Hierarchical Convolutional Attention Networks Using Joint Chinese Word Embedding for Text Classification

Kaiqiang Zhang[1,2], Shupeng Wang[1], Binbin Li[1], Feng Mei[3](✉), and Jianyu Zhang[3]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{zhangkaiqiang,wangshupeng,libinbin}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3] Changan Communication Technology Co., LTD., Beijing, China
{meifeng,zhangjianyu}@chanct.com

Abstract. We propose a hierarchical convolutional attention network using joint Chinese word embedding for text classification. Compared with previous methods, our model has three notable improvements: (i) it considers not only words but also their characters and fine-grained sub-character components; (ii) it employs self-attention mechanisms with the benefits of convolution feature extraction, enable it to attend differentially to more and less important content; (iii) it has a hierarchical structure that can get the document vector. We demonstrate the effectiveness of our architecture by surpassing the accuracy of the current state-of-the-art on four classification datasets. Visualization of our hierarchical structure illustrates that our model is able to select informative sentences and words in a document.

Keywords: Joint Chinese word embedding · Self-attention · Hierarchical structure

1 Introduction

Text classification is one of the fundamental tasks in natural language processing (NLP). It has broad applications including topic labeling (Wang and Manning 2012), sentiment classification (Maas et al. 2011; Pang and Lee 2008), and spam detection (Sahami et al. 1998). Traditional approaches of text classification utilize features generated from vector space models such as bag-of words or term frequency-inverse document frequency (TF-IDF) (Sebastiani 2005), and then use a linear model or kernel methods on these features (Wang and Manning 2012; Joachims 1998).

More recently, deep learning approaches typically rely on architectures based on convolutional neural networks (CNN) (Blunsom et al. 2014) or recurrent

© Springer Nature Switzerland AG 2019
A. C. Nayak and A. Sharma (Eds.): PRICAI 2019, LNAI 11672, pp. 234–246, 2019.
https://doi.org/10.1007/978-3-030-29894-4_18

neural networks (RNN) (Young et al. 2017) to learn text representations. These newer approaches have been shown to outperform traditional approaches.

Among the two, RNN has attained remarkable achievement in handling serialization tasks. As RNN is equipped with recurrent network structure which can be used to maintain information, it can better integrate information in certain contexts. For the purpose of avoiding the problem of gradient exploding or vanishing in a standard RNN, long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) and other variants (Cho et al. 2014) have been designed for the improvement of remembering and memory accesses. Living up to expectations, LSTM does show a remarkable ability in the processing of natural language. Moreover, the other popular neural network, CNN, has also displayed a remarkable performance in computer vision (Krizhevsky and Hinton 2012), speech recognition, and natural language processing (Kalchbrenner et al. 2014) because of its remarkable capability in capturing local correlations of spatial or temporal structures. In terms of natural language processing, CNN is able to extract n-gram features from different positions of a sentence through convolutional filters and then it learns both short- and long-range relations through the operations of pooling.

Although neural-network-based approaches to text classification have been quite effective (Kim 2014; Zhang et al. 2015; Johnson and Zhang 2014; Tang et al. 2015), we still find the following two shortcomings.

Firstly, when it comes to document representation, most of the above methods use word-level distributed word representation. Among these embedding methods (Bengio et al. 2003; Mnih and Hinton 2009), CBOW and Skip-Gram models can learn good embeddings of words from large scale training corpora (Mikolov et al. 2013a, 2013b). However, when use these methods which treat each word as the minimum unit, it is easy to ignore the morphological information of words. There are also related works that used character-level features for text classification (Zhang et al. 2015)(charCNN). They first applied CNN only on characters and obtained the advantage that abnormal character combinations such as misspellings and emoticons may be naturally learnt.

In Chinese, the characters themselves are usually composed of sub-character components, which have semantic information. The components of a character can be roughly divided into two types: semantic component and phonetic component[1]. The semantic component represents the meaning of the character, while the phonetic component represents the sound of the character. For example (see Figs. 1 and 2), we intercepted a sentence from the Fudan text classification dataset C36-Medical007 and extracted the keyword "症" (symptom).

