Journal of Artificial Intelligence Practice (2024)
Clausius Scientific Press, Canada

DOI: 10.23977/jaip.2024.070315
ISSN 2371-8412 Vol. 7 Num. 3

Siamese Network-Based Text Similarity Algorithm Research

Junhong Chen[1,2,a,*], Kaihui Peng[3,b]

[1]*School of Software Engineering, South China University of Technology, Guangzhou, China*
[2]*LeiHuo Studio, NetEase, Hangzhou, China*
[3]*Faculty of Business and Economics, University of Malaya, Kuala Lumpur, Malaysia*
[a]*jupyterchen@163.com*, [b]*pengkaihui66@163.com*
[*]*Corresponding author*

Keywords: Text similarity calculation; Attention mechanism; Pre-trained model

Abstract: This paper proposes a text similarity calculation model based on multi-scale convolutional neural networks and attention mechanisms. The model is capable of extracting information at different granularities within the text, enabling it to learn from multiple layers of information and thereby improving the accuracy of the text similarity calculation task. After training, the model can generate sentence vectors suitable for cosine similarity computation, which allows the model to pre-generate vectors for the text in the repository. During actual retrieval, only the sentence vector of the text to be searched is needed to calculate similarity with the pre-generated vectors in the repository.

1. Introduction

Text matching is a key technique in information retrieval tasks. Models like word2vec[1] and glove[2] are traditional and classic text similarity calculation models. However, traditional text similarity calculation models have poor text representation capabilities and are difficult to resolve polysemy issues. Therefore, the research on pre-trained models has become a popular topic in recent years. This paper proposes a text similarity calculation model based on multi-scale convolutional neural networks and attention mechanisms (CNN-Attention-Sentence-Bert, CNN-ATT-Sbert). The model fully extracts information at different granularities within the text, enabling it to learn from multiple levels of information, thus improving the precision of text similarity recognition. Unlike the traditional static text semantic computation methods such as word2vec, TF-IDF[3], etc., this paper uses Bert-wwm[4] to extract text semantics, which can extract deep semantic features, generate dynamic word vectors, and solve the problem of static word vectors. To further improve the calculation accuracy, this paper adopts the Siamese network architecture based on Bert-wwm, combines it with multi-scale convolutional neural networks and attention mechanisms, and utilizes and refines information at different granularities within the text to fully mine key information, thereby enhancing the accuracy of the text similarity calculation task. This method has both high accuracy and extremely high runtime speed.

2. Design of the Text Similarity Algorithm Based on Siamese Networks

2.1 Algorithm Overview

The Siamese network architecture consists of two coupled artificial neural networks[5]. The Siamese network uses two subnetworks to obtain inputs from two samples, with each subnetwork receiving inputs and producing high-dimensional spatial representations. Finally, a certain metric method is used, such as cosine similarity calculation, to compare the similarity of the two samples. Siamese networks usually have a three-layer structure: the first is the input layer, which is responsible for converting words into word vectors and feeding the converted word vectors into subsequent encoding layers. The second is the encoding layer, which is responsible for encoding the input word vectors to obtain sentence vectors that represent the entire sentence. The last is the similarity measurement layer, which is responsible for judging the similarity relationship between two sentence vectors and the metric method can be some simple vector distance calculations, such as cosine similarity calculation. It can also combine the two vectors and use some classifiers or multilayer perceptrons to represent the direct similarity relationship between the two vectors. The two subnetworks of the inputs are not only structurally identical but also share weights. The structure of the Siamese network is shown in Figure 1.

