A SHORT TEXT SIMILARITY CALCULATION METHOD BASED ON DEEP LEARNING

Yong XU[1], Yunke PENG[2], Hengna WANG[3,*], Xue'er WANG[4]

In addressing the limitations of traditional short text similarity calculation methods, this paper presents SMSABLC, a deep learning-based approach that takes into account polysemy, character order, and contextual semantics. Utilizing word embedding, SMSABLC converts short text into vectors, initiating the integration of a multi-head self-attention mechanism. This mechanism adeptly captures the relationship between words and the overall context within the short texts, spanning across multiple subspaces. Additionally, two opposing LSTM models are employed to acquire bidirectional sequential information from the character context in the short texts. Employing a combination of a convolutional layer and a pooling layer, CNN effectively facilitates the extraction of local character features inherent in the short text. Moving forward, the subsequent step entails the computation of the discernible difference existing between the two distinct short text vectors. This difference value is then inputted into a fully connected layer, whereupon the ensuing utilization of the Sigmoid function aids in the determination of the extent of similarity between the two textual entities. This article conducted experiments on two datasets, CCKS2018 and LCQMC, and compared with traditional methods and some deep learning based methods, SMSABLC achieved better experimental results. Through the analysis of experimental results, it can be concluded that SMSABLC achieves significant improvements in the calculation of short text similarity by effectively incorporating a wide range of deep semantic information.

Keywords: Short text similarity calculation; Deep learning; Multi-head self-attention mechanism; Bidirectional short-duration memory network; Convolutional neural network

1. Introduction

Short text similarity calculation, an integral component of Natural Language Processing, employs specific techniques to measure the similarity between texts. Its widespread application can be observed in intelligent question answering and text classification domains[1]. Short text similarity calculation plays a crucial role in

[1] Department of Computer Science & Technology, Anhui University of Finance & Economics, Bengbu 233000, China; xuyong@aufe.edu.cn
[2] School of Management Science & Engineering, Anhui University of Finance & Economics, Bengbu 233000, China; 1410939755@qq.com
[3] Department of Computer Science & Technology, Anhui University of Finance & Economics, Bengbu 233000, China; 120081267@aufe.edu.cn
[4] School of Management Science & Engineering, Anhui University of Finance & Economics, Bengbu 233000, China; 2173600168@qq.com

intelligent question answering systems by measuring the resemblance between the input question and the target question, enabling accurate and effective question response generation[2]. In the task of text classification, short text similarity calculation determines how similar a short text to be classified is to the pre-classified short texts[3].

The classical character-based method on short text similarity calculation calculates the matching and distance of the original text directly, without considering the implicit meaning of words or characters. The corpus-based method trains a model, which could obtain text meaning, based on corpus (such as Wikipedia corpus, Baidu encyclopedia corpus, web dynamic corpus). The method learns the meaning of words or characters, without considering the order of words or characters and cannot solve the problem of polysemy of a word. The method relying on a knowledge-base, which includes an ontology knowledge base and a network knowledge base, is employed to compute text similarity. This approach heavily relies on the expertise of knowledge base developers and necessitates the continual maintenance and updating of the knowledge base.

To address the issues with conventional approaches for calculating short text similarity, this research paper proposes the utilization of deep learning method. Specifically, a siamese neural network model called SMSABLC is developed. Within the SMSABLC model, multi head self attention (MHSA) component captures semantic information within separate subspaces across characters in diverse sentences. Bidirectional LSTM (BiLSTM) captures bidirectional sequential information within the text. Convolutional neural network (CNN) extracts pivotal features from text sequences, ultimately generating vectors that encapsulate profound semantic meaning. The model considers word or character order in sentences and word polysemy, eliminating the need for a knowledge base. This enables it to achieve improved accuracy in short text similarity calculations.

In the experiment, CCKS2018 and LCQMC datasets were used. On the CCKS2018 dataset, the accuracy and F1 of SMSABLC are both 93%; on the LCQMC, the accuracy and F1 of SMSABLC are 88.7% and 90.3%. Comparing SMSABLC model with traditional short text similarity calculation methods and several deep learning based short text similarity calculation methods, it is found that its accuracy and F1 are significantly improved. To assess the impact of various modules in SMSABLC, we compared it with individual modules such as MHSA, BiLSTM, and CNN, as well as combined modules including MHSA-BiLSTM, MHSA-CNN, and Bista-CNN. It is shown that the combined modules outperform the combined modules in terms of accuracy and F1. Furthermore, the SMSABLC model exhibits higher accuracy and F1 improvement compared to the combined module.