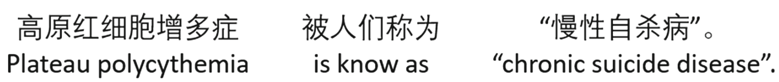

Fig. 1. Component example.

[1] https://en.wikipedia.org/wiki/Written_Chinese.

Fig. 2. Semantic and phonetic component. (Color figure online)

As mentioned above, "症" consists of "疒" (green part in the figure) and "正" (blue part in the figure). "疒" (sick) is the semantic component, while "正 (positive)" is the phonetic component. If methods pay more attention to the character "症" and the component "疒", it is easy to correctly classify this document into the Medical class.

Secondly, not all parts of a document are equally relevant for a query. In the above example, "高原红细胞增多症" (Plateau polycythemia) plays a more important role than "被人们称为" (is known as) in the final classification result.

The current state-of-the-art in text classification are Hierarchical Attention Networks (HAN), developed by Yang et al. (2016). HAN use a hierarchical structure in which each hierarchy uses the same architecture - a bidirectional RNN with gated recurrent units (GRU) (Chung et al. 2014), followed by an attention mechanism that creates a weighted sum of the RNN outputs at each timestep. The HAN processes documents by first breaking a long document into its sentence components, then processing each sentence individually before processing the entire document. By breaking a document into smaller, more manageable chunks, the HAN can better locate and extract critical information useful for classification. This approach surpassed the performance of all previous approaches across several text classification tasks.

Our work focuses mainly on Chinese text classification, which is known as a completely different language from English. We propose Hierarchical Convolutional Attention Network using Joint Chinese Word Embedding(HCAje), an architecture based off joint Chinese word embedding that can generate document representations from words as well as their characters and fine-grained sub-character components. Meanwhile, we use the convolution feature extraction to improve the self-attention architecture (Vaswani 2017) and adapt it into an effective approach for text classification. To evaluate the performance of our model in comparison to other common classification architectures, we look at 4 Chinese data sets. HCAje can achieve accuracy that surpasses the current state-of-the-art on several classification tasks.

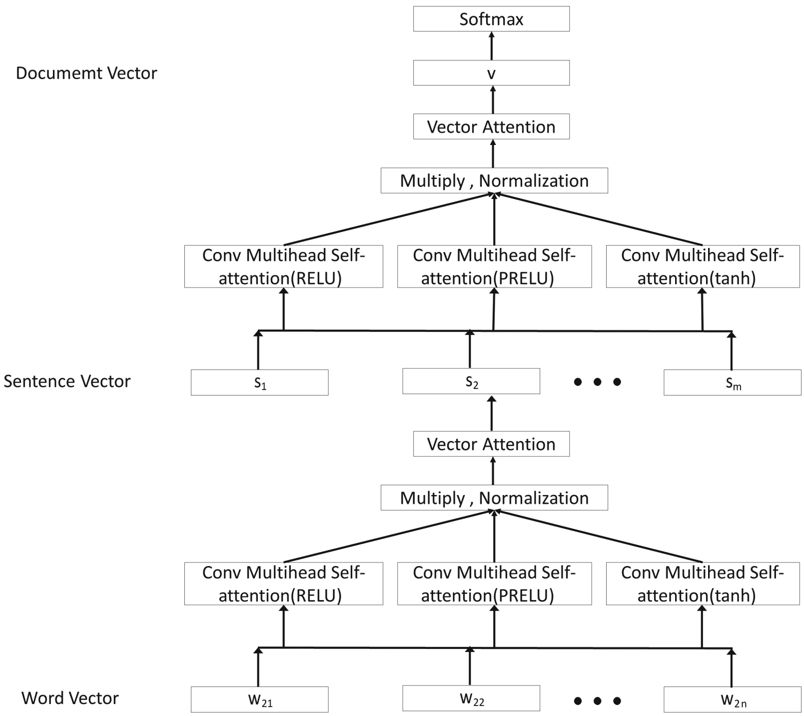

Fig. 3. Architecture for HCAje.

2 Hierarchical Convolutional Attention Network Using Joint Chinese Word Embedding

The overall architecture of our HCAje is shown in Fig. 3. It consists of several parts: joint Chinese word embedding, hierarchical structure, convolution feature extraction, parallel convolutional multihead self-attention and vector attention. We describe the details of different components in the following sections.

