network. Long Short-Term Memory (LSTM) [21] and Gated Recurrent Unit (GRU) [22] are the most famous two famous variants of RNN to overcome the gradient vanishing problem. Therefore, we explore LSTM and GRU as well for global sequential feature extraction.

In addition, RNN extracts features at each step from the information of the inputs of the current step and the former steps. However, in sequence labeling tasks, the label of each step depends on the whole sentence instead of the former words. In order to extract information of the whole context, Bidirectional RNN (BRNN) was proposed [24] by adding a backward sequence in each hidden layer; thus, the output of each token is determined by the whole sequence. Therefore, we explore BRNN as well. We abbreviate the bidirectional model of SRNN, LSTM and GRU with BSRNN, BLSTM and BGRU, respectively.

E. Labeling

We use the *PNO* tagging scheme [3] to formulate sentiment parsing as a sequence labeling task. In the *PNO* tagging scheme, *P* indicates tokens inside a positive target, *N* denotes tokens inside a negative target and *O* for other tokens. In labeling layer, each label is encoded into a three-dimensional vector. For the label l_j of token t_j,

$$\hat{y}_j^l = \begin{cases} (0,0,1), where & l_i = P, \\ (0,1,0), where & l_i = N, \\ (1,0,0), where & l_i = O. \end{cases} \quad (4)$$

Labeling layer calculates a three-dimensional vector with the features extracted at each step and normalize it with a *softmax* function:

$$y_j^l = softmax(W_l y_j + b_l). \quad (5)$$

where W_l is a weight matrix and b_l is a bias vector. In such process, each element in y_j^l can be viewed as the probability of its related label. For instance, vector *(0.1, 0.3, 0.6)* denotes that the label of the corresponding token has the probability of *0.6* to be *P*, *0.3* to be *N*, and *0.1* to be *O*.

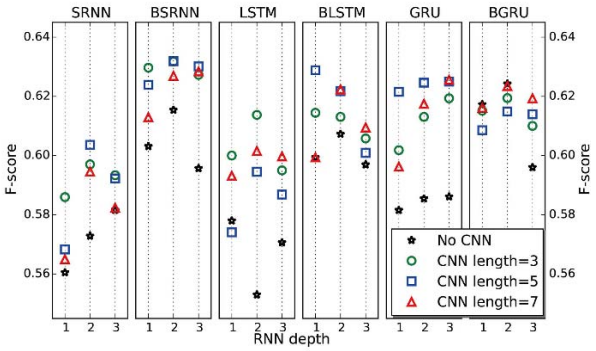

Figure 2. The F-scores of different RNN models with a CNN layer and without CNN. There are 6 sub figures. Each sub figure is one basic RNN model in unidirectional or bi-direction with depth 1-3.

IV. NETWORK TRAINING

Let $\theta = \{E, W_*^*, U_*^*, V_*^*, b_*^*\}$ be the set of model parameters. For each θ, convRNN model will calculate an output Y^l for a sentence S. Each column y_j^l of the output layer is a three-dimensional vector, which determines the label of t_j in the sentence. Because y_j is the output of *softmax* function, the summary of the elements in y_j is 1. We use the cross-entropy error of $\hat{y}_j^l$ and y_j^l as the loss of token t_j in sentence S:

$$L_{ce}(\hat{y}_j^l, y_j^l; \theta) = -\sum_{k=0}^{2} \hat{y}_{jk}^l log y_{jk}^l \ . \quad (6)$$

Let N be the number of sentences in the dataset. The loss function is the average value of all tokens in all sentences in the dataset:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{s_i} \sum_{j=1}^{s_i} L_{ce}(\hat{y}_j^l, y_j^l; \theta) \ . \quad (7)$$

V. EXPERIMENTS

A. Experiment Setting

We evaluate the proposed neural network architecture and different models on a Chinese microblogging dataset. There are 67,033 unlabeled messages and 5000 labeled sentences. We use the 67,033 unlabeled messages to train the initial word embedding and 5000 labeled sentences to train and test the performance of convRNN for sentiment parsing. We use F-score of extracted opinion tuples as the evaluation metric.

