Bert-based Siamese Network for Semantic Similarity

Bert-based Siamese Network for Semantic Similarity

Xu Feifei[1,a], Zheng Shuting[1], Tian Yu[1]

[1]School of Computer Science and Technology Shanghai University of Electric Power Shanghai, China

[a]xufeifei@shiep.edu.cn

Abstract—As a key technology in the field of natural language processing, text matching is widely used in many tasks such as question answering systems, search, recommendation, and advertising. Aiming at the insufficient feature extraction capabilities of traditional Siamese-based text matching methods, a fast matching method based on large-scale pre-training model BSSM (BERT based Siamese Semantic Model) is proposed. The proposed method uses a pre-training model to encode two texts separately, which interact the representation vectors of the two texts to obtain attention weights and generate new representation vectors, so that the new and old representation vectors can be pooled and aggregated, and finally two representation vectors are concatenated in some strategy and sent to the prediction layer for prediction. Experimental results on the AFQMC and LCQMC datasets show that the proposed method not only improves the accuracy rate, but also improves the computational efficiency.

1. INTRODUCTION

In the process of natural language processing, text matching is an important basic problem, which generally plays a core supporting role in the application system in the form of text similarity calculation. It can be applied to various types of natural language processing tasks, such as web search, text deduplication, automatic question and answer, information flow recommendation and retelling questions. These common NLP tasks can be abstracted as text matching problems to a certain extent. For example, the question and answer system can be attributed to the matching of user questions and standard questions, that is, the business usually configures corresponding answers for some common standard questions. What needs to be done when a user asks a question is to calculate the similarity between the user question and the configured standard question. Then the standard question most similar to the user's question can be selected, and the corresponding answer to the user will be returned. In search, recommendation, advertising and other businesses, it can actually be divided into two stages: recall and sorting. Recall is a process of pulling candidate sets, which is often a matching problem.

Traditional text matching methods include BM25, TF-IDF (term frequency–inverse document frequency), VSM [1], SimHash [2] and so on. For example, the BM25 algorithm divides the input sentences, and calculates the relevance of each word between the sentence and the document, and then performs a weighted summation to obtain the relevance score between the sentence and the document. The higher the score, the better the match between the sentence and the document. These traditional methods mainly match at the vocabulary level, which can also be considered to be a similarity problem at the vocabulary level. In fact, the matching algorithm based on vocabulary coincidence has great limitations, such as the limitation of word meaning, that is, the same word in different contexts, but the semantic information expressed may be different. For example, the word

"apple" can mean either a kind of fruit or a mobile phone brand. In the same way, the same semantics can also be expressed by different words, for example, "Apple mobile phone" and "iphone" have the same meaning.

Therefore, for text matching tasks, we should not only focus on matching at the lexical level, but also at the semantic level. The main problem of semantic level matching is how to represent the semantics and how to calculate it. In the 1990s, the proposal of LSA [3] (Latent Semantic Analysis) provided a new idea for semantic analysis technology, which is to map sentences to a fixed-length low-dimensional vector space, and then perform similarity calculations on this vector space. The LSA model is based on the assumption that words with similar meanings have similar contexts. Therefore, LSA can solve the problem of synonyms, but it cannot solve the problem of ambiguity. Subsequent PLSA [4] (Probabilistic Latent Semantic Analysis), LDA [5] (Latent Dirichlet Allocation) and so on are also commonly used theme models. These techniques can be concisely represented semantics by low-dimensional vectors, and the calculation is convenient, which can make up for the deficiency of vocabulary matching methods, but can not replace them.

In recent years, deep learning has been successfully used in many fields, and the research on text matching problems has naturally shifted from traditional methods to deep text matching models. Using the word embedding obtained by training the neural network language model to perform text matching calculation, the training method is more concise, as well as the semantic computability of the word embedding representation obtained is further enhanced. There are usually two types of methods for obtaining word embedding: one is a local context window method, such as the CBOW and Skip-Gram methods proposed by Mikolov et al. [6] in 2013; the other is a method based on global matrix decomposition, which is named LSA mentioned above. However, these two methods have their own shortcomings. Among them, although LSA effectively uses statistical information, it is very poor in terms of vocabulary analogy, while CBOW and Skip-Gram can perform vocabulary analogy very well. These two methods are based on a local context window, so the global lexical co-occurrence statistics cannot be effectively used. In 2014, Pennington et al. [7] proposed a new GloVe method, which learned word embedding based on the statistical information of global vocabulary co-occurrence, thus combined the statistical information with the advantages of the local context window method. The effect was indeed improved, and the defects of global matrix decomposition and local context window were solved.