2.1 Joint Chinese Word Embedding

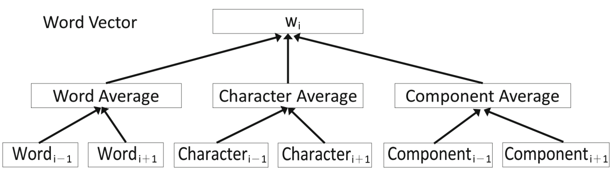

Fig. 4. Illustration of joint Chinese word embedding.

Our joint Chinese word embedding model is based on CBOW model (Mikolov et al. 2013a) and combines words, characters and components information. In the Sect. 3.5, we verified the effects of different combinations, and the following combinations worked best. It uses the average of context word vector, the average of context character vector and the average of context component vector to predict the target word, and use the sum of these three prediction losses as the objective function. We denote D as the training corpus, $W = (w_1, w_2, ..., w_n)$ as the vocabulary of words, $C = (c_1, c_2, ..., c_m)$ as the vocabulary of characters, $O = (o_1, o_2, ..., o_k)$ as the vocabulary of component, and T as the context window size respectively. As illustrated in Fig. 4, our model aims to maximize the sum of log-likelihoods of three predictive conditional probabilities for target w_i:

$$L(w_i) = \sum_{k=1}^{3} \log P(w_i \mid h_{i_k}) \tag{1}$$

where h_{i_1}, h_{i_2}, h_{i_3} are the composition of context words, context characters, context components respectively. Let v_{w_i}, v_{c_i}, v_{o_i} be the "input" vectors of word w_i, character c_i, and component o_i respectively, $\hat{v}_{w_i}$ be the "output" vectors of word w_i. The conditional probability is defined by the softmax function as follows:

$$p(w_i \mid h_{i_k}) = \frac{\exp\left(h_{i_k}^T \hat{v}_{w_i}\right)}{\sum_{j=1}^{N} \exp\left(h_{i_k}^T \hat{v}_{w_j}\right)}, \ k = 1,2,3 \tag{2}$$

where h_{i_1} is the average of the "input" vectors of words in the context:

$$h_{i_1} = \frac{1}{2T} \sum_{-T \leq j \leq T, j \neq 0} v_{w_{i+j}} \tag{3}$$

Similarly, h_{i_2} is the average of character "input" vectors in the context, h_{i_3} is the average of component "input" vectors in the context. Given a corpus D, our model maximizes the overall log likelihood:

$$L(D) = \sum_{w_i \in D} L(w_i) \tag{4}$$

where the optimization follows the implementation of negative sampling used in CBOW model (Mikolov et al. 2013a).

2.2 Convolution Feature Extraction

As mentioned in the Introduction, our model refers to Scaled Dot Product Attention which is a type of self-attention and multihead attention developed by Vaswani et al. Rather than use the same input for Q, K, and V, we used a convolution to extract different features from input for each of the Q, K, and V embeddings. This allows for more expressive comparison between entries in a sequence; for example, certain features may be useful when comparing Q and

K but may not be necessary when creating the output sequence from V. For our feature extractor function, we use a 1D convolution with d filter maps and a window size of three words, which provides more context for each center word when extracting important features.

$$\begin{aligned} Q &= GELU(Conv1D(E, W^q) + b^q) \\ K &= GELU(Conv1D(E, W^k) + b^k) \\ V &= GELU(Conv1D(E, W^v) + b^v) \end{aligned} \tag{5}$$

In the equation above, $Conv1D(A, B)$ is a 1D convolution operation with A as the input as B as the filter. We found gaussian error linear units (GELU) (Hendrycks and Gimpel 2016) to perform better than rectified linear units (ReLUs) and other activation functions. The GELU nonlinearity is the expected transformation of a stochastic regularizer which randomly applies the identity or zero map to a neuron's input. The GELU nonlinearity weights inputs by their magnitude, rather than gates inputs by their sign as in ReLUs.

