Improved Intelligent Semantics Based Chinese Sentence Similarity Computing for Natural Language Processing in IoT

Jiahao Ye[3], Lin Zhang[3], Ping Lan[1], Hua He[2], Dan Yang[2,4], and Zhiqiang Wu[1,2(✉)]

[1] College of Engineering, Tibet University, Lhasa 850000, China
lightnesstibet@163.com
[2] Center of Tibetan Studies (Everest Research Institute), Tibet University, Lhasa 850000, China
[3] School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou 510006, China
[4] Beijing Foreign Studies University, Beijing 10089, China

Abstract. It is desired in the Internet of Things (IoT) networks to apply natural language processing (NLP) technology to complete the information exchange tasks such as text summary or text classification between IoT devices. To achieve higher precision for the NLP of Chinese sentences, in this paper, we propose to utilize the deep neural network (DNN) to compute the semantic similarity of Chinese sentences. The proposed DNN consists of the input layer, the semantic generation layer, the concat layer, the dropout layer, the hidden layer, and the output layer. We propose to train the intelligent semantic similarity calculator sequentially to extract the semantic feature and the context information feature. After the offline training, the resultant configured intelligent semantic similarity calculator could evaluate the semantic similarity of Chinese sentences. Furthermore, we provide numerical analysis to demonstrate the improved similarity calculation precision and the consistency of the calculation accuracy in different fields.

Keywords: Similarity computing · Chinese sentences · Deep neural network · Semantics

1 Introduction

Along with the increasing demands of Internet of things (IoT) devices for extracting the feature from the texts, the sentence similarity calculation technology has been widely applied in natural language processing (NLP). With the aid of the sentence similarity evaluation, IoT devices could provide information retrieval in search engines [10], generate the web page text summary [3], and implement the text classification [5], etc.

B. Li et al. (Eds.): IoTaaS 2020, LNICST 346, pp. 234–246, 2021.
https://doi.org/10.1007/978-3-030-67514-1_19

According to the classification strategy of data sources, the scheme of text similarity can be divided into two categories: (1) the calculation scheme based on the knowledge base; and (2) the calculation scheme based on the neural network. On the one hand, the similarity calculation based on the knowledge base, such as the ontology database [12], can quantify the semantic similarity between words by evaluating the shortest phrase semantic distance of words in the tree knowledge base, the shallowest common parent node depth, and the node density. Then the preliminary sentence similarity could be calculated with the weighted sum of word similarity since all sentences are composed of words. Subsequently, the sentence similarity could be evaluated with the consideration of the length, grammar, and location of words to achieve semantic disambiguation [12]. On the other hand, the similarity calculation scheme based on the neural network uses the neural network to learn and extract the context information features of the data set to generate the distribution probability of word semantics. This kind of intelligent similarity calculation method uses the context information in the comparison sentences to judge the specific semantics of words in the sentences and quantify the similarity of words. In [1], the authors propose to calculate the similarity of sentences via a weighted average of the word vectors to obtain the sentences vector, and the singular value decomposition is applied to remove the irrelevant singular value to achieve higher accuracy of similarity calculation.

The first method has the advantage of being capable of providing consistent calculation accuracy in different fields. However, the ontology knowledge base might not be renewed in time, thereby leading to lower similarity calculation accuracy. By contrast, the second method provides higher calculation accuracy, but it has the disadvantage of inconsistent calculation accuracy in different fields.

Against this background, in this paper, we propose to utilize the preliminary semantic similarity calculation results as the data set for further evaluation with the deep neural network which consists of the input layer, the semantic generation layer, the concat layer, the dropout layer, the hidden layer, and the output layer.

In our design, we first apply the encoder to generate the sentence coding results, wherein we quantify the semantic similarity between words by evaluating the shortest phrase semantic distance of words in the tree knowledge base and the shallowest common parent node depth. Then we compose the deep neural network (DNN) based on the long short-term memory (LSTM) [8] to generate a normalized classification probability vector via the multi-layer perception(MLP), the dropout layer, and the softmax function, which determines the sentence similarity. At the offline training stage, the mixed semantics of the text are used to train the neural network to calculate the sentence similarity. The similarity of sentences at the online deployment stage is sequentially calculated by the encoder and the DNN.