This paper builds upon the Siamese network architecture by adding multi-scale convolutional neural networks[6] and attention mechanisms[7], proposing a text similarity calculation model called CNN-ATT-Sbert based on these components. This model effectively refines information at various granularities within text, allowing it to learn from multiple hierarchical levels and thoroughly extract key information in the text, thereby enhancing the accuracy of the text similarity calculation task. The input layer utilizes parameter-sharing Bert-wwm, which offers improved Chinese support. Since cosine similarity is used for text similarity calculations during training, the word vectors generated upon completion of the model are well-suited for distance measurement using the cosine similarity method[8]. As the two sentences are encoded independently without depending on each other's information, the model can function as a standalone sentence encoder after training, making offline vector generation feasible. In practical retrieval tasks, the model can be used to pre-generate the sentence vectors for the text in the database. During retrieval, it is only necessary to generate the sentence vector for the text being searched once, which can then be compared with pre-generated vectors, significantly speeding up the retrieval process. This characteristic makes the Siamese network architecture-based model particularly suitable for the coarse retrieval phase of text retrieval, which involves quickly finding the top N most similar text. The model structure is illustrated in Figure 2.

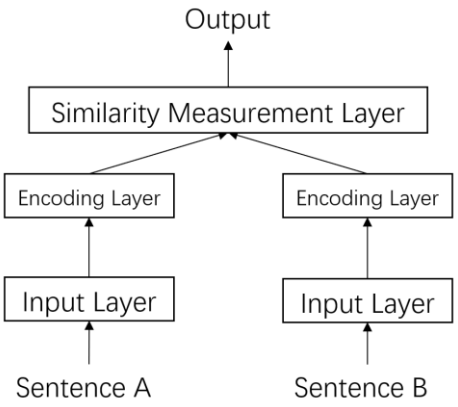

Figure 1: Siamese Network Architecture

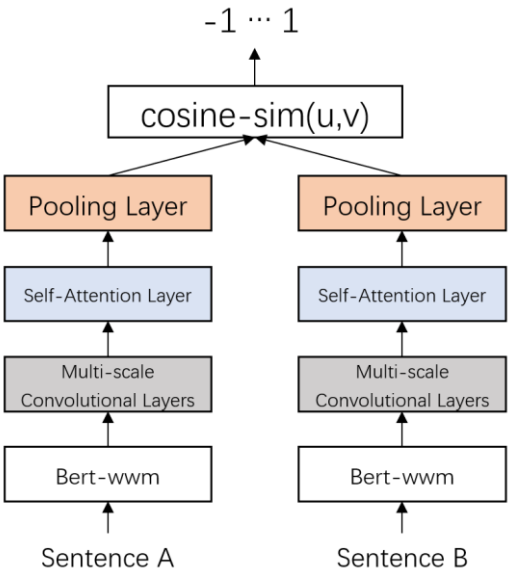

Figure 2: CNN-ATT-SBert Model Structure

2.2 Input Layer

The input layer also employs the standard Bert-wwm input approach, which involves segmenting the sentence by characters, adding the [CLS] marker at the beginning of the sentence, and the [SEP] marker at the end. These are denoted as S_a and S_b respectively, and they are input into Bert-wwm for sentence semantic capture. The following encoded vectors are obtained.

$$h_a = \{h_{a_0}, h_{a_1}, \cdots, h_{a_{l-1}}\} = \text{BERT} - \text{WWM}(S_a) \quad (1)$$

$$h_b = \{h_{b_0}, h_{b_1}, \cdots, h_{b_{l-1}}\} = \text{BERT} - \text{WWM}(S_b) \quad (2)$$

Among them, $h \in R^{l \times d}$, where l is the length of the sequence and d is the dimension of the word vectors produced by Bert-wwm, typically 768. h_{a_i} represents the semantic representation of the i-th character in the a sequence, and h_{b_i} represents the semantic representation of the i -th character in the b sequence. After encoding the sentences, the next layer is a special CNN structure. CNN can be used to capture the multi-grained information of the text.

2.3 Multi-scale Convolutional Layers

Due to the characteristics of CNN convolutional kernels[9], they can capture feature information at the word level, such as keywords and phrases. By combining three different sizes of convolutional kernels to create a hierarchical structure, the model can capture more spatial and hierarchical information from the initial text, thereby improving the accuracy of text matching. In designing this CNN layer structure, the design concept of “Network in Network” was referenced, which facilitates easier extraction of text features. As shown in Figure 3.

