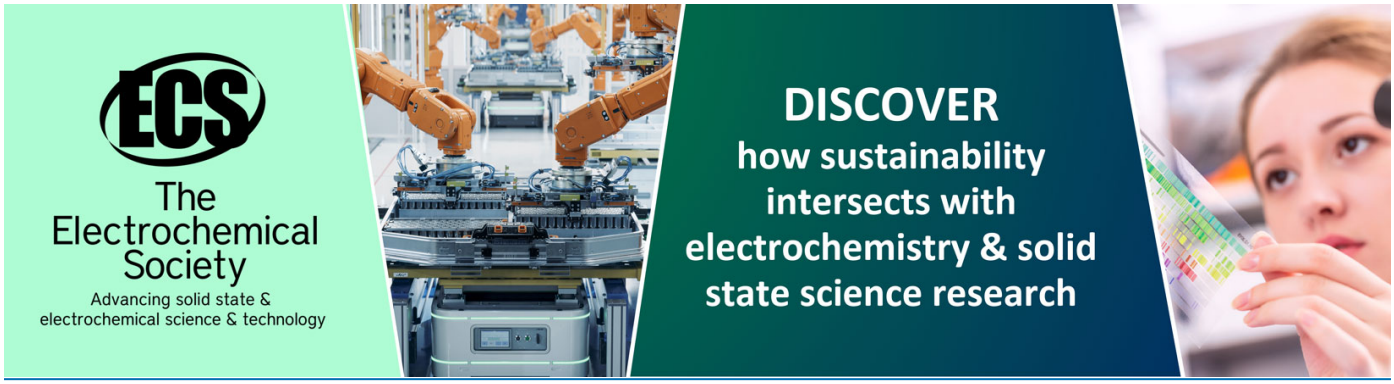
A Scalable Attention Mechanism Based Neural Network for Text Classification



A Scalable Attention Mechanism Based Neural Network for Text Classification

Jianyun Zheng[1], Jianmin Pang[1*], Xiaochuan Zhang[1], Di Sun[2], Xin Zhou[1], Kai Zhang[1], Dong Wang[1], MingLiang Li[1] and Jun Wang[1]

[1]State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, 450002, China

[2]Xi'an Institute of Surveying and Mapping, Xi'an, 710000, China

[*]Corresponding author's e-mail: jianmin_pang@hotmail.com

Abstract. In general, deep learning based text classification methods are considered to be effective but tend to be relatively slow especially for model training. In this work, we present a powerful, so-called "scalable attention mechanism", which performs better than conventional attention mechanism in terms of both effectiveness and the speed of model training. Based on the scalable attention mechanism, we propose a neural network for text classification. The experimental results on eight representative datasets show that our method can obtain similar accuracy to state-of-the-art methods with training in less than 4 minutes on an NVIDIA GTX 1080Ti GPU. To the best of our knowledge, our method is at least twice faster than all the published deep learning classifiers.

1. Introduction

In this paper, we focus on an important natural language processing (NLP) task—text classification—widely used in many applications, such as sentiment detection [5], article recommendation [15] and even vulnerability detection[12]. Many solutions for text classification[3][9][17][18][20] have achieved satisfactory results in practice. However, the vast majority of these solutions are time consuming, particularly when addressing large scale datasets [9][18] since these methods are based on either recurrent neural networks (RNN) or convolutional neural networks (CNN).

Attention mechanism is proved to be effective and fast in various of tasks [2][13][16][21]. But the conventional attention mechanism is not scalable, making it hard to capture the key part of a large input (we will elaborate it in the following section).

To overcome this shortage, in this work, we present the scalable attention mechanism which is superior to the conventional attention mechanism in terms of both the effectiveness and time consumption. Based on the scalable attention mechanism, we propose a neural network for text classification. We conduct experiments on an NVIDIA GTX 1080Ti GPU by training our neural network on eight representative datasets, accelerating the training process into 4 minutes without degradation of the accuracy of models.

2. Scalable Attention Mechanism

Attention mechanism based neural networks have demonstrated success in a wide range of tasks ranging from machine translations[13], speech recognition[2] and sentiment classification[16] to relation classification[21].

Attention mechanisms use the weight sum of the input matrix and a attention vector to represent the summary of the input, focusing on information linked to the task. The key point of attention mechanism is the calculation of the attention vector. In this paper, we focus on self-attention mechanism which is also known as intra-attention mechanism.