Thus with the aid of the sequential processing for Chinese sentences with the serially concatenated siamese neural network encoder and the DNN, the consistency of the similarity calculation can be improved to achieve higher accuracy. Subsequently, we use the large-scale Chinese question matching corpus

(LCQMC) [7] as the dataset. Then we analyze and compare the performances of the proposed design, including the accuracy, precision, F1 score, and recall with counterpart systems.

Briefly, the main contributions of this paper include:

1. The output of bidirectional encoder representation from transformers (BERT) constructed in this paper is the encoding of all tokens of input text, while traditional BERT encoder is the rough semantic representation of the input text under the constraint of the downstream task model. Compared with the traditional output method, our scheme could achieve a better semantic representation.
2. To evaluate the length of the input text, in our design, except for the evaluation of the semantics, the length feature of the input text could also be provided. By contrast, no length feature could be identified in the traditional BER encoder.
3. The output mode constructed in this paper is input to the downstream task model in a sequential manner, which includes the position feature of the text. In the traditional BERT encoder, the length of a token output by the encoder cannot express the position feature of the input text.

The remaining part of the paper is organized as follows. The proposed similarity calculation model is detailed in Sect. 2, followed by results and the associated analysis in Sect. 3. Conclusions are given in Sect. 4.

2 System Model

In this section, we will present the similarity calculation details, including the sentence input vector generation, the training of the encoder, the similarity calculation, and the classification.

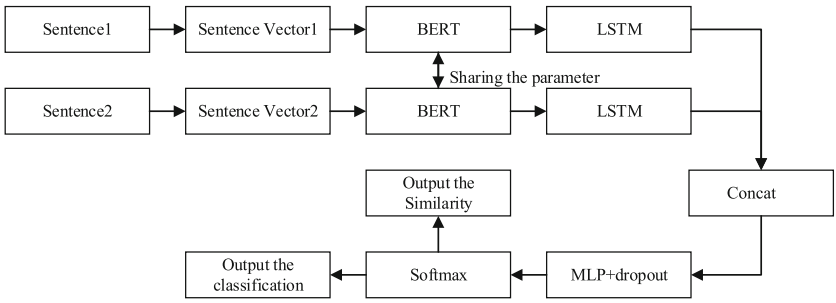

Fig. 1. The framework of the proposed sentences similarity calculation scheme.

As illustrated in Fig. 1, the proposed intelligent sentence similarity calculator is serially concatenated by the BERT [4] and the LSTM based DNN.

In this calculator, the input sentences are firstly represented as 1×512-dimensional sentence input vectors, which act as the input to the siamese neural network encoder. After the sentences encoding vectors pass through the LSTM module, the corresponding sentence overall information vectors are generated. Subsequently, the proposed calculator splice the whole sentence variables into pieces, and the spliced vectors are input into the MLP and dropout layers. The resultant 1×2-dimensional probability vector is then used for the classification by applying the Softmax function. Finally, the classification probability vector is generated.

Notably, the value of the second element of the classification probability vector represents the obtained similarity calculation results, while the category represented by the maximum value of the elements in the vector denotes the sentence classification result. At the offline training stage, the intelligent encoder and the classifier are trained separately. At the online deployment stage, the similarity of sentences could be evaluated and output to the terminals. More details about how to generate the sentence input vector, train the encoder, calculate the similarity and classify the sentences are given as follows.

2.1 Generation of Sentence Input Vectors

As shown in Fig. 1, the sentences are first converted to sentence vectors. In this model, the input sentence vectors are constructed based on 3500 commonly used Chinese characters in the first level character list of the general standard Chinese character list issued by the Chinese language and character working committee in 2013. The construction method is as follows: first, the input sentence is divided by the punctuation as a separator, and the special characters are deleted to reduce the noise in the sentence. Then, 3500 common Chinese characters are sequentially numbered in the first level character table, and the input sentences are represented by $S = (id_1, id_2, ...id_i, ...id_{512})$, where id_i represents the location of the i_{th} word of the input sentences in the first level character table. When the vector length is greater than the length of the input sentence, "0" will be padded to the end of the sequence, i.e., $id = 0$.