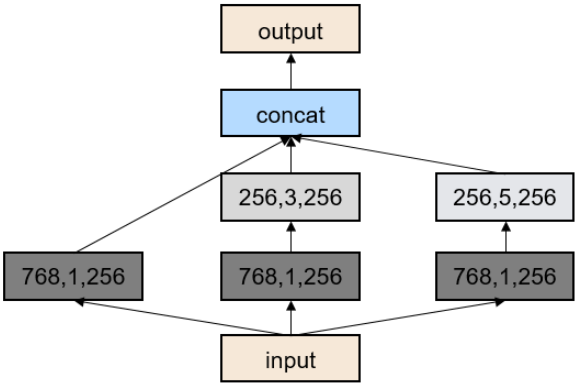

Figure 3: Multi-scale Convolutional Layer Structure

Among these, the first number represents the dimension of the word vectors. Since the word vector dimension of Bert-wwm is 768, the input dimension of the convolutional kernel is also 768. The middle number represents the width of the convolutional kernel, where the text uses 1, 3, and 5. The last number is the number of channels of the convolutional kernel, and in this text, it is set to 256. Due to the characteristics of CNN convolutional kernels, they can capture feature information at the word level, such as keywords and phrases. By combining three different sizes of convolutional kernels, the model can capture more spatial and hierarchical information from the initial text, thereby enhancing the ability to identify such features. Let O_a and O_b denote the final outputs of sentences a and b after passing through the CNN layer, respectively. The overall effect of the multi-scale convolutional layer is as follows.

$$O_a = \text{ImprovedCNN}(h_a) \tag{3}$$

$$O_b = \text{ImprovedCNN}(h_b) \tag{4}$$

The detailed calculation steps for the multi-scale convolutional layer are as follows. For the CNN layer, where the size of the convolutional kernel is 1, let p_1, p_3, and p_5 represent the outputs of the three convolutional kernels, respectively. Their calculation process is as follows.

$$p_{1_i} = f\left(W_{p,1} \cdot h_i + \text{bias}\right) \tag{5}$$

$$p_{3_i} = f\left(W_{p,3} \cdot h_i + \text{bias}\right) \tag{6}$$

$$p_{5_i} = f\left(W_{p,5} \cdot h_i + \text{bias}\right) \tag{7}$$

Among them, W represents the parameters of the respective convolutional kernels, and h is the input text. For the CNN layer where the size of the convolutional kernel is 3, let q denote the output of this particular convolutional kernel. The input to this kernel is the output of the previous layer's convolutional kernels. The calculation process for this convolutional kernel is as follows. For edge values, padding operations are used.

$$q_i = f\left(W_q \cdot p_{3_{i-1:i+1}} + \text{bias}\right) \tag{8}$$

And for the CNN calculation unit with a kernel size of 5, let v denote the output of this convolutional kernel. The calculation process for this particular kernel is as follows, with padding operations also applied to handle edge values.

$$v_i = f\left(W_v \cdot p_{5_{i-2:i+2}} + \text{bias}\right) \tag{9}$$

In this context, W represents the parameters of the convolutional kernel. In the above formulas, each f denotes a nonlinear activation function. In the present work, the Tanh activation function is

used. At the end of the CNN layer, the outputs of all the CNN units are concatenated to obtain the final output of the CNN layer. The calculation is as follows.

$$O_i = p_{1_i} \oplus q_i \oplus v_i \tag{10}$$

The symbol $\oplus$ denotes the concatenation operation of vectors. After passing through the multi-scale convolutional layer, each sentence extracts features of different granularity, forming new vectors. After forming these new vectors, they need to be processed by the self-attention layer.

