ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230426

FBC: Fusing Bi-Encoder and Cross-Encoder for Long-Form Text Matching

Jianbo Liao[a;†], **Mingyi Jia**[a;†], **Junwen Duan**[a;*] **and Jianxin Wang**[a]

[a]Hunan Provincial Key Lab on Bioinformatics, School of Computer Science and Engineering, Central South University, China
{jbliao, jiamingyi, jwduan}@csu.edu.cn, jxwang@mail.csu.edu.cn

Abstract. Semantic text matching has a wide range of applications in natural language processing. Recently proposed models that have achieved excellent results on short text matching tasks are not well suited to long-form text matching problems due to input length limitations and increased noise. On the other hand, long-form texts contain a large amount of information at different granularities after encoding, which cannot be fully interacted and utilized by existing methods. To address above issues, we propose a novel long-form text-matching framework which **f**uses **B**i-Encoder and **C**ross-Encoder (**FBC**). Specially, it first employs an entity-driven key sentence extraction method to obtain the crucial content of the text and filter out noise. Subsequently, it integrates Bi-Encoder and Cross-Encoder to better capture semantic features and matching signals. Extensive experiments on several publicly available datasets demonstrate the effectiveness of our approach, compared with strong baselines. Furthermore, our model exhibits greater stability and accuracy in determining the matching relationship between documents describing the same event, which outperforms previously established approaches. [1]

1 Introduction

Semantic text matching techniques can be applied to a wide variety of applications, including community question answering [35], information retrieval [11], and dialogue [21]. Many deep text matching models have been proposed and gain improvement. Mainstream methods include representation based methods such as DSSM [12] and CDSSM [30], interaction based models such as DRMM [8], MVLSTM [32], K-NRM [37], Conv-KNRM [5], Match-Pyramid [25], Match-SRNN [33], PACRR [13], and their combinations like RE2 [39] and DUET [22].

Pre-trained language models (PLM) have recently been used to facilitate the development of text semantic matching due to their powerful ability to learn representations [7, 19, 17], and have reached state-of-the-art performance when it comes to general semantic matching benchmarks [34]. However, PLM models are not well suited to long-form text matching problems due to the excessive length and increased noise. BERT, for example, can handle documents with a length of less than 512 tokens. Thus in practice, documents exceeding this length will be truncated, which may result in the loss of crucial information. On the other hand, the significant amount of noise present in long documents can greatly interfere with the model's ability to make accurate predictions. Take Figure 1 as an example, though both texts describe the same news that "*Ke Jie had a Go match with the AlphaGo*", the model outputs a wrong prediction. The aforementioned phenomenon may be attributed to the truncation of some crucial sentences in $\mathbf{doc_1}$ and $\mathbf{doc_2}$, or the retention of irrelevant sentences containing extraneous information. Additionally, our analysis of existing long-form text matching methods suggests that they are prone to error when identifying text pairs that describe the same event due to inadequate semantic interactions among the pairs.

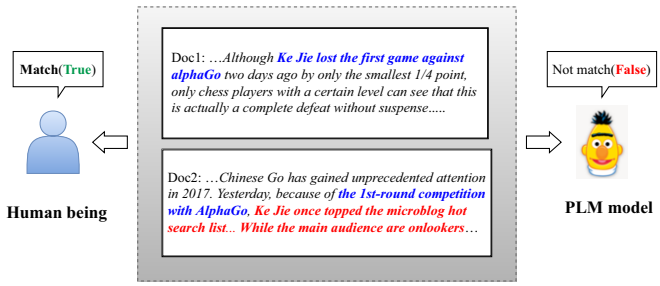

Figure 1: A mismatched example of long-form text matching. Although humans can easily judge whether two paragraphs match, the PLM will make wrong predictions, which may be because it retained some noise (the sentences marked in red in the figure) or the loss of critical information that contributes to model's accurate prediction (the sentences marked in blue) due to the limitation of input length.