When using unlabeled text data to train the obtained word embedding, the actual effect is almost the same as using the topic model technology, because the essence is still training based on co-occurrence information. In addition, the word embedding itself does not solve the problem of semantic representation of words and sentences, and the semantics cannot be changed with the change of context.

In response to the above problems, this paper proposes a Siamese network semantic model based on BERT (Bidirectional Encoder Representation from Transformers). By dynamically adjusting word representation through BERT, it can effectively solve the problem of semantic representation of sentences, that is, the meaning of word will change with context.

2. Related Work

Different from the traditional feature-based matching methods, the text matching method in the deep learning era can be summarized into three types: representation model, interaction model and large-scale pre-training model.

Representation model: The text is converted into a representation vector through the presentation layer, so this method focuses more on the construction of the presentation layer. The Siamese structure is used to extract the semantics of the text separately and then match. The parameters of the two towers are shared, and networks such as MLP, CNN, RNN, Transformer, etc. can be used. For interactive calculation at the matching layer, methods such as dot product, cosine, Gaussian distance, MLP or similarity matrix can be used. The advantage is that shared parameters make the model smaller and easier to train. The representation model architecture is shown in Figure 1. Although the sentence vector of the short text is derived from the model and has certain semantic

information, the shortcomings cannot be ignored: There is no clear interaction between the two texts in the mapping process, where a lot of information about each other is lost.

One of the most famous models is Microsoft's 2013 DSSM [8] (Deep Structured Semantic Models), which has two innovations. One is to replace the word bag model with Tri-Gram's Word Hashing, which reduces the vocabulary. It uses a fully connected neural network, while its shortcomings cannot take the text order into account. In 2014, Microsoft designed CDSSM [9]. The main change is to replace MLP with a CNN model, which can extract N-gram features. In 2016, DSSM-LSTM [10] and MV-DSSM [11] were successively published.

The representation-based approach is simple and effective, but its shortcomings are also obvious. Representations are used to represent high-level semantic features of text, but high-level representations of features such as word relationships and syntax in the text are more difficult to capture, and it is difficult to determine whether a representation can well represent a piece of text. If you want to be able to model the matching relationship of each level of the text, it is best to make the text interact as soon as possible. In layman's terms, the earlier you know, the more likely the two texts will understand each other.

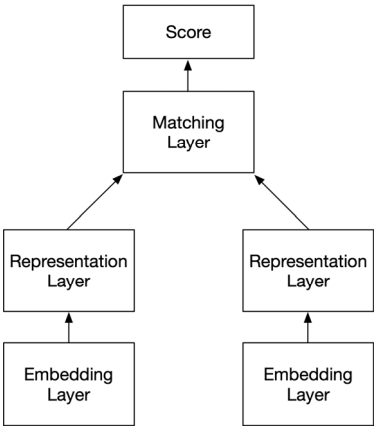

Fig. 1 Representation model architecture

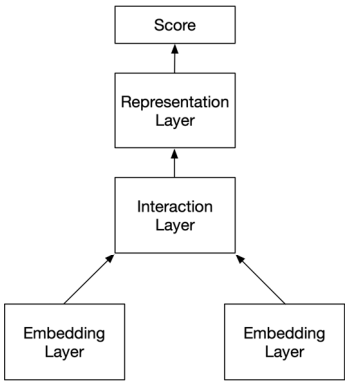

Fig. 2 Interactive model architecture

Interaction model: The interaction model abandons the idea of post-matching. Assuming that the global matching degree depends on the local matching degree, the first matching between words is performed in the interaction layer, and the interactive results are subsequently modeled. This method does not directly learn the semantic representation vectors of the two sentences, but at the bottom layer, it lets the two sentences interact in advance to establish some basic matching signals, and then find a way to fuse these basic matching signals into a matching score. The interaction model architecture is shown in Figure 2. This framework captures more interactive characteristics between two sentences, so it significantly improves compared to the first framework. The disadvantage is that it neglects the global information such as syntax, and the global matching information cannot be depicted by local matching information.