2.2 Offline Training

Since the minimum number of layers of the BERT model is 12, the encoder would need a large amount of training data to use the extracted features to represent the text data. To avoid the data overfitting problem of the large-scale neural network, we adopt four ways to prevent overfitting: building precoding model, fine-tuning, data growth, and adding dropout layer.

Specifically, the training set of the encoder could be divided into two parts: Chinese entry interpretation of Wikipedia, and the train set and verification set in LCQMC.

Figure 2 illustrates the training procedure of the encoder. We propose to concatenate the encoder and the DNN serially to extract the semantic features for

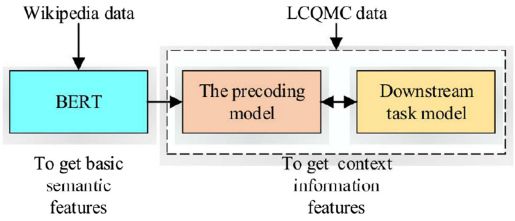

Fig. 2. Offline training of the similarity calculator.

further context information feature extractions. Firstly, we utilize the character interpretation from the Chinese version of Wikipedia to train the encoder to obtain the semantic features. Then we use the precoding model to learn the extracted semantic features. In the downstream model, the data of train set and verification set in LCQMC, as well as the data of train set and the verification set in LCQMC data set, are used to extract the context information features of the text to enable the vectors to extract the specific semantic features.

Unlike the traditional BERT model, whose output is a token length encoding, i.e., the encoding result of the first identifier $[CLS]$ of the sentence, we propose modifying the output of the BERT model to make the encoder output the coding of each token. In our design, the input of the downstream model is transformed into a 512×768 dimensional text semantic vector matrix. Thus the semantic table of the sentences becomes more refined. Then, the encoder outputs the coding results of each token of input sentences 1 and 2 to the LSTM module and performs zero paddings on the coding part that exceeds the input sentence's length.

2.3 Sentences Similarity Calculation and Classification Model

Figure 3 illustrates the intelligent DNN-based downstream model. Considering that the neural network layer number of the BERT encoder is 12, to avoid the overfitting problem of the large-scale model, here, the DNN consists of two LSTM modules, a concat layer, an MLP layer, a dropout layer, and a softmax layer. The details about how to calculate the similarity in each layer are given as follows.

Semantic Generation Layer. In this model, the LSTM module reads the output of the encoder sequentially. It generates the full semantic information on the premise of retaining the input text information by using the memory network characteristics of the LSTM. The parameters of the LSTM module used in the scheme are as follows: $BatchSize$ is 64, $NSteps$ is 512, $Inputdimension$ is 768, $Hiddendimension$ is 768, $Outputdimension$ is 768.

The coding results of the first token into the LSTM are regarded as the initial memory state C_0, and then the LSTM reads the remaining token coding results in turn as the input at a time. Namely, the coding result of the input sentence is

expressed as $(C_0, X_1, X_2, ...X_t, ...X_{511})$, where X_t represents the input at time t. X_t would pass through the forget gate in the LSTM module to determine the importance of the last moment's memory state and whether it needs to forget part of the content.

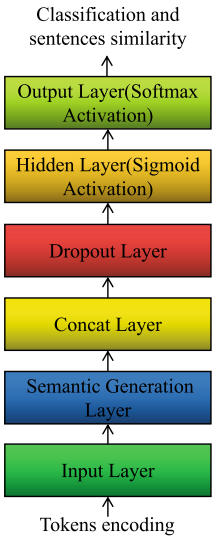

Fig. 3. The intelligent DNN-based downstream model.