2. Related works

1.1 One-Hot coding

In the mechanism of One hot encoding, characters are represented using Sparse matrices. Each character is mapped to a binary vector, such as $[1,0,0 \dots \dots ,0]_{1*N}$ for the first character and $[0,1,0 \dots \dots ,0]_{1*N}$ for the second character. However, if the vocabulary size (N) is large, the text vector encoded by One hot becomes increasingly sparse, resulting in larger vector dimensions and higher consumption of computational resources.

1.2 Short text similarity calculation

Table 1

Methods for Calculating Short Text Similarity

Method	Model	Advantages and disadvantages
Character-based method	Longest common sequence[4]	The principle is simple, easy to implement, and does not rely on external data. Characters or words are independent, without considering the meaning of words and the relationship between words.
	N-Gram[5]	
	Jaccard[6]	
Corpus-based method	Vector Space Model[7]	Simple and effective, utilizing the semantic information of words. Unable to solve the problem of polysemy, ignoring the order and correlation between words.
	Word2Vec[8]	
	LSA[9]	
	LDA[10]	
Knowledge-based method	Based on HowNet[11]	Apply semantic knowledge of words in the knowledge base. The knowledge in the knowledge base needs to be maintained and updated.
	Based on WordNet[12]	
Deep learning-based method	MaLSTM[13], AttMaLSTM[14], MAS-BI-LSTM[15], CNN-LSTM[16], SiaCNN-BiLSTM[17]	Learn deep semantic information of words, such as text order information, local features, relationships between words, and so on. Need more resources for training.

1.2.1 Character-based method

Character-based short text similarity calculation method utilizes character conversion techniques to evaluate similarity without relying on external knowledge bases or corpora.

In order to determine the similarity between two texts, Elhadi[4] employed the longest common sequence algorithm, which takes into account both word parts and syntactic characteristics.

Sultana and Biskri[5] utilized the N-Gram method to generate N-Gram Distance matrices for two texts of length 3, and determined the similarity using the Jaccard distance. The Jaccard algorithm, a computational technique widely used in text analysis, capitalizes on the intrinsic properties of sets to effectively gauge the degree of similarity between short texts. This measure is achieved by meticulously

calculating the ratio of the number of identical words shared by the two texts to the total count of distinct words present in both texts[6].

The character-based method calculates the similarity of short texts by matching characters, without considering the semantic meaning, word relationships, synonyms, or polysemy. As a result, it suffers from low accuracy in similarity calculations.

1.2.2 Corpus-based method

By using a large text corpus to convert text data into semantic vectors, and then using the similarity measure of semantic vectors as the similarity of the text, the corpus-based method makes it possible to combine accurate similarity calculation with semantic information.

Salton et al.[7] introduced a vector space model, in which they proposed a process that includes determining the feature index system, constructing feature vectors for short texts. Several scholars have employed neural network methods to derive vector representations of text from corpora.

One highly influential approach is the Word2Vec method introduced by Mikolov et al.[8] It has two training methods, CBOW and Skip Gram. These vectors can subsequently be used to evaluate the similarity between texts.

Schwarz[9] employed latent semantic analysis (LSA) to determine the similarity between texts. A matrix is constructed in the domain of LSA, with rows representing words and columns representing texts. The entries in the matrix can represent various characteristics, such as the frequency of words in the text. When processing large amounts of text, the matrix dimensions can be large, then take a specific approach to dimensionality reduction. This dimensionality reduction helps in making computations more efficient while retaining the essential semantic information.

Latent Dirichlet Allocation is a widely utilized Topic model proficient in capturing the correlation between a document and a topic, as well as the interrelationship between a topic and words[10]. Through the utilization of learned relationships, LDA constructs a document vector and evaluates similarity.

The inclusion of semantic information extracted from the text remains a fundamental aspect of the corpus-based method. However, it is necessary to underline the impact of the choice of training corpora on determining the algorithm's outcomes and effectiveness. It is important to consider potential limitations in capturing the necessary contextual information, as this approach may encounter challenges in accurately understanding and interpreting the diverse meanings of words within specific contexts.