We explore Simple RNN, LSTM and GRU three models with unidirectional and bidirectional models in depth 1-3 in our experiments. In addition, considering that the lengths of microblogging sentences are often very short, we only explore convolution operation with length 3, 5 and 7 in depth 1-5. Therefore, we explore 270 models—18 different RNN models multiply 15 different convolution operations—in total in our experiments. We take 18 different RNN models as the baseline and focus on analyzing how different convolution operations affect RNN models.

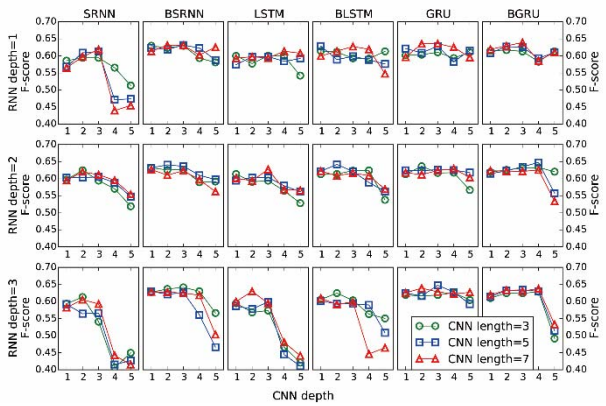

Figure 3. The F-scores of different convRNN models. There are 18 sub figures. Each column has the same basic RNN model and direction, and each line has the same RNN depth. Each subfigure displays the performance of 15 different convolution operations with the RNN model. The x-axis of each sub figure is the number of convolution layers and the lines in sub figures with same marker and color have the same CNN filter length.

Empirically, we set the dimension of word embedding to be 100 and the dimension of CNN and RNN to be 64. We use Adam optimizer [25] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and learning rate 0.001 to train the models with dropout rate 0.3.

B. Experimental Results

ConvRNNs VS. RNNs. We first check whether convolution operation improves RNN models. Figure 2 displays the F-scores of 18 different RNN models with 1 convolution layer, compared with no convolution layer. 16 of 18 RNN models achieve higher F-scores with a CNN layer. The highest improvement reaches 0.061 from 0.553 to 0.614, achieved by LSTM in depth 2 with convolutional length 3. The average improvement of adding a convolution layer is 0.019. Finally, convRNNs with a convolution layer have pushed the highest F-score of RNN models from 0.624 to 0.632. Therefore, the experiment results demonstrate that it is significant to add a convolution layer to extract local features of tokens for sentiment parsing.

Deep convRNNs. We then validate whether deep convRNNs perform better than the shallow ones and analyze the effect of different convolution hyper-parameters to convRNNs. Figure 3 shows the results of all 270 convRNNs. It can be seen that most convRNNs achieve better results with deep models than the shallow models. When convolution layers increases, the F-scores of most models increase or keep stable at first and decrease after a point. Moreover, when RNN depth increases, the decrease tends to become faster. Most convRNNs achieve the best performance with 2-4 convolution layers.

TABLE I. BEST PERFORMANCE OF DIFFERENT MODELS.

	F-Score	*CNN depth*	*RNN model*	*RNN depth*
Deep RNN(Baseline)	0.624	-	BGRU	2
Shallow CNN+ Shallow	0.630	-	BSRNN	-
Shallow CNN+ Deep RNN	0.632	-	BSRNN	2
Deep CNN+ Shallow RNN	0.639	3	BGRU	-
Deep CNN+ Deep RNN	0.648	4	BGRU	2

Best results. The best performance of different combinations of CNNs and RNNs are shown in Table I. All best performances are achieved by bidirectional RNN models. All the highest F-scores of convRNNs are higher than the baseline, demonstrating that extracting local features of words by convolution operation improves the result of RNNs. In addition, when CNN models become deep ones, the best performance improves significantly, indicating that multilayer of convolution operations can extract local features of words better than shallow ones. Finally, the best result of deep convRNNs is 0.648, achieved by a 2-depth bidirectional GRU with four convolution layers.

VI. CONCLUSIONS

In this paper, we propose convRNN to extract the local and global contexture features of words in a sentence for sentiment parsing of Chinese microblogging texts. Experimental results demonstrate that local feature extraction with convolution operations improves the performance of RNN models. Deep convRNN achieves the best F-score and outperforms the RNN-based method substantially by nearly 4%.

