Semantic Matching for Short Texts: A Cross Attention Mechanism

Senhong Zheng*, Fan Chen, Xuejie Wang

Computer engineering and Science School, Shanghai University, Shangda Road, Shanghai 200444, China

*983124136@qq.com

Abstract. Semantic matching is a hot topic in the field of NLP. It is widely used in information matching, machine translation, intelligent Q&A, and answering system. With the rapid development of artificial intelligence, people pay more attention to friendly human-computer interaction and how to obtain accurate and effective information. Aiming at the problem of the diversity and structure of Chinese, this paper introduces the cross attention mechanism. Texts provide attention to each other, then to information retrieval. Experiments on aetc-nlp dataset show that the method is effective.

Keywords: Semantic matching, NLP, Cross attention mechanism, Deep neural network.

1. Introduction

With the rapid development of artificial intelligence and deep learning, it promotes the rapid development of the NLP. The semantic matching is widely used in intelligent applications, such as search engines, intelligent Q&A. Whether it is search or QA, the most fundamental problem is how to perform fast and correct text matching, so how to accurately understand the user's semantics is the most important step.

In the process of semantic matching, there are differences between Chinese and English. For example, English is separated by spaces. Compared with Chinese, it is easy to segment words by spaces and reduce the semantic differences caused by word segmentation. At the same time, it is reasonable to switch the position of English words in different contexts. Therefore, the model in this article is mainly to research the diversity and position of words in Chinese.

The diversity of word [1] means that different words can express the same meaning, such as the synonyms, Lotus and Water lotus represent the same plant; And the same word has different semantics in different contexts, Apple can be expressed as a kind of fruit, a company, or a brand. In addition, the position of words also has an impact on semantic matching. Different positions of the word have different semantics, such as Machine learning and Learning machine. They are matched in terms of words, but not in semantics.

In short text matching, we use Bi-LSTM [2] to capture the temporal information of the text to learn and encode the short text. At the same time, considering the influence of the position of the word in the text, we introduce position embedding and combine with the word vector into the vector space.
Then we use Bi-LSTM to get the context relationship between them. The attention mechanism can make the text focus on the information that has a close semantic relationship with the target.

According to the position information and the correlation between two texts, this paper introduces a cross attention semantic model (CASM) and the model uses a cross attention mechanism. The main contributions of this paper are as follows:

- Considering the position of the word in the text, we use the combination of word embedding and position embedding as an input vector to improve the matching accuracy of the model.
- Considering the correlation between texts, a cross-attention mechanism is introduced to obtain the semantic information under the action of the other text.

2. Related work

The traditional text matching algorithms are mainly based on keyword or pattern matching [3], which calculates similarity according to the common occurrence times of words. The classic algorithm is VSM [4]. This model was proposed by Salton et al. in 1975. Based on the Term Frequency-Inverse Document Frequency (TF-IDF), the text is represented by a high-dimensional space vector, and the distance between the vectors is calculated to represent the relationship between the text. However, the model is based on the Bag-Of-Words model (BOW) and does not consider the context relationship between the words in Chinese. It is easy to generate a sparse matrix and affect the final result.

In recent years, the research on text semantic matching has gradually shifted from traditional methods to deep semantic matching methods. Kim et al. [5] first used Convolutional Neural Network (CNN) to model sentences and achieved good results in text classification tasks. Based on, Hu et al. [6] extracted the semantic features from the two sentences, and obtained the representation vectors of the two sentences with the same length; then they are spliced and utilized the fully connected network to calculate the text similarity. Huang et al. [7] proposed a Deep Semantic Structured Model (DSSM) for the text semantic matching tasks. The model has a significant improvement compared with traditional methods. Shen et al. [8] proposed a covert semantic matching model to solve the problem that context information cannot be captured in. Palangi et al. [9] proposed a text-matching model based on long and short-term memory networks. The model encodes query items and documents separately through LSTM to obtain semantic vectors. Then calculating both vectors’ cosine distance as a measure of their similarity. Mueller et al. [10] used the twin model and manhattan distance to learn text matching and used the advantages of the LSTM network in text processing to extract text features through the weight-sharing LSTM network, and finally got good results.