1.2.3 Knowledge-based method

By employing the knowledge in the knowledge base, the knowledge-based method computes similarity.

HowNet is a semantic resource capturing Chinese and English word concepts,

relationships, and attributes. Based on HowNet knowledge, Zhang Lin et al.[11] computed semantic similarity by analyzing word relationships within sentences. They constructed a word similarity matrix for the two texts and determined text similarity based on this matrix.

WordNet functions as a repository of concepts or words, encompassing definitions, a thesaurus, and semantic connections among words. Shajalal[12] measured the similarity of short texts by leveraging a combination of semantic similarity derived from WordNet's concept relationships and pre-trained word embeddings.

By leveraging semantic information, the knowledge-based approach demonstrates efficient utilization, leading to highly accurate similarity calculation for short texts. However, it overlooks the influence of word order and context on sentence semantics, potentially constraining its effectiveness.

1.2.4 Deep learning-based method

Deep learning has become a powerful tool in various domains, offering valuable insights in the calculation of short text similarity.

A deep learning model that has garnered significant attention is LSTM, attributed to its proficiency in processing sequence data. The utilization of LSTM to gauge the similarity of short texts has become commonplace among researchers. Liu et al.[18] used BiLSTM model to encode the text, and then learned important parts of the text using self-attention components, and finally output similarity through SoftMax function. Mueller et al.[13] employed the siamese LSTM model for calculating short text similarity. This model is effective in capturing sequential information within the text and measuring similarity through the Manhattan distance. Othman et al.[19] and Bao et al.[14] enhanced Muller's foundational work by incorporating an attention mechanism, which effectively determines the significance of words within the text. This addition aimed to improve the overall effectiveness of the model. Wang[15] employed the Word2Vec technique to acquire word vectors. Subsequently, feature vectors were extracted using a siamese BILSTM and a multi-head attention mechanism model. Finally, the similarity between short texts was computed based on these feature vectors.

CNN has the ability to extract local features, so many scholars pay attention to it, and it is combined with other deep learning models to calculate short text similarity. Both CNN and LSTM are integrated into Mansoor et al.[20] and Pontes et al.'s[16] short text similarity model for improved performance. By combining these two architectures, the model achieves an effective calculation of text similarity, while also capturing the interdependencies between words and sentences. Agarwal et al.[17] used CNN and BiLSTM to assess the similarity of texts, BiLSTM could learn the information of bidirectional text sequences. Mahmoud et al.[21] combined CNN and attention mechanism to calculate Arabic text similarity. The Glove technique was employed to extract word vectors representing global textual information.

To summarize, the deep learning-based method for evaluating similarity in short

texts utilizes deep learning models to extract various levels of text features. This method surpasses traditional approaches by incorporating more sophisticated semantic features, thereby enhancing the accuracy of similarity computation.

3. SMSABLC short text similarity calculation model

a. Model's architecture

To address the limitations of conventional approaches for calculating similarity in short texts, which do not account for semantic information, sequence information, and contextual meaning, the paper introduces a novel model called SMSABLC. This model aims to tackle these challenges and provide an effective solution for short text similarity calculation. Fig. 1 illustrates the architectural design of the model. Initially, the short text undergoes conversion into its respective vector representation via the Embedding layer. Following that, the implementation of MHSA enables the acquisition of semantic vectors for the short text across multiple distinct semantic spaces. And after processing by MHSA, the vectors obtained for the same word in different short texts are different, solving the problem of polysemy. Then, BiLSTM is used to learn the information in the left and right directions of the short text sequence. Next, the local information of the short text was learned through CNN; Finally, in the output layer, the two short texts are combined with the vectors obtained through the input coding layer, and the similarity between 0 and 1 is obtained through the full connection layer and Sigmoid function.