2.4 Self-Attention Layer

Essentially, the attention mechanism filters out non-essential information from a wealth of data, focusing only on the key information, thereby reducing the weight of unimportant data[10]. By using the self-attention layer, key information can be given more emphasis, while reducing the impact of non-key information. Figure 4 illustrates a diagram of the relationship in the calculation process of the attention mechanism. Compared to CNN and RNN, the advantage of the attention mechanism is that it has fewer parameters, making the computational speed faster. Moreover, attention mechanisms support parallel computation, which further enhances computational speed, and are superior to sequential models in many natural language processing tasks.

This section uses the self-attention mechanism, with the calculation following formula (11).

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{11}$$

In this case, Q, K and V are matrices obtained from the input vectors through linear transformations, as calculated below.

$$Q = O \cdot W_q \tag{12}$$

$$K = O \cdot W_k \tag{13}$$

$$V = O \cdot W_v \tag{14}$$

Where W_q, W_k, and W_v are the parameters for linear transformations that can be trained during the model training process. These parameters enhance the model's fitting ability and act as a buffering effect. The term $\sqrt{d_k}$ in formula (11) represents the square root of the dimensionality of the word vectors. Dividing by $\sqrt{d_k}$ is intended for gradient stability.

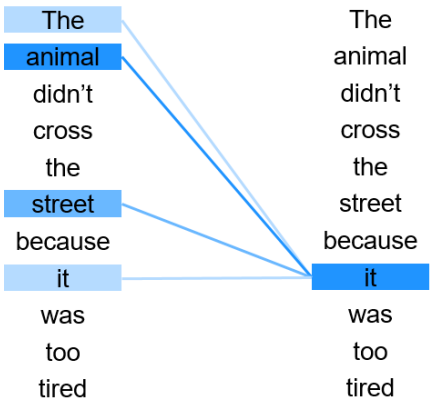

Figure 4: Diagram of Relationships in the Calculation of the Self-Attention Mechanism

2.5 Similarity Measurement Layer

After the extraction of multi-grained information through the multi-scale convolutional network layer and the enhancement of key information via the attention mechanism layer, sentences a and b have been enriched with semantic content. The next step is to input them into the similarity measurement layer to calculate their similarity. Prior to the similarity calculation, a pooling operation is required to be performed through a pooling layer[11], resulting in a sentence vector that represents the entire sentence. The pooling layer employs average pooling, and the calculation process is as follows.

$$Z_{a,avg} = \sum_{i=1}^{l_a} \frac{T_{a_i}}{l_a} \tag{15}$$

$$Z_{b,avg} = \sum_{j=1}^{l_b} \frac{T_{b_j}}{l_b} \tag{16}$$

Here, T_a and T_b represent the outputs of the attention layer, while $Z_{a,avg}$ and $Z_{b,avg}$ are the final vectors outputted from the pooling layer. Subsequently, the cosine similarity distance is calculated for the vectors Z_a and Z_b using the cosine similarity calculation method, as detailed in formula (17).

$$d = \frac{\sum_{i=1}^{n}\left(Z_{a_i} \cdot Z_{b_i}\right)}{\sqrt{\sum_{i=1}^{n}(Z_{a_i})^2} \cdot \sqrt{\sum_{i=1}^{n}(Z_{b_i})^2}} \tag{17}$$

The loss function of the model is the Mean Squared Error loss function, denoted as MSELoss, as shown in formula (18).

$$loss = \frac{\sum_{i=1}^{n}(d_i - y_i)^2}{n} \tag{18}$$

In this context, d_i represents the cosine similarity distance for the i-th pair of training samples. y_i takes a value of -1 or 1, which is the actual label for that pair of samples.

3. Experimental Results and Analysis

3.1 Experimental Dataset

The experiments in this chapter use the LCQMC dataset for testing[12]. The LCQMC dataset is a Chinese semantic matching dataset, with a training set consisting of 238,766 data items. LCQMC places more emphasis on semantic matching rather than literal matching and has been widely applied in the task of Chinese text similarity calculation. Specific information is presented in Table 1.