At first, the attention mechanism was used in computer vision. Bahdanau et al. [11] first used the attention mechanism in the field of natural language processing and applied it to Neural Network Machine Translation (NMT). It not only improves the accuracy of the translation but also shows obvious advantages in a long text. Luong et al. [12] introduced the application of attention mechanisms in cyclic neural networks and divided the attention mechanism into the global mechanism and local mechanism. The global mechanism calculates the data of all time steps, and the local attention mechanism only calculates the data in the window. The method reduces the amount of calculation. Wang et al. [13] proposed an LSTM model for text matching and used LSTM to learn the matching degree between words and predict text matching. At each step of LSTM, the similarity between words is calculated and integrated through the attention mechanism. Because the cyclic neural network cannot be calculated in parallel, and the speed is slow. Yin et al. [14] first combined the attention mechanism and convolutional network and applied it to the text matching field. introduces three methods to integrate the attention mechanism into the convolutional neural network. ABCNN-1 uses the attention mechanism in the input layer to obtain the attention feature matrix and then combines with the original input matrix to construct a feature matrix with a channel number of 2. Then input it into the convolutional neural network. ABCNN-2 uses the attention mechanism to calculate the attention feature matrix after convolution and then sums the sentence representation matrix by weight. ABCNN-3 is a combination of ABCNN-1 and ABCNN-2.
3. The CASM model
The paper introduces the CASM model consists of four parts.
- **Input:** using the mixed embedding of word embedding and position embedding
- **Code:** using the Bi-LSTM to get the context information from the text
- **Interactive:** introducing a cross attention mechanism to get important semantic information of text
- **Output:** using the classifier to get the result of matching, and using sigmoid to update parameter

Figure 1 is the architecture diagram of the model. Firstly, we use the word embedding and position embedding to get a mixed embedding as an input vector. Then, we utilize the Bi-LSTM to consider the context information for each word to obtain coding information. After, we introduce the cross attention mechanism from the two texts to get more important information on the coding information. Finally, The classifier is used to analyze the vector and get the result. Using the sigmoid to update parameter.

![Architecture diagram of the CASM model](image)

3.1. Input
The input layer mainly maps the text to vector space. Then we obtain the word vector and position vector by word embedding and position embedding. Finally, we splice the word vector and position vector to get a new vector and send it to Bi-LSTM to code and get context information.

3.1.1. Word embedding
To obtain the word vectors for the text P and Q, we convert the short text P into a word vector \( P^c \), and the short text Q into a vector \( Q^t \). the manifestation is:

\[
P^c = [P^c_1, P^c_2, P^c_3, ..., P^c_n]
\] (1)

\[
Q^t = [Q^t_1, Q^t_2, Q^t_3, ..., Q^t_m]
\] (2)

where \( P^c_n \) is the nth word vector in the short text P, \( P^c_n \in R^{d_p} \), \( d_p \) is the dimension of the word vector, \( Q^t_m \) is the mth word vector in the short text Q, \( Q^t_m \in R^{d_q} \), \( d_q \) is the dimension of the word vector.

3.1.2. Position embedding
The position of words affects the semantic characteristics and can get different semantics expressed. This article uses position vectors to distinguish the word of different positions, and reflect the degree of association between words. As shown in formula 2, the sine function and cosine function are used to describe the distance between two words.

$$A(pos, 2i) = \sin \left( \frac{pos}{10000\pi} \right)$$  \hspace{1cm} (3)

$$A(pos, 2i + 1) = \cos \left( \frac{pos}{10000\pi} \right)$$  \hspace{1cm} (4)

where pos is the position of the word, the sine coding is used to the even position and the cosine coding is used to the odd position. and the final manifestation is: $A_i = [A_1, A_2, A_3, ..., A_n], A_i \in R^d, d$ is the dimension of the word vector.