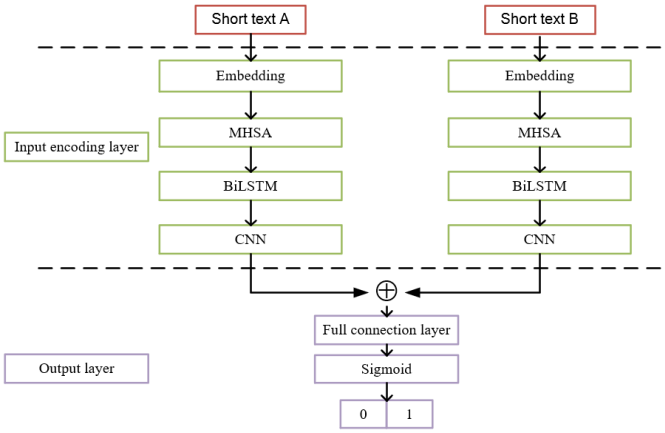

Fig. 1. Model's architecture

b. Input coding layer

i. Embedding layer

Deep learning models cannot process raw text directly, so text should be converted into vector. Using One-hot encoding to encode a short text with m characters (length), a two-dimensional vector m*n is obtained (n represents the number of

characters). When the number of characters in the character table is very large, the two-dimensional vector dimension n of One-hot encoding is large, which leads to a large demand for computing resources in the following steps. With the aim of resolving this predicament, the SMSABLC model employs the Embedding layer to diminish the dimensionality of the m*n short-text One-hot coding vector, which is originally high-dimensional.

$$\begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix}_{m*n} * \begin{pmatrix} w_{11} & \cdots & w_{1d} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nd} \end{pmatrix}_{n*d} = \begin{pmatrix} a_{11} & \cdots & a_{1d} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{md} \end{pmatrix}_{m*d} \quad (1)$$

Where, d represents the dimension of short text vector after dimensionality reduction. Short-text One-hot coding vector matrix is multiplied by n*d matrix to obtain a low-dimensional short-text vector in m*d dimension.

ii. Multi Head Self Attention layer

Within the SMSABLC model, the initial step involves obtaining a short text vector E_{m*d} through the Embedding layer. Subsequently, this vector is fed into the MHSA module, which accomplishes the task of capturing and integrating multiple layers of semantic information. In natural language, the phenomenon of "same character but different semantics" will often appear. In MHSA, the self attention mechanism calculates the correlation between words and other words in short text, so the word vectors obtained by the self attention mechanism in different short texts of the same word are different, thus solving the problem of "same word but different meanings". The self-attention mechanism learns sequence dependencies and utilizes weights to gauge their significance. At first, the self-attention mechanism requires linear transformation of the initial feature matrix, which is transformed into the query matrix Q_{m*d_k}, the key matrix K_{m*d_k} and the value matrix V_{m*d_v}, and then the attention value is calculated by the attention function 2.

$$\mathrm{Attention}(\mathrm{Q},\mathrm{K},\mathrm{V}) = \mathrm{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)\mathrm{V} \quad (2)$$

d_k represents the dimensions of both the query matrix and the key matrix.

By applying h linear transformations, MHSA generates h numbers of (Q, K, V) using the self-attention mechanism. Using formula 3, we calculate the self-attention value for each $(\mathrm{Q},\mathrm{K},\mathrm{V})$.

$$\mathrm{h}_i = \mathrm{Attention}(\mathrm{Q}_i,\mathrm{K}_i,\mathrm{V}_i) \quad (3)$$

$(\mathrm{Q}_i,\mathrm{K}_i,\mathrm{V}_i)$ represents the result of one of the linear transformations.

The self-attention value h_i of each head is calculated and then spliced together to get the multi-head self-attention value.

$$\mathrm{MultiHeadSelfAttention}(\mathrm{Q},\mathrm{K},\mathrm{V}) = \mathrm{concat}(\mathrm{h}_1,\ldots\ldots,\mathrm{h}_i) \quad (4)$$

Where the function of concat joint d_k dimension, the obtained multi-head self-attention value dimension is $m*(d_k*\mathrm{h})$.

iii. BiLSTM layer

The acquisition of knowledge from distant characters is facilitated by LSTM, a type

of recurrent neural network. Its structure is illustrated in Fig. 2. At first, semantic vector containing the information between words in different semantic spaces was learned through MHSA, then it was inputted into BiLSTM to extract the bidirectional sequential information of the short text.

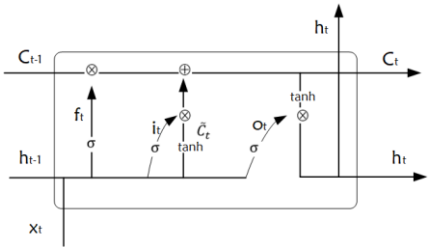

Fig. 2. LSTM model

The forget gate merges the preceding unit's output and the current unit's input, facilitating the identification of information to be discarded.

$$f_t = S\left(W_f * [h_{t-1}, x_t] + b_f\right) \quad (5)$$

In this equation, $0 \leq f_t \leq 1$, S is a sigmoid function, x_t represents a single character vector in the short text input by this unit, h_{t-1} is the previous unit's output, W is weight matrix.