Representation models include ARC-II proposed by Huawei's Noah's Ark Laboratory in 2014 [12], IBM's ABCN [13] in 2016, MatchPyramid [14] of the Chinese Academy of Sciences, 2017 Zhu Xiaodan's ESIM [15] and 2018 Tencent's MIX [16].

Large-scale pre-training model: In recent years, with the rapid development of deep learning, pre-training techniques for natural language processing have made considerable progress. In the early days of natural language processing, word embedding methods such as Word2Vec were used to encode text for a long time. These word embedding methods can also be regarded as static pre-training techniques. However, this context-free text representation brings very limited improvement to subsequent natural language processing tasks, and cannot solve the problem of ambiguity. ELMo [17] proposed a context-sensitive text representation method, which can effectively deal with the problem of polysemous words. Since then, pre-training language models such as GPT [18] and BERT [19] have been proposed one after another. Among them, the BERT model has a significant effect on many typical downstream tasks, which has greatly promoted the technical development of natural language processing. Since then, the era of dynamic pre-training technology comes. A large number of pre-training language models such as BERT-based improved models XLNet [20], Roberta [21], ELECTRA [22] have emerged, and pre-training technology has become an indispensable mainstream technology in the field of natural language processing.

BERT uses Transformer's Encoder, the main innovation of which lies in the pre-train method, that is, it uses Masked Language Model (MLM) and Next Sentence Prediction to capture word and sentence-level representations respectively. The main contribution of ELECTRA is to propose a new pre-training task and framework, changing the generative MLM pre-training task to the discriminative Replaced Token Detection (RTD) task to determine whether the current token has been replaced by a language model.

Based on the advantages of BERT, this article proposes a new architecture BSSM (BERT based Siamese Semantic Model) suitable for text matching. The model uses BERT to encode two texts to obtain word embedding. Let the representation vectors of the two texts interact to obtain attention weights and generate new word embedding. Then the new and old word embedding are pooled and aggregated, and finally the two word embedding are concatenated and sent to the final prediction layer for prediction. Because of the Siamese architecture, the texts to be matched can be encoded in advance, which can greatly reduce the matching time. And the use of BERT for word granularity encoding can reduce the impact of OOV (out of vocabulary), also reduce the risk of word segmentation errors, and enhance generalization capabilities.

3. BERT BASED SIAMESE SEMANTIC MODEL

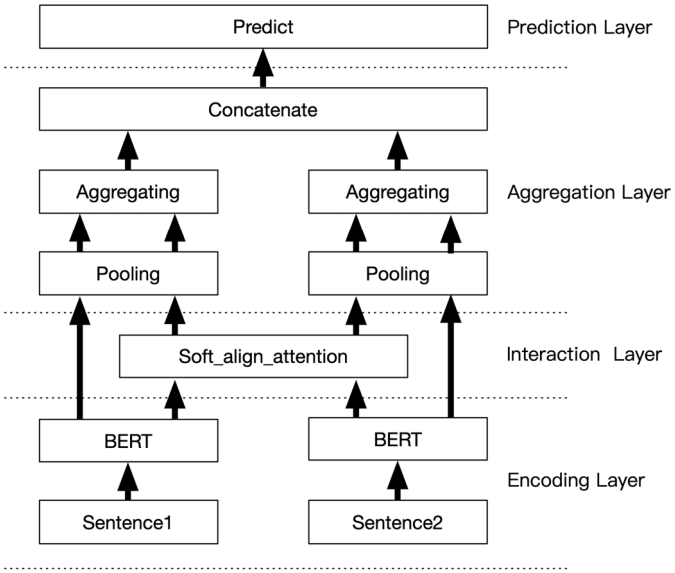

Fig. 3 BSSM（BERT based Siamese Semantic Model）model architecture

The model is mainly divided into 4 parts, namely encoding layer, interaction layer, aggregation layer and prediction layer, as shown in Figure 3.