Finally, we need to combine the word vector and the position vector, $SP^c = P^c \oplus A_i, SP^c$ is the final input vector of the text $P, SQ^t = Q^t \oplus A_j, SQ^t$ is the final input vector of the text $Q$, the manifestation is:

$$SP^c = [SP_1^c, SP_2^c, SP_3^c, ..., SP_n^c]$$  \hspace{1cm} (5)

$$SQ^t = [SQ_1^t, SQ_2^t, SQ_3^t, ..., SQ_m^t]$$  \hspace{1cm} (6)

3.2. Code
The Bi-LSTM network has a natural advantage in processing sequence data. It can make full use of context information to ensure that the past and future information in the time sequence can be fully considered. Bi-LSTM is a combination of forward LSTM and backward LSTM, The unit structure of LSTM is shown in Fig.1:

![Fig.2 The unit structure of LSTM](image)

The model of LSTM [15] is composed of the input word $X_t$, cell state $C_t$, temporary cell state $\tilde{C}_t$, hidden layer state $h_t$, forget gate $f_t$, memory gate $i_t$ and output gate $o_t$. Forgetting the information in the cell state and remembering new information and transfer the useful information for subsequent calculations, while useless information is discarded. According to the hidden layer state at the last time step $h_{t-1}$ and the input word $X_t$ to calculate the $f_t, i_t$ and $o_t$. The overall calculation steps are as follows:

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

$$i_t = \sigma(W_i, [h_{t-1}, X_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_c, [h_{t-1}, X_t] + b_c)$$

$$C_t = f_t \ast C_{t-1} + i_t \ast \tilde{C}_t$$

$$o_t = \sigma(W_o, [h_{t-1}, X_t] + b_o)$$

$$h_t = o_t \ast \tanh(C_t)$$  \hspace{1cm} (7)
Finally, we will get the hidden layer state sequence \( \{ h_0, h_1, ... h_{n-1} \} \). This paper uses the Bi-LSTM to code the text vector and get the semantic representation of the context information and words, the manifestation is:

\[
\begin{align*}
    h_t^P &= \text{BiLSTM}(h_{t-1}^P, P) \\
    h_t^Q &= \text{BiLSTM}(h_{t-1}^Q, Q)
\end{align*}
\]  

where \( h_t^P \) and \( h_t^Q \) is the semantic representation of one word which is coded by Bi-LSTM, and generate the input vector \( h_t^P \) and \( h_t^Q \).

### 3.3. Interactive

For getting a full integration between text P and text Q, in the interaction layer, this paper introduces a cross attention mechanism. The mechanism can help me get more important and accurate feature representation for the current task. \( \alpha_t \in R^{m \times n} \) is attention weight, which is calculated by a nonlinear function \( \tan \). Q-Attention: add attention to \( P \) with \( Q \), then the machine can pay more attention to the information in the text \( P \) that closely related to \( Q \) and obtain a matching semantic information of \( P \). The manifestation is:

\[
\begin{align*}
    u_P &= \sum_{i=1}^{n} \alpha_Q h_i^P, \forall i \in [1, ..., n] \\
    \alpha_Q &= \frac{\exp(u_t)}{\sum_{k=1}^{n} \exp(u_t)} \\
    u_t &= \tanh(W^T h_i^P + b)
\end{align*}
\]

where \( \alpha_Q \) is the attention weight of word \( h_i^P \) from vector \( h_i^Q \), \( W \) is offset matrix and \( b \) is offset, they are updated during the train. By suming the attention weight, we get the matching information \( u^P \) for text \( P \). In the same way, we get the matching information \( u^Q \) for text \( Q \).

When we get both matching information of texts, we need to combine matching information and coding information. The paper uses the method of calculating the difference sum element product. By this method, the model can reduce the difference between them. The manifestation is:

\[
\begin{align*}
    G^P &= [h^P, u^P, h^P - u^P, h^P \odot u^P] \\
    G^Q &= [h^Q, u^Q, h^Q - u^Q, h^Q \odot u^Q]
\end{align*}
\]

### 3.4. Output

In the output layer, we use two pooling mechanisms to aggregate two vectors into one vector, the average pooling, and the max pooling. Finally, we use the classifier to get the result for the matching. The formula of two pooling mechanisms is as follows:

\[
\begin{align*}
    G^P_{\text{avg}} &= \frac{1}{n} \sum_{i=0}^{n} G_i^P \\
    G^Q_{\text{avg}} &= \frac{1}{m} \sum_{j=0}^{m} G_j^Q \\
    G^P_{\text{max}} &= \max_{i=0}^{n} G_i^P \\
    G^Q_{\text{max}} &= \max_{j=0}^{m} G_j^Q
\end{align*}
\]