To control the influx of new information, the input gate harmonizes the previous unit's output and the current unit's input..

$$\tilde{C}_t = \tanh(W_C * [h_{t-1}, x_t] + b_C) \quad (6)$$

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \quad (7)$$

$$C_t = f_t * C_{t-1} + \tilde{C}_t * i_t \quad (8)$$

In this equation, i_t is the updated information determined by the sigmoid function, $\tilde{C}$ is the generated alternative content via tanh, C_t is the updated information.

The output gate decides what information would be output.

$$h_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) * \tanh(C_t) \quad (9)$$

Where, h_t refers to the output or hidden state of an LSTM unit at a specific time step.

The BiLSTM architecture consists of dual LSTM layers working in opposite directions. By employing the positive and negative directions LSTM, the input sequence is generated. This enables the extraction of forward and reverse information from the sequence individually, which are later merged together as the output of the BiLSTM.

iv. CNN layer

In order to extract vital local features from the short text, the CNN is employed to process the semantic information obtained from the BiLSTM layer. By leveraging a convolutional layer, the CNN effectively captures the local features inherent

within the short texts, which are subsequently subjected to dimensionality reduction via a pooling layer. This approach ensures the extraction of key features vital to the overall understanding of the short texts. The main transformation functions are shown as equation 10 and 11.

$$\mathrm{C} = \mathrm{f}(\mathrm{E} \otimes \mathrm{W}_c + \mathrm{b}) \tag{10}$$

$$\tilde{C}_{ij} = \max(C_{mn}) \tag{11}$$

Where, E is the input short text vector, $\otimes$ is the convolution operation, f represents the activation function (such as Relu, Sigmoid, Tanh, etc.); C_{mn} indicates the coverage range of pooled cores and max indicates the maximum pooled operation.

c. Output layer

Utilize the input encoding layer to generate embeddings E1 and E2 for two short texts. Employ matrix subtraction to compute the discrepancy between the two vectors.

$$\mathrm{E} = E_1 - E_2 \tag{12}$$

Where E is the difference between two short text semantic vectors.

The dimensionality of E is reduced using the full connection layer to obtain a value x. By employing the Sigmoid function, x can be transformed into a bounded value within the interval of 0 to 1, and a threshold value c is set. If the value exceeds the threshold c, the output is set to 1, indicating a similarity between the two short texts; conversely, if the value falls below the threshold, the output becomes 0, indicating dissimilarity between the two short texts. The transformation functions are shown as equation 13 and 14.

$$f(x) = \frac{1}{1+e^{-x}} \tag{13}$$

$$g(x) = \begin{cases} 1 & f(x) > c \\ 0 & f(x) \leq c \end{cases} \tag{14}$$

Where, $f(x)$ is the Sigmoid function; $g(x)$ output 1 or 0 according to and threshold c.

4. Experimental

a. Dataset

The datasets used in this article is as follows.

CCKS2018 is a dataset designed for Task 3 of the 2018 National Knowledge Graph and Semantic Computing Conference, titled "Wezubank Intelligent Customer Service Problem Matching Competition". The data theme is about banking issues. The dataset contains a total of 100000 pieces of data, with one piece containing two questions and one label, which represents the semantic relationship between questions. A 0 tag indicates that two questions have different semantics, while a 1 tag indicates that two questions have the same semantics. The ratio of 0 label data to 1 label data in the dataset is 1:1.

LCQMC[22] is a dataset composed of questions extracted from Baidu Zhidao, which

includes questions from multiple fields. The composition of a single piece of data is the same as that of CCKS2018. The test data provided in the LCQMC dataset does not have labels, so this article uses train data and dev data as the datasets. There are a total of 247569 pieces of data in the dataset, and the ratio of 0 label data to 1 label data in the dataset is 10:7.3.