Encoding layer: Use the BERT pre-training language model to encode the text a and b respectively, so that the representation information of the word a_i and b_j can be better learned, where the subscripts i and j respectively represent different moments of the text. Here $\overline{a}_i$ is used to represent the hidden layer output of the input text a at the time i of the BERT model (the representation vector of the word a_i), and $\overline{b}_j$ is the same, and the calculation formula is shown in equation (1).

$$\begin{aligned} \overline{a}_i &= BERT(a,i), i \in [1,...,l_a] \\ \overline{b}_j &= BERT(b,j), j \in [1,...,l_b] \end{aligned} \tag{1}$$

The reason why BERT is used here instead of other models such as LSTM or GRU is that their structure is only suitable for one-way language models (from left to right or from right to left), which also limits the model's representation ability, making it only obtain unidirectional contextual information. The BERT uses MLM for pre-training and uses a deep two-way Transformer (a one-way Transformer generally refers to the Transformer Decoder, and each word only pays attention to the word before it. The two-way Transformer refers to the Transformer Encoder, each of which word will notice all the word in the entire sentence.) to build the entire model, so finally generate a deep two-way language representation that can integrate left and right context information.

Interaction layer: The soft attention layer calculates the attention weight of the word information $<\overline{a}_i, \overline{b}_j>$ as the similarity between text a and text b. The calculation formula is shown in equation (2).

$$e_{ij} = \overline{a}_i^T \overline{b}_j \tag{2}$$

Through the obtained attention weight e_{ij}, the local correlation between text a and text b is obtained. Different attention weights are obtained for text a and text b, that is, text a is relative to text b, and text b is relative to text a. This is to capture the interactive information between two texts. For the representation of a word in text a, such as $\overline{a}_i$, the semantic similarity of each word in text b can be calculated by e_{ij}, and the calculation formula is shown in formula (3).

$$\begin{aligned} \tilde{a}_i &= \sum_{j=1}^{l_b} \frac{\exp(e_{ij})}{\sum_{k=1}^{l_b} \exp(e_{ik})} \overline{b}_j, i \in [1,...,l_a] \\ \tilde{b}_j &= \sum_{i=1}^{l_a} \frac{\exp(e_{ij})}{\sum_{k=1}^{l_a} \exp(e_{kj})} \overline{a}_i, j \in [1,...,l_b] \end{aligned} \tag{3}$$

When calculating $\tilde{a}_i$ here, the calculation method is weighted with sequence $\{\overline{b}_j\}$, so does $\tilde{b}_j$. It can be understood intuitively that the correlation between $\overline{a}_i$ and each word in the sequence $\{\overline{b}_j\}$ can be represented by $\tilde{a}_i$. Similarly, the correlation between $\overline{b}_j$ and each word in sequence $\{\overline{a}_i\}$ can also be represented by $\tilde{b}_j$. Figure 4 shows the operation of the interaction layer.

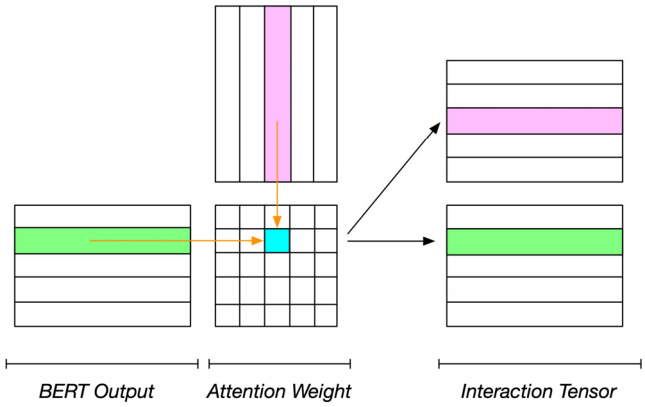

Fig. 4 Overview of the interaction layer

Aggregation layer: Since the lengths of text a and text b are not equal, the tensor dimensions of text features are inconsistent. Through the pooling layer, representation vectors with the same dimensions will be obtained. We first perform pooling operations on the previously extracted text features respectively, here is the max pooling, the calculation formula is shown in equation (4).

$$\begin{aligned} v_{\bar{a}} &= \max_{i=1}^{l_a} \bar{a}_i \quad ; \quad v_{\bar{b}} = \max_{j=1}^{l_b} \bar{b}_j \\ v_{\tilde{a}} &= \max_{i=1}^{l_a} \tilde{a}_i \quad ; \quad v_{\tilde{b}} = \max_{j=1}^{l_b} \tilde{b}_j \end{aligned} \tag{4}$$