On the other hand, the common paradigms for text matching are generally divided into Bi-Encoder and Cross-Encoder (Figure.2). Bi-Encoder encodes text A and B respectively, generating corresponding embeddings u and v, which are then compared for similarity to determine the matching degree. But since the two texts are encoded separately and lack interactions, the accuracy is not ideal. Cross-Encoder encodes text pairs as a whole and uses the output directly for similarity calculation, which has poor interpretability and impacts the matching speed.

To address the above issues, we propose a novel framework that **f**use **B**i-Encoder and **C**ross-Encoder for long-form text matching (FBC). We further propose an entity-driven key sentence extraction method, which extracts key sentences ranked by the entity score from the lengthy document. Specifically, FBC first takes the titles and key

* Corresponding Author. Email: jwduan@csu.edu.cn.
† These authors contributed equally to this work.
[1] The code is released at https://github.com/CSU-NLP-Group/FBC

sentences of two documents as input, and outputs the encoded vectors for each input. The encoded title vector pair and text vector pair are input into two shared Bi-Encoder networks. Finally, we concatenate the global matching vector, i.e. the encoded vector of `[CLS]` token, with the title matching vector and text matching vector from the feature matching layer, and pass them to the classification layer to get the final matching results.

The primary contributions of this paper are as follows: (i) We propose an entity-driven key sentence extraction method that effectively utilizes the number and type of entities contained in the sentence, as well as the position of the sentence within the text; (ii) based on (i), we propose our framework that fuses Bi-Encoder and Cross-Encoder to enhance the interaction between document pairs and capture more matching signals, thereby improving the final matching performance; (iii) Experimental results on several benchmark datasets demonstrate the superiority of our framework over several strong baselines.

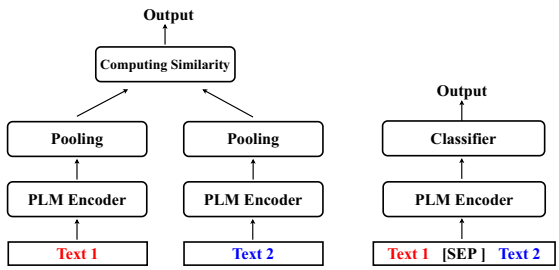

Figure 2: The architecture of Bi-Encoder and Cross-Encoder.

2 Related Work

In this section, we first discuss common short-form text matching models. Then as an extension, we will further summarize the more effective long-form text matching models currently proposed and discuss their limitations.

2.1 *Short-form Text Matching*

Current mainstream text matching methods include representative methods, interactive methods, and their combinations. These categories of methods have been widely applied to short-text matching.

Typical representation-based matching approaches include DSSM [12], C-DSSM [30], ARC-I [9], RNN-LSTM [23] and MV-LSTM [32], they usually encode text pairs into high-dimensional semantic vectors using deep neural networks, and then judge whether the two texts match by computing the similarity between their corresponding vectors. DSSM is the progenitor of representation-based matching models, featuring a simple yet high-effect structure composed of word-hash and multi-layer MLP and softmax. Subsequent variants, such as C-DSSM, are optimized based on it. ARC-I model utilizes a repetitive stacking of "CNN + k-pooling" to capture semantic correlation between adjacent n-grams, and eventually obtains sentence similarity from MLP.

Unlike representation-based approaches, the interactive approach does not directly learn semantic representation vectors for text pairs, but allows them to interact at the bottom layer, thus establishing some fundamental matching signals. Representative examples of such methods include ARC-II [10] and MatchPyramid [25] which focuses on the associations between words and makes creative reference to the principles employed by CNNs in image processing, utilizing a convolutional approach to manage interactions between two sequences of texts.

As for the combination of the aforementioned two types, RE2 [39] emphasizes three key features, namely previous alignment features, original point-wise directional features, and context-aware alignment between sequences, while simplifying most other components. This setting renders it fast, powerful and applicable to a wide range of correlative applications.

However, although the above methods perform well on short text matching tasks, they cannot be directly transferred to long text matching tasks since their patterns cannot process long-form text's massive information.