In the experiment, the dataset was divided in an 8:1:1 ratio and applied to all experimental models.

b. Experimental parameter

Setting the embedding layer dimension d to 128. We employed 8 MHSA headers with d_k and d_v values set to 64. The BiLSTM architecture utilized a one-way hidden layer with 128 units. The convolutional neural network (CNN) consisted of 512 filters, each with a size of 3. In the pooling layer, we adopted two different sizes for the CNNs, namely 4 and 2, respectively. In order to conduct the training, a total of 50 epochs were executed, utilizing the Adam optimizer[23] , the learning rate is 0.0001. A batch size of 128 was employed, and the maximum text length was defined as 32, taking into consideration the question length present in the dataset.

c. Evaluation index

This research focuses on assessing the similarity of short texts, using accuracy rate (Acc) and F1 as the evaluation criteria. Equations 15 and 16 depict the respective calculation formulas for these evaluation criteria.

$$\mathrm{Acc} = \frac{TP+FN}{TP+TN+FP+FN} \tag{15}$$

$$\begin{cases} P = \frac{TP}{TP+FP} \\ R = \frac{TP}{TP+FN} \\ F1 = \frac{2*P*R}{P+R} \end{cases} \tag{16}$$

TP means: predication is 1, truth is 1. FN means: predication is 0, truth is 1. FP means: predication is 1, truth is 0. TN means: predication is 0, truth is 0.

d. Results and analysis

(1) Comparison with classical short text similarity calculation model

In order to compare the performance of different short text similarity calculation models, this article trains these models on the CCKS2018 and LCQMC datasets.

We compared it with traditional methods, such as Jaccard[6], Word2Vec[8], and HowNet-based approaches[11]. Additionally, we incorporated deep learning-based approaches like MaLSTM[13], AttMALSTM[14], and CNN-LSTM[16], ABCNN[24] to ensure a comprehensive evaluation. Table 2 presents the experimental results, showcasing an outstanding achievement for the SMSABLC model with an accuracy and F1 score of 93%. Notably, the SMSABLC model outperforms traditional methods of short text similarity calculation, exhibiting a noteworthy improvement in accuracy and F1 score. Furthermore, compared with deep learning based methods,

the experimental results of SMSABLC on both datasets also showed some improvement.

With the development of deep learning, researchers have trained pre trained language models using large-scale corpus. Pre trained language models can output text vectors containing semantic information. Therefore, based on this, researchers have added deep learning models to pre trained language models for different natural language processing tasks, hoping to achieve better training results. This article reproduces two short text similarity models based on pre trained language models, BERT-BiLSTM-Attention[25] and BERT-BiLSTM-MaxPool[26], and obtains experimental data trained on the selected dataset in this article. According to Table 2, The experimental results of SMSABLC are higher than BERT BiLSTM MaxPool and BERT BiLSTM Attention.

In order to obtain the application effect of pre trained language models on the SMSABLC model, this paper selects BERT[27] and ERNIE3.0[28] pre trained language models as the methods for converting text into vectors based on SMSABLC. The selected dataset is used for training and experimental data is obtained. According to Table 2, on CCKS2018 dataset，when selecting BERT and ERNIE3.0 as the embedding layers of the model, the accuracy and F1 of the SMSABC model were slightly improved; on LCQMC dataset, when BERT is selected as the embedding layer, the evaluation index slightly decreases; when ERNIE3.0 is selected, the evaluation index slightly improves. On the premise of similar accuracy and F1 results, adding BERT and ERNIE3.0 pre trained language models takes up more graphics memory and uses more training time.

Table 2

Comparison of experimental results of traditional models

Models	CCKS2018		LCQMC	
	Acc(%)	F1(%)	Acc(%)	F1(%)
Jaccard	63.0	68.8	76.3	80.9
Word2Vec	64.2	71.1	78.2	82.2
HowNet	59.9	60.9	61.8	67.3
ABCNN	82.5	82.5	83.0	85.8
MaLSTM(Word2Vec)	85.4	84.7	84.0	86.3
AttMaLSTM(FastText)	86.3	86.0	86.2	88.1
CNN-LSTM(Word2Vec)	89.3	89.0	84.5	86.8
BERT-BiLSTM-MaxPool	90.9	90.9	87.0	88.8
BERT-BiLSTM-Attention	92.0	91.9	87.7	89.4
SMSABLC	93.0	93.0	88.7	90.3
BERT+SMSABLC	93.2	93.3	88.0	89.8
ERNIE3.0+SMSABLC	93.5	93.5	89.1	90.7