The text representation is further enhanced through aggregation, and the two values of the representation $<\bar{a},\tilde{a}>$ of the text a are calculated for the difference. The calculation method is through the subtraction, and finally the two values of word representation and the subtracted value are concatenate together. The operation is the same for text b. The formula (5) is followed.

$$\begin{aligned} V_a &= Concatenate[v_{\bar{a}}; v_{\tilde{a}}; v_{\bar{a}} - v_{\tilde{a}}] \\ V_b &= Concatenate[v_{\bar{b}}; v_{\tilde{b}}; v_{\bar{b}} - v_{\tilde{b}}] \end{aligned} \tag{5}$$

Then, the vector representation of text a and text b is fused. Formula (6) is given as follows.

$$V = Concatenate[V_a; V_b; |V_a - V_b|] \tag{6}$$

Prediction layer: In the prediction layer, we input the results obtained from the previous step into a fully connected network to predict the similarity of text pairs. The entire model is trained end-to-end and uses Hinge Loss as the loss function. Hinge Loss is a loss function in the field of machine learning, and its most famous application is as the objective function of linear SVM. Tang et al. [23] studied neural networks with different SVM loss functions, and believed that the L2-SVM loss is the best. So we choose it in our model as shown in equation (7).

$$\min_w \frac{1}{2} w^T w + C \sum_{n=1}^{N} \max(1 - y\hat{y}, 0)^2 \tag{7}$$

Among them, $y \in \{+1, -1\}$ is the true label and $\hat{y}$ is the predicted value of the model.

4. EXPERIMENTAL RESULTS AND ANALYSIS

4.1. Data and its preprocessing

The data sets used in the experiment include LCQMC [24] and AFQMC (Ant Financial Question Matching Corpus). LCQMC provides 260068 pairs of labeled data, of which 238766 pairs are used as training data, 8802 pairs are used as verification data, and 12,500 pairs are used as test data. All the data in AFQMC come from the actual application scenario of the financial brain of Ant

Financial. The data set provides 42511 pairs of labeled data, of which 34334 pairs are used as training data, 4316 pairs are used as verification data, and 3861 pairs are used as test data. Including synonymous and non-synonymous pairs, the average length of the text is 13, the shortest text length is 3, and the longest is 112.

Each data sample has three properties, "sentence1," "sentence2," and "Label." Sentence1 and sentence2 are text pairs, Label is 1 for the semantics of sentence1 and sentence2 are similar, 0 means that the meaning of the two sentences is different, in order to adapt to the loss function used in this article, will use -1 to mean that the meaning of the two sentences is different. The evaluation metric is Accracy, which is defined as the proportion of the number of correctly classified text pairs to the total.

In the training process, some enhancements are made to the training data, and Back-translation is used to increase the amount of data, that is, the Chinese is first translated into English, and then translated back to Chinese. The Back-translation method not only has the ability to replace synonyms, but it also has the ability to add or remove words and reorganizes sentences while maintaining the original meaning. Through data enhancement methods, generating enhanced data similar to the original data will introduce new vocabulary, which will introduce a certain degree of noise, and help to prevent overfitting and allow the model to generalize to words that are in the test set but not in the training set.

4.2. Adversarial training

Szegedy et al. [25] first introduced the concept of adversarial examples: By adding some perturbations to the image, these subtle perturbations are difficult for the human eye to perceive, but they can make the neural network make wrong predict. In 15 years of ICLR, Goodfellow et al. first proposed the concept of adversarial training. In short, it is to add a perturbation r_{adv} to the original input sample x, and uses it for training after obtaining the adversarial sample. In other words, the problem can be abstracted into such a model. Formula (8) is as follows:

$$\min_{\theta} -\log P(y \mid x + r_{adv}; \theta) \tag{8}$$

Among them, y is the real label, and θ is the model parameter. At the same time, the authors proposed the Fast Gradient Sign Method (FGSM) to calculate the disturbance of the input sample. Disturbance can be defined as equation (9):

$$r_{adv} = \varepsilon \cdot \operatorname{sgn}(\nabla_x L(\theta, x, y)) \tag{9}$$