2.2 *Long-form Text Matching*

Early long-text matching tasks mostly employed term-based methods, such as TF-IDF which is based on bag-of-words models or LDA which leverages topic overlap. These methods typically viewed texts as combinations of multiple groups of words without considering the order they appeared in the text. With the advent of deep learning, researchers began employing deep learning methods to tackle long-text matching tasks. SMASH RNN [15] was the first to propose a hierarchical RNN model under a siamese architecture, integrating information from different granularities such as paragraphs, sentences, and words. Then in 2019, the Concept Interaction Graph (CIG) [18] employed graph-based methods to represent document information, and utilized graph convolutional networks to convert the matching task into a binary classification task.

The advent of powerful pre-trained models has changed the landscape of this field. Many transformer-based models, such as match-BERT and sentenceBERT [28], have achieved very impressive performances on long-text matching tasks. Considering the input length limit of BERT models (512), SMITH [38] proposed a two-layer transformers architecture to process sentence-level and word-level features, thereby breaking this limit. CIG [18] mentioned before has also incorporated the BERT architecture into their own model and achieved superior performance than the standard BERT-finetuning model on certain text-matching tasks. In 2021, Match-Ignition [24] integrated the page-rank algorithm into the transformers model to effectively filter out noise of different granularities, and achieved the current best performance on the CNSS and CNSE datasets.

It has been demonstrated in most previous works, especially those transformer-based approaches, that long-form text provides an abundance of information for matching. That is to say, learning an embedding that embodies the context of an entire document can be quite challenging. Current solutions for addressing these limitations include various pooling techniques [28] and contrastive learning methods [14]. These previous works provide strong evidence that our framework can be effective for long-form text matching tasks, especially for improving the ability to identify a pair of texts that describe the same event.

3 Our Framework

3.1 *Problem Definition*

Given the training set $S_{train} = \{d_1^i, d_2^i, y^i\}_{i=1}^N$, where d_1^i and d_2^i denote the long-form documents pair in i-th sample, and $y^i \in \{0, 1\}$ denote the corresponding sample label indicating whether the two texts in this sample match or not [4]. The target of this task is to train

a model $f : f(d_1^i, d_2^i) \Rightarrow \{0, 1\}$, where 1 indicates that the pair of documents is mutually matching, and 0 indicates the opposite.

3.2 Model Architecture

The high-level overview of our framework is shown in Figure. 3. In this section, we will separately introduce the implementation of our novel key sentences extracting method and the details of three components of our FBC model, namely the Cross-Encoder, Bi-Encoder, and the final classification layer.

Given the two input documents d_1 and d_2, we first extract their key sentences $c_1 = [s_1^1, s_1^2, ..., s_1^n]$ and $c_2 = [s_2^1, s_2^2, ...s_2^m]$ respectively through our key sentence extraction module. Then, we take the titles t_1, t_2 and key sentences of the two documents as inputs to the Cross-Encoder to obtain high-dimensional vector representations of the two documents, i.e., h_{c_1}, h_{t_1} and h_{c_2}, h_{t_2}. After completing the preliminary encoding, we feed the encoded vectors into the Bi-Encoder for matching feature extraction vector h_{f_c} and h_{f_t}. Finally, the output of the Bi-Encoder will be concatenated with the vector representation of the `[CLS]` token $h_{\texttt{[cls]}}$ from the Cross-Encoder and then fed into the classification layer to get the final result.

3.2.1 Entity-driven Key Sentence Extraction

Compared with short texts, long texts contain a large amount of text noise which often interferes with the matching signals between texts. At the same time, pre-trained language models require the removal of a large amount of text noise before long texts can be used as inputs due to the limitations of algorithm complexity. Therefore, before inputting text into the model, we first need to extract several sentences that contain key semantic information from long texts.