2.3 Parallel Multihead Convolutional Self-attention

As we know, attention mechanisms are designed to produce a weighted average of an input sequence. However, a weighted average is not sufficient when capture the overall content within a linguistic sequence. To better capture the linguistic information, we use two multihead convolutional self-attentions (Eq. 6) in parallel followed by elementwise multiplication. Compared to simple weighted average, our approach can capture more complex interactions between elements in the sequence. After many attempts, we found that the combination of these two activation functions obtains the best performance. Among them, tanh outputs a value between −1 and 1, and prevents the final output from becoming too small or large after elementwise multiplication. GELU' s convergence rate is rapid.

$$\begin{aligned} Parallel\,(E) &= Multihead\,(Q^a, K^a, V^a) \\ &\odot Multihead\left(Q^b, K^b, V^b\right) \\ where\ Multihead\,(Q, K, V) &= [head_1, ..., head_h] \\ head_i &= Attention\,(Q, K, V) \\ Q^a &= GELU\,(Conv1D\,(E, W^{qa}) + b^{qa}) \\ K^a &= GELU\left(Conv1D\left(E, W^{ka}\right) + b^{ka}\right) \\ V^a &= GELU\,(Conv1D\,(E, W^{va}) + b^{va}) \\ Q^b &= tanh\left(Conv1D\left(E, W^{qb}\right) + b^{qb}\right) \\ K^b &= tanh\left(Conv1D\left(E, W^{kb}\right) + b^{kb}\right) \\ V^b &= tanh\left(Conv1D\left(E, W^{vb}\right) + b^{vb}\right) \end{aligned} \tag{6}$$

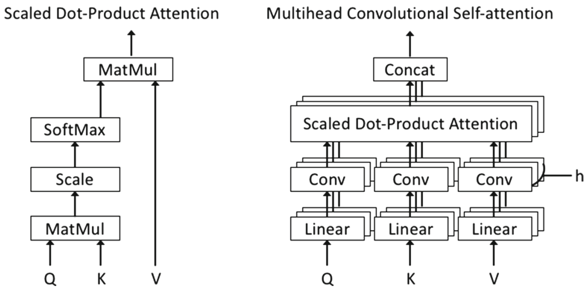

Fig. 5. Scaled dot-product attention and multihead convolutional self-attention.

2.4 Vector Attention

The output of parallel multihead convolutional self-attention is a sequence $E^{output} \in R^{l \times d}$ in which l is the length of the input sequence, and d is the embedding dimension. To obtain a single fixed-length vector which represents each sequence, we introduce vector attention which is same as the traditional attention mechanism that is used on RNN. For word level,

$$u_{it} = \tanh\left(W_w z_{it} + b_w\right) \tag{7}$$

$$\alpha_{it} = \frac{\exp\left(u_{it}^\top u_w\right)}{\sum_t \exp\left(u_{it}^\top u_w\right)} \tag{8}$$

$$s_i = \sum_t \alpha_{it} z_{it} \tag{9}$$

Firstly, we feed the word vector z_{it} (obtained by w_{it} through parallel multihead convolutional self-attention and elementwise multiplication) through a one-layer MLP to get u_{it} as a hidden representation of z_{it}. Then we measure the importance of the word as the similarity of u_{it} with a word level context vector u_w and get a normalized importance weight α_{it} by softmax. After that, we compute the sentence vector s_i as a weighted sum of the word annotations based on the weights. The context vector u_w is randomly initialized and learned in training process.

For sentence level,

$$u_i = \tanh\left(W_s z_i + b_s\right) \tag{10}$$

$$\alpha_i = \frac{\exp\left(u_i^\top u_s\right)}{\sum_i \exp\left(u_i^\top u_s\right)} \tag{11}$$

$$v = \sum_i \alpha_i z_i \tag{12}$$

where v is the document vector that summarizes all the information of sentences in this document, z_i is obtained by s_i through parallel multihead convolutional self-attention and elementwise multiplication, u_s is a sentence level context vector.

2.5 Hierarchical Structure

We utilize a hierarchical structure that breaks up documents into sentences and attained state-of-the-art performance. This structure consists of two levels: the word level and the sentence level. The word level reads in word vectors from a given sentence and outputs a sentence vector representing the content within that sentence, and the sentence level reads in the sentence vectors and outputs a document vector representing the content of the entire document. Each hierarchical consists of two parallel multihead convolutional self-attentions followed by a normalization layer and an vector attention layer (see Fig. 3).