Let X be an input matrix. [21] described a typical method to calculate the attention vector, which is shown as follows:

$$\alpha = softmax\left(w^T tanh(X)\right)$$

where $X \in \mathbb{R}^{m \times n}$ and w^T is the transpose of an m-dimensional vector w. Note that to obtain the attention vector α, the only trainable parameter is w, which is a vector with limited dimensions. When processing inputs with a considerable number of rows, i.e., n is equal to a large number, it is challenging to obtain an effective attention vector α through w that has only m dimensions.

To capture the key part of a sentence more effectively, we propose a scalable attention mechanism for text classification. There are several sublayers in the scalable attention layer, which are used to produce an m-dimensional attention vector α, while each sublayer contains several neurons. Let X be a matrix consisting of input vectors $[x_1, x_2, \dots, x_n]$. The operation performed by a neuron is shown as follows:

$$y = f\left(\begin{bmatrix} w_1^T x_1 + b_1 \\ w_2^T x_2 + b_2 \\ \vdots \\ w_n^T x_n + b_n \end{bmatrix}^T\right)$$

where f is the activation function, $X \in \mathbb{R}^{m \times n}$, and m is the dimension of x_i. w_i is a trainable weight vector, and w_i^T is the transpose; b is a trainable bias vector, and y is the output vector. The dimensions of w_i, b and y are m, n and n, respectively. Figure1 schematizes operation of a neuron of the sublayers in the scalable attention layer. The number of sublayers, the number of neurons and the activation function are the hyperparameters to be chosen by users. Note that the sublayers ultimately output an n-dimensional vector α; hence, the last sublayer consists of only one neuron.

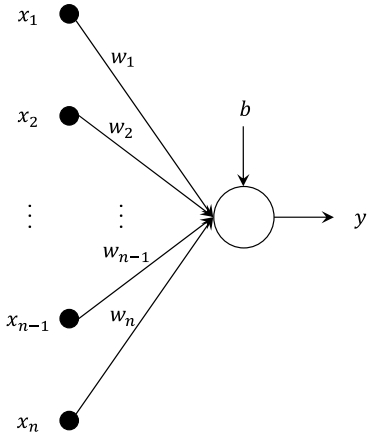


Figure 1. Operation Performed by the Neurons of the Sublayers in the Scalable Attention Layer.

The output of the scalable attention layer is calculated as follows:

$$h = tanh(X\alpha^T)$$

where h is the final output vector of the scalable attention layer with size m.

3. Model

In this section, we introduce our model in detail.

Figure 2 shows the model of our approach. The model is constructed with five components:

(1) Input layer: input sentences to this model;

(2) Word embedding layer: map words of sentences into vectors of real numbers;

(3) Scalable attention layer: capture the key part of the embedded sentence;

(4) Fully connected (FC) layers: collect the outputs from the scalable attention layer and make a final decision;

(5) Output layer: output the classification vectors.

These components are presented in detail in this section.