Table 1: Data Distribution of the LCQMC Dataset

	train	dev	test	all
PAIRS	238766	8802	12500	260068
SENS	245951	15917	23557	256433
POS	138574	4402	6250	149226
NEG	100192	4400	6250	110842
LEN	6.12	7	5.65	6.26

In addition, the BQ Corpus dataset was also employed [13]. The BQ Corpus is a credit text similarity matching dataset, where a value of 0 represents dissimilarity and a value of 1 represents similarity.

3.2 Baseline Models

The baseline models compared in this experiment include the classic and commonly used word vector models: Word2vec, FastText[14], and GloVe. Additionally, the performance of using vectors generated by Bert-wwm directly was compared, which includes the use of the CLS token and the average value. Lastly, a comparison was made with a pre-trained model, Sentence-Bert.

3.3 Evaluation Metrics

The evaluation metrics employed are the Pearson correlation coefficient and the Spearman rank correlation coefficient. Correlation coefficients are statistical measures used to reflect the degree of correlation between two variables. These metrics range from [-1, 1], with values greater than 0 indicating a positive correlation, values less than 0 indicating a negative correlation, and a value of 0 indicating no correlation. The closer the absolute value is to 1, the higher the degree of positive or negative correlation. Both the Pearson and Spearman coefficients are commonly used types of correlation coefficients. When evaluating the model results, sentence vectors are first generated for sentences using each model, then the sentence vectors of sentence pairs are calculated using cosine similarity, and the correlation coefficient is computed by comparing these results with the true labels.

3.4 Analysis of Experimental Results

For Word2vec, GloVe, and other static word vector models, the method to obtain a sentence vector is to first tokenize the text, then generate a word vector for each word, and finally average these word vectors to obtain the sentence vector. In the experiment, to achieve higher-quality vectors, in addition to using word segmentation tools to tokenize the articles, stopwords in the text were also removed to avoid their impact on semantics. For Bert-wwm, no word segmentation is performed; the entire sentence is processed through standard input and fed into Bert-wwm. There are two ways to obtain a sentence vector from Bert-wwm: one is to directly use the vector corresponding to the [CLS] token to represent the whole vector, denoted as Bert-wwm[CLS]. The other is to average the output of Bert-wwm to obtain a representation of the entire vector, denoted as Bert-wwm avg. For Siamese network-based methods such as Sentence-Bert and CNN-ATT-Sbert, after the model is trained, each sentence's vector is generated using the model respective to it. Table 2 shows the results of the models on the LCQMC dataset, and Table 3 shows the results of the models on the BQ Corpus dataset.

Table 2: Results of Various Models on the LCQMC Dataset

	Pearson	Spearman
Word2vec	0.40695	0.47952
Glove	0.27835	0.46207
FastText	0.43622	0.49265
Bert-wwm avg	0.54452	0.58520
Bert-wwm [CLS]	0.27300	0.40637
Sentence-Bert	0.68815	0.69183
CNN-ATT-SBert	0.69096	0.69690

The tables indicate that both pre-trained methods have a high accuracy, and the word vectors they generate for computing text similarity result in a higher correlation with the original labels. Despite Sentence-Bert's relatively simple model structure, which does not employs attention mechanisms or multi-scale convolutional network layers compared to CNN-ATT-Sbert, it achieved very good results. This suggests that using a Siamese network architecture and guiding the model with supervised data can produce higher-quality vectors. Moreover, the models that integrate multi-scale convolutional

neural networks and attention mechanisms can further enhance performance, indicating that this structural approach has a positive effect on deep semantic extraction from sentences.