Among them, sgn is the symbolic function, and L is the loss function. In 2017 ICLR, Goodfellow [26] made a simple modification to the calculation of disturbance in FGSM. Assuming that the word vector $[v_1, v_2, ..., v_T]$ of the input text sequence is x, the perturbation calculation of the word vector is shown in equation (10). In fact, the sign function is cancelled and the “L2-norm” is used for scaling.

$$\begin{aligned} r_{adv} &= \varepsilon \cdot g / \| g \|_2 \\ g &= \nabla_x L(\theta, x, y) \end{aligned} \tag{10}$$

In a nutshell, adversarial training is to perform a gradient rise on the input (increase the loss), and performs a gradient descent on the parameters (to reduce the loss). Since the input will query the word embedding table, the actual method is to perform a gradient rise on the word embedding table. The following pseudo code shows the process of adversarial training:

For each x:

a) Calculate the forward loss of x and backpropagate to get the gradient

b) Calculate r according to the gradient of the Embedding matrix, and add it to the current Embedding, which is equivalent to x+r

c) Calculate the forward loss of x+r, backpropagate to get the gradient of the confrontation, and add it to the gradient of (a)

d) Restore Embedding to the value at (a)

e) Update the parameters according to the gradient of (c)

4.3. Training detials

When training BSSM, Adam is used as Optimizer, where BATCH SIZE is set to 32, and EPOCHS is set to 6. Training uses Early Stopping, which stops training when the model's performance on the validation set continues to decline. The learning rate is set to 2e-5, and the strategies used are Warmup and Cosine annealing. The learning rate increases linearly from 0 to 2e-5 in the Warmup phase,and then decreases as the value of the cosine function increases. This is because in the early stage of training, due to the distance frfffom the target, it is generally necessary to choose a large learning rate, but using too large a learning rate may easily lead to instability. So make a learning rate warm-up phase. At the beginning, use a smaller learning rate, and then adjust the learning rate back when the training process is stable. In the experiment, it is found that this method is indeed effective. The black line in Figure 5 is the learning rate curve, and the orange line is the loss curve during training.

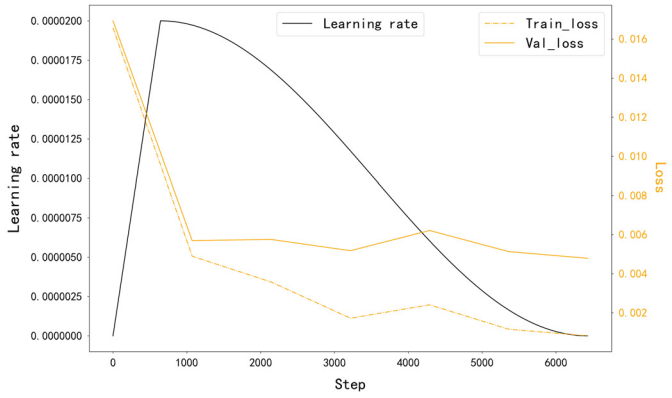

Fig. 5 Learning rate curve and Loss curve

4.4. Experimental results

Several text matching models with good effects are selected for comparative experiments. Among them, DSSM is a representation model, ARC-II, MatchPyramid and ESIM are interaction model, and BERT-base and RoBERTa-base are large-scale pre-training model. Table 1 shows the accuracy of each model on the data set. By comparing the results of experiments, it can be found that the proposed BSSM model is si sgnificantly best of the representation model and the interaction model, and the effect is also improved compared to the large-scale pre-training model.

In order to better understand the relative importance of BSSM, an ablation study was conducted. The results are shown in Table 2. First, evaluate the impact of different pooling strategies (mean, max, and first feature vector) and interaction layer on the results, and then test the use of different text pair concatenate methods in the aggregation layer and the use of different loss functions in the prediction layer. Finally verify the effect of adversarial training on the model.