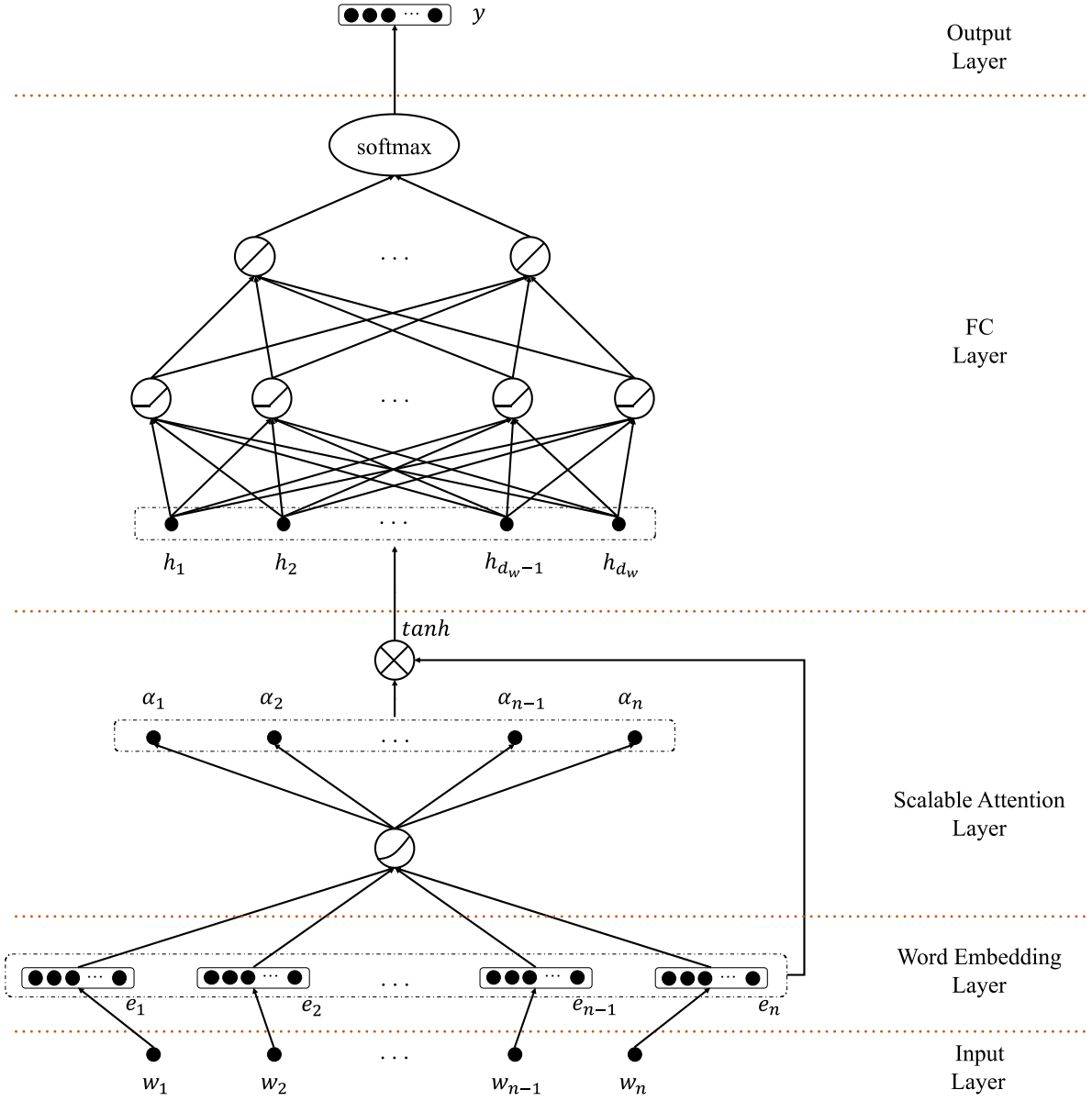


Figure 2. Text Classification Model with Scalable Attention.

3.1Word embedding layer

The purpose of the word embedding layer is to use dense vectors to represent words instead of using sparse vectors.

The input of the word embedding layer is a sentence consisting of N words $S = \{w_1, w_2, \dots, w_n\}$, where w_i is the one-hot encoding format of a word, i.e., $w_i \in \{0,1\}^{|V|}$ and V is a fixed-sized vocabulary. We convert w_i to its word embedding vector e_i as follows:

$$e_i = w_i E$$

where $E \in \mathbb{R}^{|V| \times d_w}$ and $e_i \in \mathbb{R}^{d_w}$, and d_w is a hyperparameter that we can set arbitrarily.

The output of this layer is the word embedded sentence $S_{emb} = \{e_1, e_2, \dots, e_n\}$.

3.2Scalable attention layer

We position a scalable attention layer following the word embedding layer. The scalable attention layer takes the embedded sentence S_{emb} produced in the word embedding layer and returns a d_w-dimensional vector h. In this model, we apply one sublayer to produce the attention vector α. In order to amplify/diminish the important/unimportant part of the input, unlike other attention mechanisms that use softmax function, we use softplus function[4] to make the attention factor of each embedded word ranging from 0 to $+\infty$. Thus, the operation of this layer can be denoted as follows:

$$\alpha = softplus\left(\begin{bmatrix} w_1^T e_1 + b_1 \\ w_2^T e_2 + b_2 \\ \vdots \\ w_n^T e_n + b_n \end{bmatrix}^T\right)$$

$$h = tanh(S_{emb}\alpha^T)$$

3.3FC layers

The neural network collects the outputs from the scalable attention layer and makes a final decision using two FC layers. The operation performed by these layers is denoted by L_1^{FC} and L_2^{FC}. We represent the outputs of the two FC layers as $f_1 = L_1^{FC}(h)$ and $f_2 = L_2^{FC}(f_1)$, where $f_1 \in \mathbb{R}^{|fc_1|}$ are intermediate vectors and $f_2 \in \mathbb{R}^k$ denotes the final output of the FC layers; $|fc_1|$ is the number of nodes in the first FC layer, and k is the count of labels. The first FC layer uses a rectified linear unit (ReLU) function[7] as its activation function, while the last FC layer uses an identity activation function. The final output is the softmax normalization of f_2.