At time t, the expression in the forget gate is presented as:

$$f_t = sigmoid(W_f * [h_{t-1}, X_t] + b_f) \tag{1}$$

where W_f is the weight matrix of the forget gate, $*$ is the matrix multiplication operation, h_{t-1} is the output state of the previous time slot, b_f is the offset coefficient of the forget gate. In addition, the sigmoid function is given by:

$$sigmoid(x) = \frac{1}{1 - e^{-x}}. \tag{2}$$

Next, X_t passes through the input gate to determine the importance of the X_t in order to determine the extent to which the current input needs to be updated into C_{t-1}, The expression of updating the coefficient i_t is given by:

$$i_t = sigmoid(W_i * [h_{t-1}, X_t] + b_i) \tag{3}$$

where W_i is the weight matrix of the input gate, b_i is the offset coefficient of the input gate. In addition, the memory state C_t^* of the input gate can be expressed as:

$$C_t^* = tanh(W_c * [h_{t-1}, X_t] + b_c) \tag{4}$$

where W_c is the weight matrix to calculate the update memory state, b_c is the offset coefficient to calculate the update memory state, and the function $tanh(\cdot)$ is given by:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{5}$$

It is evident that the greater the value of i_t, the higher the update degree of C_{t-1}.

Finally, with the outputs from the forget gate and the input gate, the memory state C_t and output state h_t at time t can be calculated jointly by:

$$C_t = f_t * C_{t-1} + i_t * C_t^* \tag{6}$$

$$o_t = sigmoid(W_o * [h_{t-1}, X_t] + b_o) \tag{7}$$

where W_o is the weight matrix of the output gate, b_o is the offset coefficient of the output gate, o_t is the output weight coefficient, and h_t is

$$.h_t = o_t * tanh(C_t). \tag{8}$$

Concat, Dropout and MLP Layer. After the encoding results of the two input sentences are generated by the LSTM module, the semantic expressions of the two input texts are spliced to form a 1×1536 dimensional splicing vector as the inputs to the MLP and dropout layers, where the dropout rate is 0.1.

The process of splicing vectors in the MLP layer is carried out as follows:

1. With the aid of the weight matrix in the MLP layer, the dimension degree of splicing vectors is reduced to 1×768 to complete the transfer from the input layer to the hidden layer.
2. After the vector of the hidden layer is processed by the activation function (sigmoid function), the dimension degree drops to 1×2. The transfer from the hidden layer to the output layer is completed, and the classification probability vector between input sentences is obtained.

Output Layer. Last, the output layer will process the resultant data from the MLP and dropout layers by using the Softmax function to obtain the normalized probability vector. In the normalized probability vector, the second element represents the similarity between the input sentences, and the category represented by the maximum value of the elements is the classification result. Additionally, the Softmax function is defined as follows:

$$P(S_i) = e^{g_i} / \sum_{k}^{n} e^{g_k} \tag{9}$$

where i is the category of the text classification, g_i represents 0 or 1.

3 Performance Analysis

In this section, we will evaluate the performance of the proposed scheme with the LCQMC data set and compare the accuracy, recall, precision, F1 score and cross-entropy loss performances of the proposed design with the counterpart systems. In the performance analysis, the learning rate is set as 10^{-5}, the epoch is 2, the batch size is 64, the amount of data in the train set is 238766, the amount of data in the verification set is 8802, and the amount of data in the test set is 12500.

3.1 Definitions

To evaluate the performance of the proposed design, we provide the following definitions.

Table 1. Classification of similarity evaluation precisions

	Predicted label(class 0)	Predicted label(class 1)
Ture label(class 0)	True Positive	False Negative
Ture label(calss 1)	False Positive	True Negative

Accuracy. Accuracy reflects the classification performances, and we define it as the correctness of the classification as below:

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{10}$$

where TP represents the true positive (the predicted label of the sample is class 0, and the true label is class 0), TN represents the true negative, FP represents the false positive, FN represents the false negative.