Then combine the two pooling vector to get a new vector of texts:

\[
G = \left[ G^P_{\text{avg}}; G^P_{\text{max}} \right] \quad G = \left[ G^Q_{\text{avg}}; G^Q_{\text{max}} \right]
\]
Using the method of difference sum element product to reduce the difference between them and get the final vector G:

\[ G = [G^p - G^q; G^p \odot G^q] \]  
(14)

Finally, use the classifier to get the result for the matching:

\[ y^\wedge = softmax(G) \]  
(15)

In the output layer, we select the loss function of sigmoid to update the \( y^\wedge \), the calculation formula is as follow:

\[ L = - \sum_{i=1}^{N} [y \log y^\wedge + (1 - y) \log(1 - y^\wedge)] \]  
(16)

where \( y \) is the real tag, \( y^\wedge \) is the forecast tag.

4. Experiments

4.1. Experimental data

In the part of experiments, the model in this paper experiment on the AETC-NLP[16]dataset. The dataset was provided by the financial brain competition of Ant Financial Services Developer Competition. It contains 1000 thousand pairs of Chinese semantic similarity calculation data. Tab.1 shows the example of the dataset. According to the example, we know it includes synonymous pairs and non-synonymous pairs. It contains id, two sentences, and one label which indicates whether they are synonymous pairs.

| Text pairs                                                                 | Label |
|----------------------------------------------------------------------------|-------|
| 1 How to change my mobile phone for the Alipay? The phone of Alipay is the old | 1     |
| 2 Can I opened the Alipay which is freezed? Can I open Alipay loan on my condition? | 0     |
| 3 Can't be staged after Alipay is overdue? Can I stage the Alipay when I pay off the | 0     |
| 4 The security verification of Alipay failed. What's wrong with not passing this security | 1     |

**Tab. 1** Example for ATEC-NLP

4.2. Parameter

In this paper, the unknown words were randomly initialized by uniform distribution, which obeyed the normal distribution of N (0,0.1). Tab.2 shows the other specific experimental parameters in the model.

| Parameters                  | Value |
|-----------------------------|-------|
| Bi-LSTM hidden layer size   | 160   |
| dropout rate                | 0.3   |
| Batch Size                  | 128   |
| Learning rate               | 0.01  |
| Word Size                   | 200   |

**Tab.2** Parameters
4.3. Evaluating indicator

We always use Accuracy, Recall, and F1-score as an indicator in NLP. In this paper, we also use the three indicators to measure the effect of the matching model. Accuracy and Recall can better show the effectiveness of classification. The calculation formula is as follows:

\[
\begin{align*}
    \text{pr} & = \frac{TP}{TP + FP} \\
    \text{rec} & = \frac{TP}{TP + FN} \\
    \text{acc} & = \frac{TP + TN}{TP + FP + TN + FN} \\
    F1 & = 2 * \frac{\text{pr} \times \text{rec}}{\text{pr} + \text{rec}}
\end{align*}
\]

where pr is the Precision Rate, acc is the Accuracy Rate, rec is the Recall Rate, TP is the number of samples for correct synonymous determination; FP is the number of samples for wrong synonymous determination; TN is the number of samples for correct non-synonymous determination, and FN is the number of samples for wrong synonymous determination.

4.4. Results and analysis

In this paper, the experimental result will be compared with the other four depth matching models, and finally, the results of each model are shown through the histogram. Four models included: DSSM [17], ESIM [18], ABCNN, BIMPM, uses a double layer of LSTM and attention mechanism to obtain semantic information; uses Bi-LSTM to encode the text and then uses a Bi-LSTM to fuse information; model uses CNN structure to extract features, and uses attention mechanism to further feature processing; [19] adopts bidirectional multi-angle matching, and adopts two-way multi-angle matching to obtain semantic information. Using matching-aggregation structure to calculate the similarity between two sentences. Five models are applied to the same dataset to get the matching results and analyse them.

| Model   | Acc (Dev) | F1 (Dev) | F1 (Test) |
|---------|-----------|----------|-----------|
| DSSM    | 0.788     | 0.553    | 0.602     |
| ESIM    | 0.815     | 0.578    | 0.642     |
| ABCNN   | 0.819     | 0.582    | 0.647     |
| BIMPM   | 0.821     | 0.585    | 0.650     |
| CASM    | 0.831     | 0.593    | 0.655     |

**Tab.3 Matching Result**

then we use the fig. 3 to show the result. It is convenient for us to analyse the model.
According to the figure, we can clearly see the F1-score of each model. Comparing with the other models, the CASM model proposed in this paper has a good effect in the field of semantic matching.