Table 3: Results of Various Models on the BQ Corpus Dataset

	Pearson	Spearman
Word2vec	0.31288	0.31373
Glove	0.26531	0.29350
FastText	0.29177	0.29877
Bert-wwm avg	0.37725	0.38688
Bert-wwm [CLS]	0.14498	0.21384
Sentence-Bert	0.63763	0.64208
CNN-ATT-SBert	0.65209	0.65361

Comparing the vectors obtained from the average method and those directly using the [CLS] token from BERT, it can be seen that the [CLS] token vector can to some extent represent the entire sentence, but its effect is not as good as the vector obtained from the averaging method. The vector obtained by averaging is more suitable for calculating text similarity. Therefore, in text similarity calculation, it is more widespread to use the full output of Bert-wwm instead of calculating using the [CLS] token.

In contrast to classical models such as Word2vec and GloVe, the series of models based on Siamese networks for pre-training have significantly outperformed the classical models. The reason for this result is the guidance from the labels. In actual text, there are often cases where adding or removing a single word can change the entire meaning of a sentence, such as negative words. Classical models find it difficult to significantly alter the sentence vector of an entire sentence due to the change of a single word, which can lead to failure in identifying sentence pairs with different meanings due to individual word differences. On the other hand, the pre-trained models based on Siamese networks, with the guidance of labels, can perform targeted training for specific cases, thus achieving better results than unsupervised models.

4. Conclusions

This paper first introduced the steps of the CNN-ATT-SBert algorithm for text similarity calculation, including the input of text, the specific implementation and steps of the multi-scale convolutional neural network layer, and the attention mechanism layer. Secondly, it presented the experimental environment and the related datasets for the text similarity calculation experiment, as well as the evaluation criteria for the experimental results. Then, through related experiments and comparisons with common baseline methods, the accuracy of the proposed Siamese-network-based text similarity calculation algorithm in this paper was validated.

References

[1] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space [J]. arXiv preprint arXiv:1301.3781, 2013.

[2] Ramos J. Using tf-idf to determine word relevance in document queries[C]//Proceedings of the first instructional conference on machine learning. 2003, 242(1): 29-48.

[3] Cui Y, Che W, Liu T, et al. Pre-training with whole word masking for chinese bert[J]. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2021, 29: 3504-3514.

[4] Koch G, Zemel R, Salakhutdinov R. Siamese neural networks for one-shot image recognition[C]//ICML deep learning workshop. 2015, 2(1): 1-30.

[5] Cai Z, Fan Q, Feris R S, et al. A unified multi-scale deep convolutional neural network for fast object detection[C]//Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14. Springer International Publishing, 2016: 354-370.

[6] Vaswani A. Attention is all you need [J]. Advances in Neural Information Processing Systems, 2017.

[7] Huang A. Similarity measures for text document clustering[C]//Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand. 2008, 4: 9-56.
[8] Mairal J, Koniusz P, Harchaoui Z, et al. Convolutional kernel networks[J]. Advances in neural information processing systems, 2014, 27.
[9] Niu Z, Zhong G, Yu H. A review on the attention mechanism of deep learning[J]. Neurocomputing, 2021, 452: 48-62.
[10] Yu D, Wang H, Chen P, et al. Mixed pooling for convolutional neural networks[C]//Rough Sets and Knowledge Technology: 9th International Conference, RSKT 2014, Shanghai, China, October 24-26, 2014, Proceedings 9. Springer International Publishing, 2014: 364-375.
[11] Liu X, Chen Q, Deng C, et al. Lcqmc: A large-scale chinese question matching corpus[C]//Proceedings of the 27th International Conference on Computational Linguistics. 2018: 1952-1962.
[12] Chen J, Chen Q, Liu X, et al. The bq corpus: A large-scale domain-specific chinese corpus for sentence semantic equivalence identification[C]//Proceedings of the 2018 conference on empirical methods in natural language processing. 2018: 4946-4951.
[13] Joulin A, Grave E, Bojanowski P, et al. Bag of tricks for efficient text classification[J]. arXiv preprint arXiv:1607.01759, 2016.
[14] Reimers N. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks[J]. arXiv preprint arXiv:1908.10084, 2019.