Recall. Recall reflects the ability of the model to recall target categories in the research field. We define it as:

$$Recall = \frac{TP}{TP + FN}. \tag{11}$$

Precision. Precision reflects the ability of the model to precisely capture the target categories in the research field:

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

F1-Score. F1-score reflects the comprehensive ability of the model:

$$F1 - score = \frac{Recall \times Precision \times 2}{Recall + Precision}. \tag{13}$$

Loss. Loss reflects the deviation degree between the overall predicted value and the overall real value of the model:

$$Loss = \frac{1}{N}(\sum_{0}^{N}(-[y_i \times log(p_i) + (1 - y_i) \times log(1 - p_i)])) \tag{14}$$

where N is the total number of samples, y_i is the label of the samples, p_i is the text-similarity between samples.

Last but not least, to analyze the accuracy, recall, precision, and F1-score, in Table 1, we apply four classification samples for similarity computations based on the real classification results and the prediction classification results of test samples.

3.2 Similarity Evaluation Performance Analysis

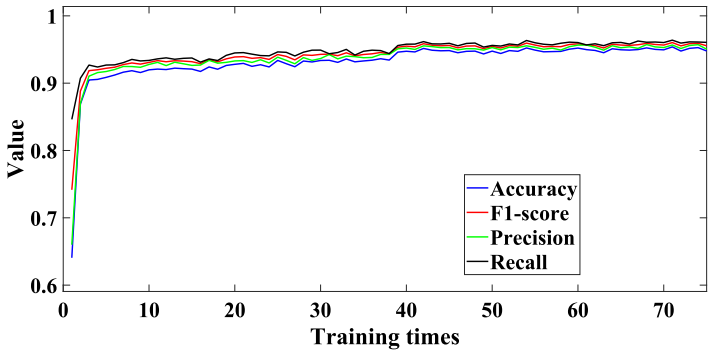

Fig. 4. The similarity evaluation performances versus the training times with the training set.

Similarity Computation Precision. First of all, we investigate the similarity evaluation performances versus the training times when applying the training set at the online deployment stage. As illustrated in Fig. 4, with the increase of training times, the accuracy, F1 score, accuracy, and recall of the scheme are improved. When the training times reach 3, the scheme's four performance indexes are all enhanced to be larger than 0.9. Moreover, it is clear that when the training times are larger than 42, little performance enhancement could be observed, and the similarity evaluation precision approaches 1.

Next, we investigate the loss versus the training times when applying the training set. As illustrated in Fig. 5, with the increase of the training times, the loss decreases rapidly. It is noticeable that when the training times reach 3, the loss drops to 50. Moreover, it is evident that the loss decreases and approaches 0 when the training times are larger than 42.

Robustness Analysis. Subsequently, to evaluate the robustness, we analyze the similarity computation performances when applying the verification set. As illustrated in Fig. 6, with the increase of the training times, the accuracy, F1 score, and precision all increase rapidly. However, the recall fluctuates when the verification set is used at the online deployment stage. When the training times reach 3, the four performances are around 0.84. When the training times are greater than 40, the four performances remain approaching 0.9. The reason is that the LCQMC verification set is aggregated data set, wherein the data with the same label category are not scattered.

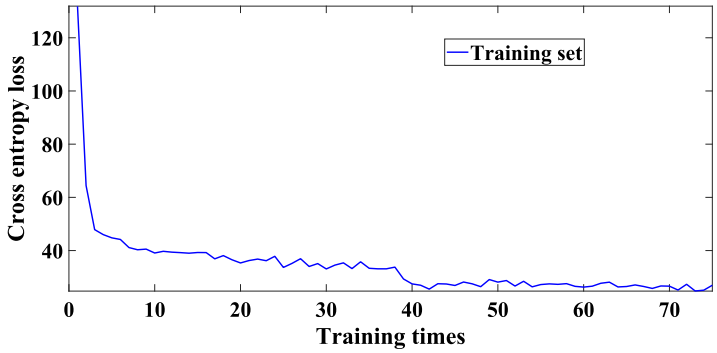

Fig. 5. The loss of the scheme versus the training times with the training set.