5. Conclusion
This paper proposes a short text semantic matching model based on cross attention. The model mainly introduces position embedding to pay attention to the semantic information of words in different positions. At the same time, a cross attention mechanism is introduced. Two texts provide attention to each other, extract the semantic information of high correlation, and perform matching analysis. Finally, good results are obtained on the ATEC-NLP dataset compared with other models.

References
[1] Liang P, Yanyan L, Jun X, Jiafeng G, Shengxian W and Xueqi C 2017 A review of deep text matching J. Chinese Journal of Computers 40(04) pp 985-1003
[2] Tiantian H, Yabo D, Jie H, Xiang L and Shaobo L 2020 News named entity recognition and sentiment classification based on Bi-LSTM and CRF based on attention mechanism J. Journal of Computer Applications 40(07) pp 1879-1883
[3] Cranias L, Papageorgiou H, Piperidis S 1994 A matching technique in example-based machine translation Proceedings of the 15th conference on Computational linguistics I pp 100-104
[4] Teruaki H and Yukio O 2019 Matrix-Based Method for Inferring Elements in Data Attributes Using a Vector Space Model 10(3)
[5] Y. Kim 2014 Convolutional neural networks for sentence classification Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing pp 1746-1751
[6] B. Hu, Z. Lu, H. Li and et al 2014 Convolutional neural network architectures for matching natural language sentences Advances in neural information processing systems pp 2042-2050
[7] Jadhav and Thepade 2019 Fake News Identification and Classification Using DSSM and Improved Recurrent Neural Network Classifier 33(12) pp 1058-1068
[8] Yi L 2018 Application of recommendation system based on LFM model in book reading J. Journal of Science & Technology Economics 26(29) pp 5-7+10
[9] Ma X and Hovy E 2016 End-to-end Sequence Labeling Labeling via Bi-directional LSTM-CNNsCRF Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics pp 1064-1074
[10] J. Mueller A and Thyagarajan 2016 Siamese recurrent architectures for learning sentence similarity Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2786-2792
[11] D.Bahdanau, K.Cho and Y.Bengio 2014 Neural machine translation by jointly learning to align and translate J.arXiv preprint arXiv:1409.0473
[12] Zhenzhen H, Xiaodong C and Ying L 2020 Semantic understanding model based on global and local Attention interaction mechanism Journal of Guilin University of Electronic Technology 40(02) pp 137-141
[13] Angen L, Sheng G, Yajing X 2018 Deep Semantic Match Model for Entity Linking Using Knowledge Graph and Text 129 pp 110-114
[14] Wenpeng Y, Hinrich S, Bing X and et al 2016 ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs 4 pp 259-272
[15] Yongcai T, Wenle W, Chaoyang H, Lei S and Lin W 2020 A text proofreading model combining LSTM and integrated algorithm Small Microcomputer System 41(05) 967-971
[16] Ant Financial. 2018 Ant Financial Artificial Competition
[17] Angen L, Sheng G and Yajing X 2018 Deep Semantic Match Model for Entity Linking Using Knowledge Graph and Text Procedia Computer Science p 129
[18] Guanyu L, Pengfei Z and Caiyan J 2020 A natural language reasoning model with enhancedattention Computer Engineering 46(07) pp 91-97
[19] Z. Wang, W. Hamza, R. Florian 2017 Bilateral multi-perspective matching for natural language sentences Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence pp 4144-4150

[20] Hamid P, Li D, Yelong S, Jianfeng G, Xiaodong H, Jianshu C, Xinying S and Rabab W 2016 Deep sentence embedding using long short-term memory networks IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP) 24(4)