3.4Training

In our design, the training objective is based on categorical cross-entropy. The loss function is expressed as follows:

$$L = -\frac{1}{N}\sum_{i}^{N}\sum_{j}^{k} t_{ij} log(p_{ij})$$

where N is the number of testing data, k is the count of labels, $t_{ij} \in T$ and $p_{ij} \in P$; T and P are the true labels and the predictions, respectively. We minimize the loss L in the training set using the Adam[10] gradient descent algorithm, which computes adaptive learning rates for each parameter. We also evaluate other gradient descent optimization algorithms, such as SGD[1] and RMSprop[14], but these approaches do not provide better results. The embedding length and the mini-batch size are set to 100 and 1000, respectively. The weights are all initialized according to Glorot's scheme[6], while biases are initialized with zeros.

4. Experiments

4.1Datasets and baselines

We employ the same 8 datasets from[20] to evaluate our model by comparing our method with the baselines. These datasets contain popular topics that use text classification, including news classification, sentiment analysis, Wikipedia article classification and questions and answers categorization. The scale of samples in these datasets ranges from hundreds of thousands to several millions. Table 1 presents a summary.

Table 1. Descriptive Statistics of the Datasets Used in Our Experiments.

Dataset	Topic	Classes	Train	Test Samples
AG	news classification	4	120,000	7,600
Sogou	news classification	5	450,000	60,000
DBP	Wikipedia article classification	14	560,000	70,000
Yelp.F	sentiment analysis	5	650,000	50,000
Yelp.P	sentiment analysis	2	560,000	38,000
Amz.F	sentiment analysis	5	3,000,000	650,000
Amz.P	sentiment analysis	2	3,600,000	400,000
Yah.A.	questions and answers	10	1,400,000	60,000

In this paper, we select several prior methods to serve as baselines. They are the bag of words (BoW)[20], n-grams[20], n-grams TFIDF variant[20], character level convolutional model (char-CNN)[20], character based convolution recurrent network (char-CRNN)[17], very deep convolutional network (VDCNN)[3], fastText[9], naïve Bayes[18], Kneser-Ney Bayes[18], MLP naïve Bayes[18], discriminative long short term memory (LSTM)[18], generative LSTM-independent comp[18], generative LSTM-share comp[18], word shallow-and-wide CNN (word-CNN)[11] and sliced recurrent neural network (SRNN)[19]. In addition, we compare the scalable attention mechanism with the conventional attention mechanism[21] by replacing the scalable attention layer by the conventional attention layer in our neural network.

4.2Results

We present the experimental results in Table 2.

Table 2. Test Accuracy [%] for Eight Datasets.

Model	AG	Sogou	DBP	Yelp.F	Yelp.P	Amz.F	Amz.P	Yah.A
BoW[20]	88.8	92.9	96.6	58	92.2	54.6	90.4	68.9
n-grams[20]	92	97.1	98.6	56.3	95.6	54.3	92	68.5
n-grams TFIDF[20]	92.4	97.2	98.7	54.8	95.4	52.4	91.5	68.5
char-CNN[20]	87.2	95.1	98.3	62	94.7	59.5	94.5	71.2
char-CRNN[17]	91.4	95.2	98.6	61.8	94.5	59.2	94.1	71.7
VDCNN [17]	91.3	96.8	98.7	64.7	95.7	63	95.7	73.4
fastText, h=10 [9]	91.5	93.9	98.1	60.4	93.8	55.8	91.2	72
fastText, h=10, bigram[9]	92.5	96.8	98.6	63.9	95.7	60.2	94.6	72.3
Naïve Bayes [18]	90.0	86.3	96.0	51.4	86.0	-	-	68.7
Kneser-Ney Bayes[18]	89.3	94.6	95.4	41.7	81.8	-	-	69.3
MLP Naïve Bayes[18]	89.9	76.1	87.2	40.4	73.6	-	-	60.6
Discriminative LSTM[18]	92.1	94.9	98.7	59.6	92.6	-	-	73.7
Generative LSTM-independent	90.7	93.5	94.8	51.9	90.0	-	-	70.5
Generative LSTM-share comp[18]	90.6	90.3	95.4	52.7	88.2	-	-	69.3
word-CNN[11]	92.2	-	98.7	64.9	95.9	-	-	73.0
SRNN[19]	92.0	96.0	98.1	62.3	95.4	60.4	95.0	72.3
Conventional Attention	**92.0**	**94.5**	**97.5**	**60.5**	**93.9**	**55.4**	**92.2**	**71.8**
Scalable Attention	**92.6**	**96.1**	**98.6**	**61.2**	**94.3**	**56.8**	**92.6**	**72.7**