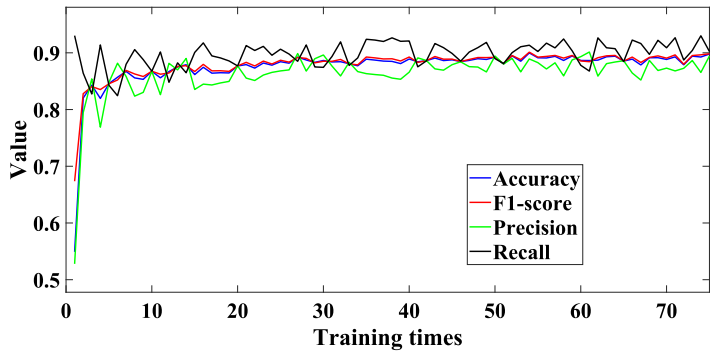

Fig. 6. The similarity evaluation performances versus the training times with the verification set.

We could also notice from the figure that the four performance indicators of the scheme have relatively high values with the verification set, and it converges rapidly. The reason is that the input sentences are processed by the encoder and the LSTM module; hence the obtained sentence semantic expression has higher accuracy than that of the traditional scheme. This also leads to a higher sentence similarity calculation accuracy than that of the traditional scheme. Meanwhile, we could also see that the construction of the siamese neural network architecture accelerates the scheme's convergence speed.

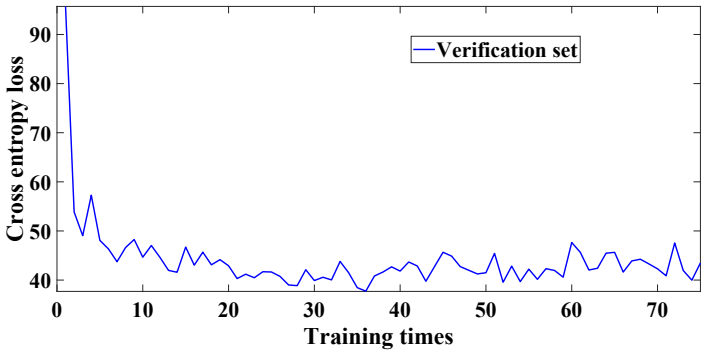

Fig. 7. The loss of the scheme versus the training times with the training set.

Subsequently, Fig. 7 investigates the loss of the scheme with the increase of training times when applying the verification set. It is clear from the figure that the cross-entropy loss decreases rapidly. When the training times are larger than 37, the loss of the scheme approaches 0. Furthermore, thanks to the high similarity evaluation accuracy, the classification precision of the proposed design is also very high, and the classification results of the scheme deviate only slightly from the real one.

Performance Comparisons. Finally, we compare the similarity evaluation accuracy of the proposed design with counterpart schemes. It could be observed from Table 2, the accuracy, F1 score, and precision of the proposed scheme are higher than those of the counterpart schemes. According to the rules of 2018 ant financial's NLP competition, we select the accuracy and the F1 score as the performance indicators for the classification models. It is evident that the proposed scheme's similarity evaluation performances are better than those of other schemes. Meanwhile, the accuracy of the proposed scheme with the LCQMC verification set and the test set are almost identical, demonstrating that our proposed scheme can effectively alleviate the problem of the inconsistent accuracy of the neural network model in different fields.