Entity information is an important textual matching signal, and specifically, the entity type and quantity in a sentence largely determine its relevance. Inspired by this, we propose an entity-driven key sentence extraction algorithm. For each text, we first segment the text into sentences based on punctuation marks, and then use the $Spacy$ library [31] to extract entity information from each sentence. $Spacy$ has trained a general $named\ entity\ recognition(NER)$ model using a pre-trained language model, which defines a wide range of entity types. In this paper, we select certain entity types from the model, as shown in Table 1.

Entity Types in *spacy*	Examples
PERSON	Alice
GPE	Beijing
DATE	April 3rd, 2016
TIME	5 hours
ORG	Microsoft
WORK_OF_ART	Mona Lisa
FAC	Eiffel Tower
EVENT	World Cup
NORP	Republican Party
PRODUCT	Bread
QUANTITY	55KG

Table 1: Description of the entity types in $Spacy$.

Then we calculate the keyness score of each sentence based on the distribution of entities in the sentence. Since entity information is an important matching signal in the text, the importance of a sentence can be determined by the number of entities it contains, as described in Eq. 1. However, using only the number of entities in a sentence has limitations. When the entities in a sentence are highly repetitive, it indicates that the matching signal in the sentence is relatively singular, and from a global perspective, the sentence may not be a key sentence. Therefore, we also use the number of entity types as a scoring criterion, as shown in Eq. 2.

$$\mathbf{entity}_{num}(s_i) = \frac{\mathbf{num}(s_i)}{\mathbf{num}(d)} \tag{1}$$

$$\mathbf{entity}_{type}(s_i) = \frac{\mathbf{type}(s_i)}{\mathbf{type}(d)} \tag{2}$$

where $\mathbf{num}(s_i)$ and $\mathbf{num}(d)$ denotes the number of entities in sentence s_i and in document d, while $\mathbf{type}(s_i)$ and $\mathbf{type}(d)$ denotes the number of entity types in s_i and d respectively.

In addition to entity information, the position of a sentence in the text is also a crucial factor in determining its importance [40]. Generally, sentences located at the beginning and end of an article have stronger summarization and contain more matching signals than those distributed in the middle of the article. Therefore, we also incorporate the position of a sentence in the text for calculating its keyness score. Eq. 3 describes the calculation of the sentence position score.

$$\mathbf{position}(s_i) = \frac{\max(i, n - i + 1)}{n} \tag{3}$$

where i is the position of sentence s_i in document d, and n is the total number of sentences of d. We obtain the final score of the sentence s_i by weighing and summing the above three score items, as shown in Eq. 4.

$$\begin{aligned}\mathbf{keyscore}\,(s_i) =& \alpha \times \mathbf{entity}_{num}\,(s_i) + \beta \times \mathbf{entity}_{type}\,(s_i) \\ &+ \gamma \times \mathbf{position}\,(s_i)\end{aligned} \tag{4}$$

The weights α, β and γ are hyper parameters controlling the weight of each scoring module. In this work, we set them to 0.6, 0.3 and 0.1, respectively, which can achieve the optimal matching effect.

3.2.2 Cross-Encoder for Text Pair Representation

In this section we describe the implementation of the Cross-Encoder module for text pair representation. We use the pretrained-model **BERT** [7] as backbone of our Cross-Encoder module.

Formally, a pair of documents d_1, d_2 can be represented as their titles T_1, T_2, their key words K_1, K_2 and their several key sentences C_1, C_2 extracted in the previous step. We then join the above text segments via special tokens to form a input instance $X =$ `[CLS]` T_1 `[SEP]` K_1 `[SEP]` C_1 `[SEP]` T_2 `[SEP]` K_2 `[SEP]` C_2 `[SEP]` with the format required by BERT model.

We then encode the input X using the Cross-Encoder module of our model to obtain a set of vector representations corresponding to all the text segments mentioned above, which includes $h_{\mathrm{T1}}, h_{\mathrm{T2}}$ for the titles of two documents, $h_{\mathrm{C1}}, h_{\mathrm{C2}}$ for key sentences, as well as the contextualized vector representation $h_{\texttt{[CLS]}}$.

$$\{h_i\}_{i=\texttt{[CLS]},T_1,T_2,C_1,C_2} = \mathbf{BERT\text{-}Encoder}(X; \theta_{be}) \tag{5}$$

where θ_{be} denotes the parameters of the BERT encoder. Note that the encoded vector of `[CLS]` token $h_{\texttt{[CLS]}}$ will be further used as a global matching signal in the final classification procedure.