(2) Utility analysis of each module in the model

Our analysis includes the SMSABLC model, single module models (MHSA, BiLSTM, CNN), and two module combination models (MHSA BiLSTM, MHSA

CNN, BiLSTM CNN). Table 3 presents the experimental results for each model. Through the integration of two single modules, the two module combination model excels in short text similarity calculation by enhancing feature learning, surpassing the capabilities of a single module. The introduction of a two module combination model yields a marked increase in accuracy, ranging from 2.3% to 7.5%, surpassing that of a single module model. Simultaneously, the F1 score demonstrates an improvement of 2.4% to 6.7%. The SMSABLC model combines three single modules, and can learn more short text features compared with the 2-module combined model. It is evident from the experimental outcomes that the implementation of the SMSABLC model resulted in a substantial increase in accuracy, with improvements ranging from 2.8% to 3.5%, while F1 scores experienced enhancements between 2.8% and 3.4%.

In comparison to the single module and two module combination models, the SMSABLC model excels at capturing a richer set of features in short texts, leading to higher accuracy and F1, as evident from the experimental evaluation.

Table 3

Experimental results of different modules

Models	Acc(%)	F1(%)
MHSA	82.7	83.5
BiLSTM	86.9	86.9
CNN	84.3	84.5
MHSA-BiLSTM	89.5	89.6
MHSA-CNN	89.2	89.3
BiLSTM-CNN	90.2	90.2
SMSABLC	93.0	93.0

5. Conclusion

The main contribution of this article is the introduction of the SMSABLC model. MHSA to accurately capture the semantic relationship between words and the entire text for short text similarity calculation, BiLSTM learns the sequential semantic features of words in short texts, and MHSA-BILSTM overcomes the polysemy problem of one word by learning the deep semantic information of words in short texts, for the same word has different semantic vectors in different short texts, CNN extracts local features of the semantic vector obtained by MHSA-BiLSTM, which are beneficial to the calculation of short text similarity. Compared with other classical models, SMSABL has better performance.

The SMSABLC could be used in intelligent question answering, short text classification, plagiarism detection and other fields, and has good accuracy and practicability. Our future work will involve the utilization of a short text similarity

computation approach to optimize the performance of intelligent question answering systems.

Acknowledgement

The work was supported by Philosophy and Social Science planning project of Anhui Province (Grant No. AHSKF2021D31), the Key Research and Development Project of Anhui Province (Grant No. 2022O07020001), The Graduate Research and Innovation Fund(Grant No. ACYC2022016).

R E F E R E N C E S

[1]. D. W. Prakoso, A. Abdi, C. Amrit, "Short text similarity measurement methods: a review", in Soft Computing, vol. 25, no. 6, Jan. 2021, pp. 1-25.

[2]. S. G. Aithal, A. B. Rao, S. Singh, "Automatic question-answer pairs generation and question similarity mechanism in question answering system", in Applied Intelligence, vol. 51, no. 11, Nov. 2021, pp. 8484-8497.

[3]. S. D. Ma, D. S. Liu, "Text Classification Method Based on Weighted Word2vec", in INFORMATION SCIENCE, vol. 37, no. 11, Nov. 2019, pp. 38-42.

[4]. M. T. Elhadi, "Text similarity calculations using text and syntactical structures", in 2012 7th International Conference on Computing and Convergence Technology (ICCCT), Dec. 2012, pp. 715-719.

[5]. S. Sultana, I. Biskri, "Identifying Similar Sentences by Using N-Grams of Characters", in Recent Trends and Future Technology in Applied Intelligence, May. 2018, pp. 833-843.

[6]. S. Wu, F. Liu, K. Zhang, "Short text similarity calculation based on jaccard and semantic mixture", in Bio-Inspired Computing: Theories and Applications: 15th International Conference, Apr. 2021, pp. 37-45.

[7]. G. Salton, "A vector space model for automatic indexing", in Communications of the ACM, vol. 18, no. 11, Nov. 1975, pp. 613-620.

[8]. T. Mikolov, K. Chen, G. S. Corrado, et al., "Efficient Estimation of Word Representations in Vector Space", in ICLR (Workshop Poster) 2013, May. 2013, pp. 1-12.

[9]. C. Schwarz, "lsemantica: A command for text similarity based on latent semantic analysis", in The Stata Journal: Promoting communications on statistics and Stata, vol. 19, no. 1, Mar. 2019, pp. 129-142.