A comparison of our method with other state-of-the-art methods of text classification, for sentiment analysis (corresponding to Yelp.F, Yelp.P, Amz.F and Amz.P), reveals that our method is better than BoW, n-grams, n-grams TFIDF, fastText and discriminative LSTM. Except for sentiment analysis, the accuracy obtained by the scalable attention mechanism based method is similar to that of VDCNN, fastText with bigram and word-CNN and slightly better than the other methods.

Relative to the conventional attention mechanism based neural network, our approach achieves better test results on all the eight datasets, improving the accuracy by approximately 0.5 to 1 percent.

5. Discussion

5.1Computational complexity

Table 3 shows the training time of the scalable attention mechanism based neural network, the conventional attention mechanism based neural network and SRNN[19] on eight datasets. The models are trained on an NVIDIA GTX 1080Ti GPU. To obtain the acceptable input of SRNN, sequences needs to be sliced into many subsequences firstly, causing extra time consumption for SRNN compared with attention mechanisms. The training time of SRNN in Table 3 includes the extra time needed for data preprocessing. On all eight datasets, the maximum training time of the proposed model is less than 4 minutes. Although the conventional attention mechanism is simpler than the scalable attention mechanism, the former takes more time to converge. In addition, our approach is at least twice faster than SRNN.

Except above methods and fastText, which is trained on CPUs, other baseline methods of this work are not open-sourced yet, to the best of our knowledge. Some works[9][18] have disclosed the training time of the previous methods. To train the smallest (largest) dataset, discriminative LSTM takes approximately 20 minutes (6 hours)[18]. In addition, for char-CNN, the training time for a single epoch on these datasets ranges from 1 hour to 5 days, while VDCNN takes 24 minutes to 7 hours[9]. Although the hardware platforms (char-CNN and VDCNN were trained on a NVIDIA Tesla K40 GPU, and[18] did not disclose the hardware platform on which the discriminative LSTM was tested) that they used for testing were different from our platform, the huge difference in training time supports the conclusion that our method is much faster than other deep learning based methods and is approximately on par with fastText, which takes less than one minute to train on a CPU using 20 threads[9].

Table 3. Training Time on Eight Datasets.

Dataset	Scalable Attention	Conventional Attention	SRNN
AG	13 s	13 s	40 s
Sogou	3 min 32 s	5 min 2 s	10 min 51 s
DBP	2 min 21 s	3 min 9 s	8 min 17 s
Yelp.F	2 min 23 s	4 min 11 s	5 min 6 s
Yelp.P	1 min 43 s	2 min 23 s	4 min 7 s
Amz.F	3 min 45 s	10 min 33 s	23 min 45 s
Amz.P	3 min 42 s	7 min 10 s	27 min 36 s
Yah.A.	3 min 43 s	12 min 20 s	17 in 19 s

5.2Overfitting

In practice, the scalable attention mechanism based neural network is susceptible to overfitting. To alleviate overfitting, we evaluated commonly used methods such as dropout[8] and regularization. However, these methods do not provide positive feedback.

Fortunately, when overfitting occurs in our model, the rate of decline in the accuracy of the test data is much slower than the rate at which the loss increases. Figure 3 shows the training history on

DBP. According to Figure 3, the overfitting occurs after 3 epochs. After overfitting occurs, the testing accuracy remains approximately unchanged even though the increase in the testing loss has doubled.

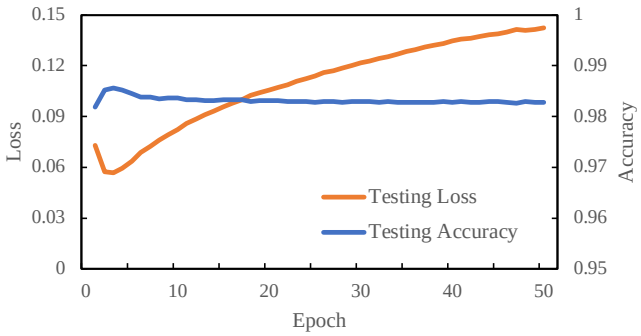


Figure 3. Training History on DBP.

6. Conclusion and Outlook

In this paper, we advance a powerful attention mechanism, the scalable attention mechanism, that performs better than the conventional attention mechanism in terms of both effectiveness and training time.