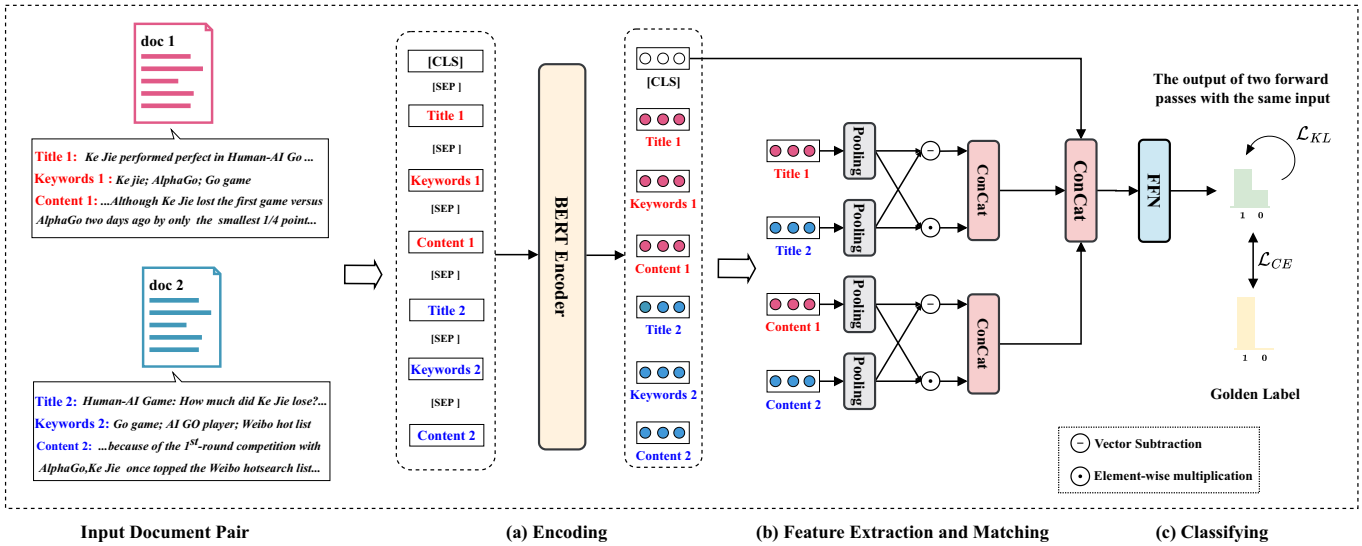

Figure 3: The overall architecture of our framework.

3.2.3 *Bi-Encoder for Matching Feature Extraction*

In this section, we describe the implementation of the Bi-Encoder module for feature extraction. Since several previous works has demonstrated the effectiveness of siamese net in matching tasks [15, 28], we decide to choose siamese net as our Bi-Encoder, to interact the features of the input vector pairs and obtain better matching features.

Specifically, given the encoded vector pairs of titles h_{T_1} and h_{T_2} and key sentences of the two documents h_{C_1} and h_{C_2}, we respectively input the title-vector-pair and content-vector-pair into two same-structured siamese nets to extract and fuse the features of the titles and contents. The siamese net also has two components, denoted as the pooling layer and the fusion layer.