[10]. V. Rus, N. Niraula, R. Banjade, "Similarity measures based on latent dirichlet allocation", in CICLing 2013: Computational Linguistics and Intelligent Text Processing, Mar. 2013, pp. 459–470.

[11]. L. Zhang, J. Hu, "Sentence Similarity Computing for FAQ Question Answering System", in Journal of Zhengzhou University(Natural Science Edition), vol. 42, no. 1, Mar. 2010, pp. 57-61.

[12]. M. Shajalal, M. Aono. "Semantic textual similarity between sentences using bilingual word semantics", in Progress in Artificial Intelligence, vol. 8, no. 2, Mar. 2019, pp. 263-272.

[13]. J. Mueller, A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity", in Thirtieth AAAI Conference on Artificial Intelligence, Feb. 2016, pp. 2786–2792.

[14]. W. Bao, W. Bao, J. Du, et al., "Attentive Siamese LSTM network for semantic textual similarity measure", in 2018 International Conference on Asian Language Processing (IALP), Nov. 2018, pp. 312-317.

[15]. Z. Wang, B. Zhang, "Chinese Text Similarity Calculation Model Based on Multi-Attention Siamese Bi-LSTM", in 2021 4th International Conference on Computer Science and Software Engineering, Dec. 2021, pp. 93–98.
[16]. E. L. Pontes, S. Huet, A. C. Linhares, et al., "Predicting the Semantic Textual Similarity with Siamese CNN and LSTM", in CORIA-TALN-RJC (TALN 2), May. 2018, pp. 311-320.
[17]. B. Agarwal, M. K. Gupta, H. Sharma, et al., "Siamese-Based Architecture for Cross-Lingual Plagiarism Detection in English-Hindi Language Pairs", in BIG DATA, vol. 11, no. 1, Oct .2022, pp. 48-58.
[18]. K. Liu, Y. Zhang, C. Xing, "Hybrid Attention Based Neural Architecture for Text Semantics Similarity Measurement", in Database Systems for Advanced Applications: 25th International Conference, Sept. 2020, pp. 90-106.
[19]. N. Othman, R. Faiz, K. Smaïli, "Learning English and Arabic question similarity with Siamese Neural Networks in community question answering services", in DATA & KNOWLEDGE ENGINEERING, vol. 138, no. C, Mar. 2022, pp. 1-14.
[20]. M. Mansoor, Z. U. Rehman, M. Shaheen M, et al., "Deep Learning Based Semantic Similarity Detection Using Text Data", in INFORMATION TECHNOLOGY AND CONTROL, vol. 49, no. 4, Dec. 2020, pp. 495-510.
[21]. A. Mahmoud, M. Zrigui, "Hybrid Attention-based Approach for Arabic Paraphrase Detection", in Applied Artificial Intelligence, vol. 35, no. 15, Dec. 2021, pp. 1271-1286.
[22]. X. Liu, Q. Chen, C. Deng, et al., "Lcqmc: A large-scale chinese question matching corpus", in the 27th international conference on computational linguistics, Aug. 2018, pp. 1952-1962.
[23]. D. P. Kingma, J. Ba, "Adam: A Method for Stochastic Optimization", in ICLR 2015, May. 2015, pp. 1-13.
[24]. W. Yin, H. Schütze, B. Xiang, et al., "ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs", in Transactions of the Association for Computational Linguistics, 2016, pp. 259-272.
[25]. R. H. Li, L. L. Cheng, D. P. Wang D P, et al., "Siamese BERT Architecture Model with attention mechanism for Textual Semantic Similarity", in Multimedia Tools and Applications, vol. 82, no. 30, May. 2023, pp. 1-22.
[26]. G. Fradelos, I. Perikos, I. Hatzilygeroudis, "Using Siamese BiLSTM Models for Identifying Text Semantic Similarity", in AIAI 2023 IFIP WG 12.5 International Workshops, Jun. 2023, pp. 381-392.
[27]. J. Devlin, M. W. Chang, K. Lee, et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", in the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Jun. 2019, pp. 4171–4186.
[28]. Y. Sun, S. Wang, S. Feng, et al., "ERNIE 3.0: Large-scale Knowledge Enhanced Pre-training for Language Understanding and Generation", in ArXiv, vol. abs/2107.02137, Jul. 2021, pp. 1-22.