In future work, we are interested in some new representation of sentiment parsing task instead of sequence labeling. We will focus on some new neural networks such as attention network and memory network as well.

ACKNOWLEDGMENT

The research is supported by National Natural Science foundation of China (No. 71331008 and No. 61105124).

REFERENCES

[1] Andranik Tumasjan, Timm Oliver Sprenger, Philipp G Sandner, and Isabell M Welpe. Predicting elections with twitter: What 140 characters reveal about political sentiment. ICWSM, 10:178-185, 2010.

[2] Bernard J Jansen, Mimi Zhang, Kate Sobel, and Abdur Chowdury. Twitter power: Tweets as electronic word of mouth. Journal of the American society for information science and technology, 60(11):2169-2188, 2009.

[3] Jiajun Cheng, Xin Zhang, Pei Li, Sheng Zhang, Zhaoyun Ding, and Hui Wang. Exploring sentiment parsing of microblogging texts for opinion polling on Chinese public figures. Applied Intelligence, pages 1-14, 2016.

[4] Fandong Meng, Zhengdong Lu, Mingxuan Wang, Hang Li, Wenbin Jiang, and Qun Liu. Encoding source language with convolutional neural network for machine translation. arXiv preprint arXiv:1503.01838, 2015.

[5] Changli Zhang. Sentiment analysis of Chinese documents: From sentence to document level. Journal of the American Society for Information Science and Technology, 2009.

[6] Danushka Bollegala, David Weir, and John Carroll. Cross-domain sentiment classification using a sentiment sensitive thesaurus. Knowledge and Data Engineering, IEEE Transactions on, 25(8):1719-1731, 2013.

[7] Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse information sources. In EMNLP, volume 4, pages 412-418, 2004.18

[8] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of EMNLP, volume 1631, page 1642. Citeseer, 2013.

[9] Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, and Noah A Smith. From tweets to polls: Linking text sentiment to public opinion time series. ICWSM, 11(122-129):1-2, 2010.

[10] Ana-Maria Popescu and Orena Etzioni. Extracting product features and opinions from reviews. In Natural language processing and text mining, pages 9-28. Springer, 2007.

[11] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. A C-LSTM neural network for text classification. arXiv preprint arXiv:1511.08630, 2015.

[12] Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional LSTM-CNNs. arXiv preprint arXiv:1511.08308, 2015.

[13] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. arXiv preprint :1603.01354, 2016.

[14] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.

[15] Haibing Wu, Yiwei Gu, Shangdi Sun, and Xiaodong Gu. Aspect-based opinion summarization with convolutional neural networks. arXiv preprint arXiv:1511.09128, 2015.

[16] Puyang Xu and Ruhi Sarikaya. Joint intent detection and slot filling using convolutional neural networks. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 2014.

[17] Tomas Mikolov, Martin Karafit, Lukas Burget, Jan Cernock, and Sanjeev Khudanpur. Recurrent neural network based language model. In INTERSPEECH, pages 1045-1048, 2010.

[18] Majid Yazdani and James Henderson. Incremental recurrent neural network dependency parser with search-based discriminative training. In Proceedings of the 19th Conference on Computational Language Learning, pages 142-152, 2015.

[19] Lishuang Li, Liuke Jin, Zhenchao Jiang, Dingxin Song, and Degen Huang. Biomedical named entity recognition based on extended recurrent neural networks. In Bioinformatics and Biomedicine (BIBM), 2015 IEEE International Conference on, pages 649-652, Nov 2015.

[20] Ozan Irsoy and Claire Cardie. Opinion mining with deep recurrent neural networks. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 720{728, 2014.

[21] Sepp Hochreiter and Jrgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735-1780, 1997.

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

[23] Ji Young Lee and Franck Dernoncourt. Sequential short-text classification with recurrent and convolutional neural networks. arXiv preprint arXiv:1603.03827, 2016.

[24] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, 45(11), 1997.

[25] Kingma D, Ba J. Adam: A Method for Stochastic Optimization [J]. Computer Science, 2014.