We propose a novel neural network model based on a scalable attention mechanism for text classification. We compared our model with recently proposed methods using several largescale datasets. The experimental results show that the accuracy of the scalable attention mechanism based neural network is on par with that of state-of-the-art methods while being much faster than other deep learning based methods.

In the future, we plan to apply the scalable attention mechanism to a broader range of tasks, particularly for NLP tasks.

Acknowledgments

We would like to thank anonymous reviewers for their feedback. This research was supported by Natural Science Foundation of China under Grant No. 61802433 and 61802435.

References

[1] Bottou, L. 1991. Stochastic Gradient Learning in Neural Networks. Proceedings of Neuro-Nımes.

[2] Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. 2015. Attention-based Models for Speech Recognition. In: Advances in Neural Information Processing Systems. New York. pp. 577–585.

[3] Conneau, A., Schwenk, H., Barreau, L., and Cun, Y. L. 2016. Very Deep Convolutional Networks for Text Classification. arXiv:1606.01781.

[4] Dugas, C., Bengio, Y., Belisle, F., Nadeau, C., and Garcia, R. 2001. Incorporating Second-Order Functional Knowledge for Better Option Pricing. In: Proceedings of the 13th International Conference on Neural Information Processing Systems. Denver. pp.451–457.

[5] Felbo, B., Mislove, A., Søgaard, A., Rahwan, I., and Lehmann, S. 2017. Using Millions of Emoji Occurrences to Learn Any-Domain Representations for Detecting Sentiment, Emotion and Sarcasm. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. Copenhagen. pp.1615–1625.

[6] Glorot, X., and Bengio, Y. 2010. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Sardinia. 249–256.

[7] Hahnloser, R. H. R., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., and Seung, H. S. 2000. Digital Selection and Analogue Amplification Coexist in a Cortex-Inspired Silicon Circuit. Nature, 405: 947-951.
[8] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Ruslan, R. 2012. Improving Neural Networks by Preventing Co-adaptation of Feature Detectors. Computer Science 3(4): 212–223.
[9] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. 2016. Bag of Tricks for Efficient Text Classification. arXiv:1607.01759.
[10] Kingma, D. P., and Ba, J. L. 2015. Adam: A Method for Stochastic Optimization. In: International Conference on Learning Representations (ICLR), 13. Ithaca.
[11] Le, H. T., Cerisara, C., and Denis, A. 2018. Do Convolutional Networks need to be Deep for Text Classification? In: Proceedings of the AAAI-18 Workshop on Affective Content Analysis. New Orleans.
[12] Li, Z., Zou, D., Xu, S., Ou, X., Jin, H., Wang, S., Deng, Z., and Zhong, Y. 2018. VulDeePecker: A Deep Learning-Based System for Vulnerability Detection. In: Proceedings 2018 Network and Distributed System Security Symposium (NDSS 2018), San Diego.
[13] Luong, M. T., Pham, H., and Manning., C. D. 2015. Effective Approaches to Attention-based Neural Machine Translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon. pp. 1412–1421.
[14] Tieleman, T., and Hinton, G. 2012. Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of its Recent Magnitude. COURSERA: Neural Networks for Machine Learning 4(2): 26–31.
[15] Wang, X., Yu, L., Ren, K., Tao, G., Zhang, W., Yu, Y., and Wang, J. 2017. Dynamic Attention Deep Model for Article Recommendation by Learning Human Editors' Demonstration. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Halifax. pp.2051–2059.
[16] Wang, Y., Huang, M., Zhu, X., and Zhao, L. 2016. Attention-based LSTM for Aspect-Level Sentiment Classification. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Austin. pp. 606–615.
[17] Xiao, Y., and Cho, K. 2016. Efficient Character-Level Document Classification by Combining Convolution and Recurrent Layers. arXiv:1602.00367.
[18] Yogatama, D., Dyer, C., Ling, W., and Blunsom, P. 2017. Generative and Discriminative Text Classification with Recurrent Neural Networks. arXiv preprint arXiv:1703.01898.
[19] Yu, Z., Liu, G. 2018. Sliced Recurrent Neural Networks. arXiv:1807.02291
[20] Zhang, X., Zhao, J., and LeCun, Y. 2015. Character-Level Convolutional Networks for Text Classification. In: Advances in Neural Information Processing Systems. MIT Press, Montreal. pp. 649–657.
[21] Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., and Xu, B. 2016. Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification. In: Association for Computational Linguistics. Berlin. pp. 207–212.