2.6 Document Classification

The document vector v is our final representation of the document and can be fed into a softmax and used for classification purposes:

$$p = \mathrm{softmax}\left(W_d v + b_d\right) \tag{13}$$

We use the negative log likelihood of the correct labels as training loss:

$$L = -\sum_{d} j_d \log p_d \tag{14}$$

where j is the label of document d.

3 Experiments

3.1 Datasets

We evaluate the effectiveness of our model on three document classification datasets. The statistics of the datasets are summarized in Table 1. We use 80% of the data for training and the rest are used as testing set to evaluate the performance.

Table 1. Datasets.

Dataset	Categories	Documents	Train Samples	Test Samples
Fudan-large	5	6,121	4,894	1,227
Fudan-small	5	294	233	61
THUCNews	14	21,000	16,800	4,200
SogouNews	11	50,000	40,000	10,000

Fudan corpus[2] contains 20 categories of documents, including economy, politics, sports and etc. The number of documents in each category ranges from 27

[2] http://www.nlpir.org/download/tc-corpus-answer.rar.

to 1061. In this paper, we refer to the processing method by Zhang et al. (2015). To avoid imbalance, we select 10 categories and organize them into 2 groups. One group is named Fudan-large and each category in this group contains more than 1000 documents. The other is named Fudan-small and each category contains less than 100 documents. The publish information for each document is also removed because it contains strong indication of the categories, which will bias the classifier with unfair benefits.

THUCNews is obtained from (Guo et al. 2016). It consists of 740,000 news spanning 2005 to 2011. For our evaluation, we selected 14 popular categories and extracted 1,500 randomly selected news from each: economy, lottery, real estate, stock, home, education, technology, society, fashion, politics, sports, constellation, entertainment and game.

Sogou news is a combination of the SogouCA and SogouCS news corpora (Wang et al. 2008), containing in total 2,909,551 news articles in various topic channels. We then labeled each piece of news using its URL, by manually classifying their domain names. This gives us a large corpus of news articles labeled with their categories. We choose 14 categories and extracted 50,000 randomly selected news from each: sports, finance, entertainment, automobile, house, education, travel, female, culture, health and technology.

3.2 Baselines and Hyperparameters

To offer fair comparisons to competitive models, we conducted a series of experiments with both traditional methods such as **Naïve Bayes (NB)** and **logistic regression (LR)**. For logistic regression, we use L1 regularization with a penalty strength of 1.0. We also compare HCAje with five deep learning methods.

Word-based CNN like that of (Kim 2014) are used. We use three parallel convolution layers with 3-, 4-, and 5-word windows, all with 100 feature maps and apply 50% dropout on the concatenated vector. We use the pretrained word2vec embedding.

Character-based CNN are reported in (Zhang et al. 2015). To ensure fair comparison, the models for each case are of the same size as Word-based CNN, in terms of both the number of layers and each layer' s output size.

Bi-directional GRU We also offer a comparison with a recurrent neural network model, namely Bi-directional Gated Recurrent Unit. The Bi-directional GRU model used in our case is word-based, using pretrained word2vec embedding as in previous models.

Hierarchical Attention Networks (HAN) are the current state-of-the-art. For our experiments, we use the same optimized hyperparameters as those used by Yang - each hierarchy is composed of a bi-directional GRU with 50 units and an attention mechanism with a hidden layer of 200 neurons.

BERT or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. We fine-tuned Google's pre-trained Chinese Bert model on the data set mentioned above.

For our HCAje, we tuned the hyperparameters on the remaining text of the Sogou news. We use 8 heads for our final implementation. As for joint Chinese word embedding, we adopt the Chinese Wikipedia Dump[3] as our training corpus and removed pure digits and non-Chinese characters. After Chinese word segmentation and POS tagging, we obtained a 1 GB training corpus with 153,071,899 tokens and 3,158,225 unique words. The components information of Chinese character is crawled from HTTPCN[4]. For fair comparison, we used the same parameter settings with the pretrained word2vec embedding. We fixed the word vector dimension to be 100, the window size to be 5, the training iteration to be 100, the initial learning rate to be 0.025, and the subsampling parameter to be. Words with frequency less than 5 were ignored during training. We used 10-word negative sampling for optimization.