Table 1 Experimental results

Model	**AFQMC**	**LCQMC**	
	Dev	***Dev***	***Test***
DSSM	63.3	76.5	74.1
ARC-II	68.2	85.1	84.8
MatchPyramid	67.9	83.9	84.1

ESIM	69.1	85.9	85.7
BERT-base-chinese	74.1	89.4	86.9
RoBERTa-wwm-ext-chinese	74.3	89.0	86.4
BSSM	75.5	90.1	86.7

Table 2 Compare the impact of different modules on the accuracy of AFQMC

		Accurate
Pooling Strategy	MEAN	75.2
	MAX	75.5
	CLS	75.0
Interaction Layer	With	75.5
	Without	72.9
Concatenate	[u, v]	68.9
	[u, v, u*v]	69.0
	[u, v, u-v]	72.7
	[u, v, |u-v|]	75.5
Loss Function	MSE	72.1
	Hinge Loss	75.5
Adversarial Training	With	75.5
	Without	75.0

It can be seen from the results of the ablation experiment that the pooling strategy has a small effect on the results, while the effect of the interaction layer is quite significant. Figure 6 shows the attention weight of the sample text in the interaction layer. The results of different concatenate methods are quite different, Where $|u*v|$ is basically not helpful to the result, the gain of $|u-v|$ is the most significant. In the choice of loss function, Hinge Loss can also effectively improve accuracy compared to MSE (Mean Squared Error). The introduction of adversarial training has improved the final result to a certain extent, which shows that adversarial training has a certain effect on improving the generalization ability of the model.

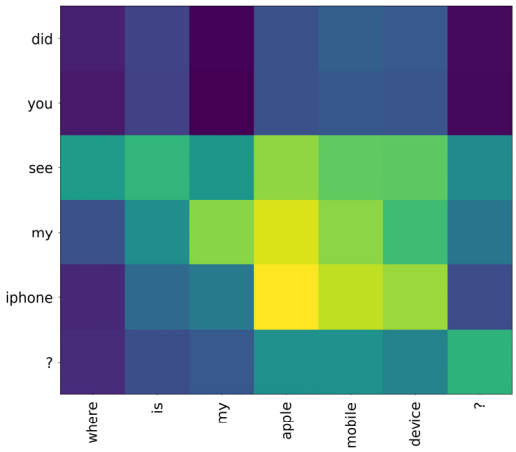

Fig. 6 An example diagram of the attention weight of the soft attention layer (the larger the square brightness, the larger the attention weight coefficient, which means the stronger the correlation)

The number of target texts for text matching is generally tens of thousands. Therefore, the model requires high computational efficiency. Using CPU and GPU hardware respectively, the necessary process of actual matching is simulated, and the calculation efficiency of different models is calculated. The results are shown in Figure 7. The experimental test is carried out in the following

server configuration: ADM-3700X CPU @ 3.6GHz, Nvidia RTX-2070 GPU, CUDA 10.1 and cuDNN, 32G memory.

The results show that the computational efficiency of the large-scale pre-training model of BERT is low. Because BERT uses a stack of 12 Transformers, the computational efficiency is very low regardless of whether it is on the CPU or on the GPU. The network used by ESIM is relatively simple, so the calculation efficiency is higher. The reason why BSSM can achieve this calculation efficiency is that it can omit the process of encoding matching text in the matching process. This process can be calculated and saved in advance, and the calculation overhead of some remaining modules is lower, so the calculation efficient.

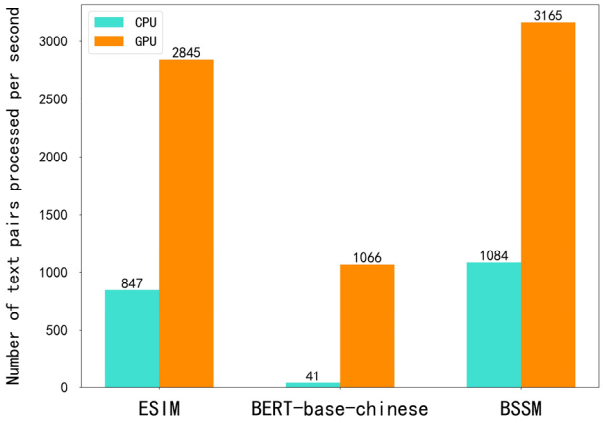

Fig. 7 Model calculation efficiency (the number of text pairs that the model can process per second)

5. Conclusion

This paper proposes a BERT-based Siamese Semantic Model (BSSM) to solve the shortcomings of the lack of interaction between text pairs in the traditional siamese architecture and the unclear semantic representation casued by the lack of feature extraction capabilities. BSSM encodes the text separately through BERT, and interacts then integrates prediction. The experimental results show that the model has a higher matching accuracy, better than the current generally recognized models, and the calculation efficiency is efficient. Due to the huge amount of text to be matched for text matching, and many applications of text matching have high requirements for real-time performance, follow-up research will continue to explore new methods to improve computational efficiency and further improve the matching accuracy.