The pooling layer aggregates sequential token representations into fix-length vectors for the further fusion operation. In this paper mean-pooling is adopted as the pooling layer (Eq. (6)) , since other methods like CNN and RNN are slower and do not lead to better performance.

$$\begin{aligned} h_{t_1}, h_{t_2} &= \textbf{Mean-Pooling}(h_{T_1}, h_{T_2}) \\ h_{c_1}, h_{c_2} &= \textbf{Mean-Pooling}(h_{C_1}, h_{C_2}) \end{aligned} \tag{6}$$

Given the h_t and h_c output from the pooling layer, the fusion layer then calculates the element-wise multiplication and the subtraction of these vector-pairs, and concatenates the results using Eq. 7 - 9:

$$h_{f_t} = \textbf{Fusion}(h_{t_1}, h_{t_2}) \tag{7}$$

$$h_{f_c} = \textbf{Fusion}(h_{c_1}, h_{c_2}) \tag{8}$$

$$\textbf{Fusion}(h_1, h_2) = [h_1; h_2; |h_1 - h_2|; h_1 \circ h_2] \tag{9}$$

where $\textbf{Fusion}(\cdot,\cdot)$ denotes the fusion layer, "$\circ$" denotes the element-wise multiplication (i.e., Hadamard product) of two vectors, and ";" denotes the concatenation of two vectors. Here, subtracting two vectors is intended to highlight the difference between them, while taking the Hadamard product of two vectors for illustrating their similarity. The final feature vector h_{f_t} and h_{f_c} will be propagated to the classification layer.

3.2.4 *Classification Layer*

Finally, the output of the previous module will be propagated into the classification layer along with the global matching vector of `[CLS]` (mentioned in section 3.2.2) to make the final prediction:

$$P(\hat{y} \mid d_1, d_2) = \sigma(\boldsymbol{W}\boldsymbol{h} + \boldsymbol{b}) \tag{10}$$

$$\boldsymbol{h} = [h_{\texttt{[CLS]}}; h_{f_t}; h_{f_c}] \tag{11}$$

where $\boldsymbol{W}, \boldsymbol{b}$ are learnable parameters, $\sigma(\cdot)$ denotes the sigmoid function, and $P(\hat{y} \mid d_1, d_2)$ represents the probability distribution of matching degree predicted by the classification layer from the input long-form document pair d_1, d_2 at the beginning.

3.3 *Regularized Dropout (R-Drop)*

Since we adopt dropout function in the Bi-Encoder and the final classification layer which may cause inconsistencies between training and prediction. In order to address this defect, we apply a special loss function called **R**egularized **d**ropout(R-Drop) [36] to enhance the model's robustness towards dropout. It can either be regarded as a regularization term, or can also be considered as a special form of data augmentation.

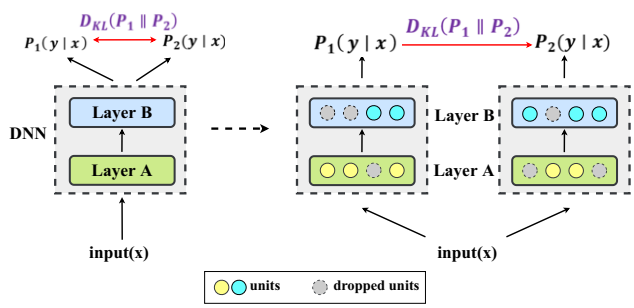

Figure 4: A example of R-drop application.

Specifically, we make each data sample repeatedly pass through the same model twice, and due to the randomness of Dropout, we can approximately assume that the sample has passed two slightly different models and gets two different outputs as $P_\theta^{(1)}(y_i \mid x_i), P_\theta^{(2)}(y_i \mid x_i)$. So the first part of our loss function is a sum of normal cross-entropy losses corresponding to each output:

$$\mathcal{L}_1 = -\log P_\theta^{(1)}(y_i \mid x_i) - \log P_\theta^{(2)}(y_i \mid x_i) \tag{12}$$

In addition, to make the two outputs as consistent as possible, we add KL divergence to the loss function as a constraint term:

$$\mathcal{L}_2 = \frac{1}{2}\left[KL\left(P_\theta^{(1)}\left(y_i \mid x_i\right) \mid\mid P_\theta^{(2)}\left(y_i \mid x_i\right)\right) + \right. \\ \left. + KL\left(P_\theta^{(2)}\left(y_i \mid x_i\right) \mid\mid P_\theta^{(1)}\left(y_i \mid x_i\right)\right)\right] \tag{13}$$

The final loss function is the weighted sum of the above two parts:

$$\mathcal{L} = \mathcal{L}_1 + \alpha\mathcal{L}_2 \tag{14}$$

where α is a hyperparameter for the strength of the regularization.