3.3 Model Configuration and Training

For training, we use a mini-batch size of 64 and documents of similar length (in terms of the number of sentences in the documents) are organized to be a batch. All models are trained using the Adam optimizer (Kingma and Ba 2015) with learning rate 2E–5, beta1 0.9, and beta2 0.99. We save the model parameters with the highest validation accuracy and use those parameters to evaluate on the test set.

3.4 Results and Analysis

The experimental results on all datasets are shown in Table 2. For each model, we record the final test accuracy. Results show that HCAje gives the best performance across all datasets.

The improvement is regardless of data sizes. For small datasets such as Fudan-small and Fudan-large, our model outperforms the previous baseline methods by 1.7% and 1.4% respectively. This finding is consistent across other larger datasets. Our model outperforms previous best models by 2.2% and 2.5% on THUCNews and Sogou news.

Within the HCAje, using joint Chinese word embedding achieves better accuracy than using word2vec, using two parallel multihead convolutional self-attentions achieves better accuracy than using one or three, and using vector attention outperforms using maxpool. Here PCMS stands for parallel convolutional multihead self-attention.

[3] http://download.wikipedia.com/zhwiki.
[4] http://tool.httpcn.com/zi/.

Table 2. Test set accuracy.

Methods	Fudan-small	Fudan-large	THUCNews	Sogou news
Naïve Bayes	65.28	74.95	84.36	80.97
Logistic Regression	68.30	76.57	84.91	82.65
Word-based CNN	70.05	78.58	85.81	83.61
Character-based CNN	70.34	78.88	85.73	84.23
Bi-directional GRU	70.70	78.84	85.53	84.07
HAN	70.92	79.75	86.76	85.67
HCAje(word2vec, 2 PCMS, maxpool)	70.81	79.43	86.38	85.13
HCAje(word2vec, 2 PCMS, vector attention)	71.29	80.15	87.07	85.89
HCAje(joint word embedding, 2 PCMS, maxpool)	70.89	79.51	86.58	85.27
HCAje(joint word embedding, 1 PCMS, vector attention)	70.44	79.62	86.12	85.49
HCAje(joint word embedding, 3 PCMS, vector attention)	71.07	79.70	86.74	85.66
HCAje(joint word embedding, 2 PCMS, vector attention)	**72.12**	**80.87**	**88.67**	**87.81**

3.5 Different Combinations of Embedding

As mentioned above, Chinese words can be divided into words, characters and components. We try different combinations to represent Chinese words, and test the classification effect on four datasets. The results are shown Table 3. By comparing the experimental results, the combination of words, words and components has the best classification results, while the combination of words and components has the worst classification results.

Table 3. Test set accuracy.

Methods	Fudan-small	Fudan-large	THUCNews	Sogou news
HCAje(word, character)	71.01	78.43	83.18	84.06
HCAje(word, component)	71.21	79.5	84.19	84.21
HCAje(character, component)	70.75	80.20	83.14	82.31
HCAje(word, character, component)	**72.12**	**80.87**	**88.67**	**87.81**

3.6 Visualization of Attention

In order to validate that our model is able to select informative sentences and words in a document, we visualize the hierarchical structure in Fig. 3 for one document from the sports classification of Fudan corpus. Every line is a sentence (the upper lines are the original Chinese texts, and the lower lines are the translated English texts). Red denotes the importance of sentences and blue denotes the importance of words (The darker the color, the more important it is). Figure 6 shows that our model can select the words carrying strong information like “国际足联 (the FIFA)”, “世界杯 (World Cup)” and their corresponding sentences.