Acknowledgment

This work was supported by Natural Science Foundation of Shanghai with Grant No. 19ZR1420800

References

[1] Salton G, Wong A, Yang C S. A vector space model for automatic indexing[J]. Communications of the ACM, 1975, 18(11): 613-620.
[2] Manku G S, Jain A, Das Sarma A. Detecting near-duplicates for web crawling[C]//Proceedings of the 16th international conference on World Wide Web. 2007: 141-150.
[3] Deerwester S, Dumais S T, Furnas G W, et al. Indexing by latent semantic analysis[J]. Journal of the American society for information science, 1990, 41(6): 391-407.
[4] Hofmann T. Probabilistic latent semantic analysis[J]. arXiv preprint arXiv:1301.6705, 2013.
[5] Blei D M, Ng A Y, Jordan M I. Latent dirichlet allocation[J]. Journal of machine Learning research, 2003, 3(Jan): 993-1022.
[6] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality[C]//Advances in neural information processing systems. 2013: 3111-3119.

[7] Pennington J, Socher R, Manning C D. Glove: Global vectors for word representation[C]//Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014: 1532-1543.
[8] Huang P S, He X, Gao J, et al. Learning deep structured semantic models for web search using clickthrough data[C]//Proceedings of the 22nd ACM international conference on Information & Knowledge Management. 2013: 2333-2338.
[9] Shen Y, He X, Gao J, et al. A latent semantic model with convolutional-pooling structure for information retrieval[C]//Proceedings of the 23rd ACM international conference on conference on information and knowledge management. 2014: 101-110.
[10] Palangi H, Deng L, Shen Y, et al. Semantic modelling with long-short-term memory for information retrieval[J]. arXiv preprint arXiv:1412.6629, 2014.
[11] Elkahky A M, Song Y, He X. A multi-view deep learning approach for cross domain user modeling in recommendation systems[C]//Proceedings of the 24th International Conference on World Wide Web. 2015: 278-288.
[12] Hu B, Lu Z, Li H, et al. Convolutional neural network architectures for matching natural language sentences[C]//Advances in neural information processing systems. 2014: 2042-2050.
[13] Yin W, Schütze H, Xiang B, et al. Abcnn: Attention-based convolutional neural network for modeling sentence pairs[J]. Transactions of the Association for Computational Linguistics, 2016, 4: 259-272.
[14] Pang L, Lan Y, Guo J, et al. Text matching as image recognition[C]//Thirtieth AAAI Conference on Artificial Intelligence. 2016.
[15] Chen Q, Zhu X, Ling Z, et al. Enhanced lstm for natural language inference[J]. arXiv preprint arXiv:1609.06038, 2016.
[16] Chen H, Han F X, Niu D, et al. Mix: Multi-channel information crossing for text matching[C]//Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018: 110-119.
[17] Peters M E, Neumann M, Iyyer M, et al. Deep contextualized word representations[J]. arXiv preprint arXiv:1802.05365, 2018.
[18] Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training[J]. URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf, 2018.
[19] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.
[20] Yang Z, Dai Z, Yang Y, et al. Xlnet: Generalized autoregressive pretraining for language understanding[C]//Advances in neural information processing systems. 2019: 5753-5763.
[21] Liu Y, Ott M, Goyal N, et al. Roberta: A robustly optimized bert pretraining approach[J]. arXiv preprint arXiv:1907.11692, 2019.
[22] Clark K, Luong M T, Le Q V, et al. Electra: Pre-training text encoders as discriminators rather than generators[J]. arXiv preprint arXiv:2003.10555, 2020.
[23] Yichuan Tang. Deep learning using support vector machines. CoRR, abs/1306.0239, 2, 2013.
[24] 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.
[25] Szegedy C, Zaremba W, Sutskever I, et al. Intriguing properties of neural networks[J]. arXiv preprint arXiv:1312.6199, 2013.
[26] Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples[J]. arXiv preprint arXiv:1412.6572, 2014.25