Table 2. Scheme performance comparisons with LCQMC data set

Scheme	Validation set's accuracy	Test set's accuracy	Test set's F1 csore	Test set's precision	Test set's recall
WMD [7]	*	0.706	0.734	0.67	0.812
C_WO [7]	*	0.707	0.706	0.611	0.887
S_COS [7]	*	0.703	0.716	0.601	0.889
CBOW [7]	*	0.737	0.774	0.679	0.899
CNN [7]	*	0.728	0.757	0.684	0.846
BILSTM [7]	*	0.761	0.789	0.706	0.833
BIMPM [7]	*	0.834	0.85	0.776	0.939
DSSM [2]	*	0.6334	*	*	*
ABCNN [2]	*	0.7992	*	*	*
ESIM [2]	*	0.818	*	*	*
DIIN [2]	*	0.8447	*	*	*
NEZHA [11]	0.8964	0.8710	*	*	*
CHARTEST [6]	0.8443	0.8618	*	*	*
ERNIE [9]	0.897	0.874	*	*	*
Proposed scheme	0.8761	0.8843	0.8890	0.8549	0.9258

4 Conclusions

In this paper, we propose to serially concatenate the encoder and the DNN to compute the similarity of Chinese sentences for enhanced performances jointly. In our proposed scheme, we use the BERT to extract the semantic feature for further context information feature extraction with the DNN. We construct the DNN, which is composed of the input layer, the semantic generation layer, the concat layer, the dropout layer, the hidden layer, and the output layer. At the offline training stage, the Wikipedia data and the LCQMC data are used as the data set to configure the neural networks. Subsequently, we evaluate the similarity of Chinese sentences at the online deployment stage with the obtained parameters and structure. Furthermore, we investigate the accuracy, precision, F1 score, recall, and cross-entropy loss of the proposed design to evaluate its robustness. The results demonstrate that by applying the serially concatenated BERT and LSTM aided DNN design, the similarity computation performances could be improved. Therefore, with higher similarity evaluation accuracy, the proposed scheme could be integrated with IoT devices to provide better NLP services such as searching, data mining, or text classifications.

Acknowledgements. This work was supported in part by the State Key Program of National Social Science of China (No. 18AZD035), the Key Research & Development and Transformation Plan of Science and Technology Program for Tibet Autonomous Region (No. XZ201901-GB-16), the Special Fund from the Central Finance to Support the Development of Local Universities (No. ZFYJY201902001) and the National Natural Science Foundation of China (No. 71964030).

References

1. Arora, S., Liang, Y., Ma, T.: A simple but tough-to-beat baseline for sentence embeddings. In: 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings, Toulon, France (2019)

2. Bshoter, J.: Text-matching. https://github.com/BshoterJ/Text-Matching
3. Gunes, E., Dragomir, R.: LexRank: graph-based lexical centrality as salience in text summarization. J. Artif. Intell. Res. **22**(1), 457–479 (2011)
4. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1, Minneapolis, Minnesota. Association for Computational Linguistics, Germany, January 2019
5. Ko, Y., Park, J., Seo, J.: Automatic text categorization using the importance of sentences. In: Proceedings of the 19th International Conference on Computational Linguistics, COLING'02, USA, vol. 1, pp. 1–7. Association for Computational Linguistics (2002)
6. Li, X., Meng, Y., Sun, X., Han, Q., Yuan, A., Li, J.: Is word segmentation necessary for deep learning of Chinese representations? In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, pp. 3242–3252. Association for Computational Linguistics, July 2019
7. Liu, X., et al.: LCQMC: a large-scale Chinese question matching corpus. In: Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, New Mexico, USA. Association for Computational Linguistics, Germany, July 2018
8. Sepp, H., Jürgen, S.: Long short-term memory. Neural Comput. **9**, 1735–1780 (1997)
9. Sun, Y., et al.: Ernie: Enhanced representation through knowledge integration (2019). https://arxiv.org/abs/1904.09223v1
10. Varelas, G., Voutsakis, E., Raftopoulou, P., Petrakis, E.G., Milios, E.E.: Semantic similarity methods in wordnet and their application to information retrieval on the web. In: Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management, WIDM'05, Bremen, Germany, pp. 10–16. Association for Computing Machinery, New York (2005)
11. Wei, J., et al.: Nezha: Neural contextualized representation for Chinese language understanding (2019). https://arxiv.org/abs/1909.00204
12. Zhao, Q., Qi, J.: A method for calculating the similarity of short texts based on semantic and syntactic structure. Comput. Eng. Sci. **40**(283), 145–152 (2018)