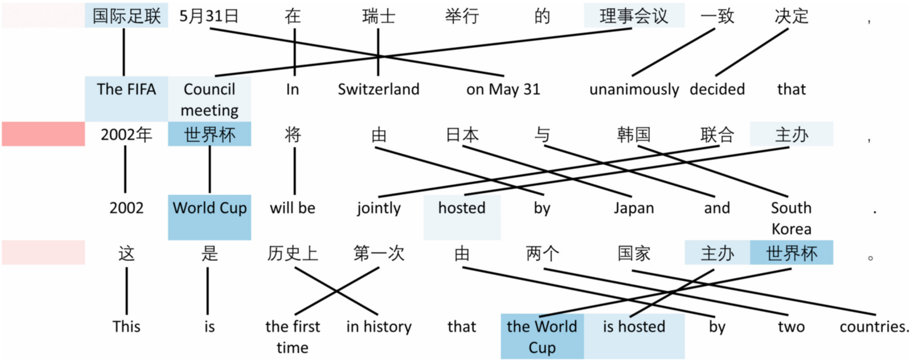

Fig. 6. Visualization of attention.

4 Conclusion

In this paper, we introduced a new self-attention based Chinese text classification architecture, HCAje, and compared its performance with the traditional approaches and deep learning approaches, in four classification datasets. In all four tasks HCAjes achieved slightly better performance than previous methods. Visualization of these attention layers illustrates that our model is effective in picking out important words and sentences. Although our model is introduced to classified Chinese text, the idea can also be applied to other languages that share a similar writing system. In the future, we would like to further explore learning representations for traditional Chinese words and Japanese Kanji.

References

Wang, S., Manning, C.D.: Baselines and bigrams: simple, good sentiment and topic classification. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers, vol. 2, pp. 90–94. Association for Computational Linguistics (2012)

Guo, Z., Zhao, Y., Zheng, Y., Si, X., Liu, Z., Sun, M.: THUCTC: An Efficient Chinese Text Classifier (2016)

Wang, C., Zhang, M., Ma, S., Ru, L.: Automatic online news issue construction in web environment. In: Proceedings of the 17th International Conference on World Wide Web, WWW 2008, New York, NY, USA, pp. 457–466 (2008)

Pang, B., Lee, L.: Opinion mining and sentiment analysis. Found. Trends Inf. Retrieval **2**(1–2), 1–135 (2008)

Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)

Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv 2014. arXiv:1406.1078 (2014)

Krizhevsky, A.I.S.A., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012, pp. 1097–1105 (2012)

Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. arXiv 2014. arXiv:1404.2188 (2014)
Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 142–150. Association for Computational Linguistics (2011)
Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A Bayesian approach to filtering junk e-mail. In: Learning for Text Categorization: Papers from the 1998 workshop, vol. 62, pp. 98–105 (1998)
Sebastiani, F.: Text Categorization [OL]. Encyclopedia of Database Technologies and Applications (2005)
Joachims, T.: Text categorization with Support Vector Machines: learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 137–142. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0026683
Blunsom, P., Grefenstette, E., Kalchbrenner, N., et al.: A convolutional neural network for modelling sentences. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (2014)
Young, T., Hazarika, D., Poria, S., Cambria, E.: Recent trends in deep learning based natural language processing. arXiv preprint arXiv:1708.02709 (2017)
Hendrycks, D., Gimpel, K.: Gaussian Error Linear Units (GELUs). arXiv:1606.08415 (2016)
Kim, Y.: Convolutional neural networks for sentence classification. In: EMNLP, pp. 1746–1751 (2014)
Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: NIPS, pp. 649–657 (2015)
Tang, D., Qin, B., Liu, T.: Document modeling with gated recurrent neural network for sentiment classification. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1422–1432 (2015)
Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. **3**(Feb), 1137–1155 (2003)
Mnih, A., Hinton, G.E.: A scalable hierarchical distributed language model. In: Proceedings of NIPS, pp. 1081–1088 (2009)
Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of NIPS, pp. 3111–3119 (2013)
Vaswani, A., et al.: Attention is all you need. In: NIPS (2017)
Lin, Z., et al.: A structured self-attentive sentence embedding. In: ICLR (2017)
Cheng, J., Dong, L., Lapata, M.: Long Short-term Memory-networks for Machine Reading, pp. 551–561 (2016)
Paulus, R., Xiong, C., Socher, R.: A deep reinforced model for abstractive summarization. arXiv preprint arXiv:1705.04304 (2017)
Kingma, D.P., Ba, J.L.: Adam: a method for stochastic optimization. In: ICLR (2015)