4 Experiments

4.1 Experimental Setup

Datasets. we employed multiple publicly available datasets which were categorized according to their languages into two groups:

- **Chinese Datasets**: we use two Chinese datasets, i.e. Chinese News Same Event dataset (CNSE) and the Chinese News Same Story dataset (CNSS) created by [18]. The detailed statistics of the two public datasets used in our experiments can be found in Table 2-I. Their samples are derived from news articles of Chinese major internet websites, covering multiple domains.
- **English Datasets**: For the English datasets, we use four datasets, namely AAN [27], OC [2], S2ORC [20], and PAN [26]. Among them, AAN, OC, and S2ORC are designed for citation recommendation tasks, while PAN is designed for plagiarism detection tasks. Detailed information about these datasets can be found in Table 2-II.

Category	Dataset	AvgWPerD	AvgSPerD	Train	Dev	Test
I	CNSE	982.7	20.1	17,438	5,813	5,812
	CNSS	996.6	20.4	20,102	6,701	6,700
II	AAN	122.7	4.9	106,592	13,324	13,324
	OC	190.4	7.0	240,000	30,000	30,000
	S2ORC	263.7	9.3	152,000	19,000	19,000
	PAN	1569.7	47.4	17,968	2,908	2,906

Table 2: Description of evaluation datasets, AvgWPerD denotes average number of words per document and AvgSPerD denotes the average number of sentences per document.

Implementation Details. We implemented all the models using pytorch. And we choose the fine-tuned BERT model [7] *bert-base-chinese* and *bert-base-uncased* as the encoder of our model. We use Adam [16] as our optimizer for the training, and an exponentially decaying learning rate with a linear warmup. We set the batch size to 8. The learning rate are set as 1e-5.

Evaluation metrics. As the text matching task can be viewed as a binary classification task, following previous work [18, 24], we adopt accuracy and F1 scores as evaluation metrics. We train our model for a total of 10 epochs and save the model with the best performance on the validation set, and finally evaluate on the test dataset.

4.2 Baseline Models

Follow the previous work [18, 24], we compare with the following baselines: term-based methods, traditional deep learning methods and transformers-based models. We implement two of the most important baseline methods, CIG [18] and Match-Ignition [24] under same data split, and the experimental results are consistent with those reported in their papers. Therefore, for all the baselines we directly adopt the results from [18, 24].

		CNSE Dataset		CNSS Dataset	
	Model	**Acc**	**F1**	**Acc**	**F1**
I	BM25 [29]	69.63	66.60	67.77	70.40
	LDA [3]	63.81	62.44	67.77	70.40
	SimNet [18]	71.05	69.26	70.78	74.50
II	ARC-I [9]	53.84	48.68	50.10	66.58
	DSSM [12]	58.08	64.68	61.09	70.58
	C-DSSM [30]	60.17	48.57	52.96	56.75
III	ARC-II [10]	54.37	36.77	52.00	53.83
	MatchPyramid [25]	66.36	54.01	54.01	62.52
	BERT-Finetuning [7]	81.30	79.20	86.64	87.08
	Match-Ignition [24]	86.32	84.55	91.28	91.39
IV	DUET [22]	55.63	51.94	52.33	60.67
	RE2 [39]	80.59	78.27	84.84	85.28
	CIG-Siam-GCN [18]	74.58	73.69	78.91	80.72
	CIG-S&Siam-GCN [18]	84.64	82.75	89.77	90.07
	CIG-S&Siam-GCN-S^g [18]	84.21	82.46	90.03	90.29
V	FBC (ours)	**86.96**	**85.35**	**93.39**	**93.60**

Table 3: Experimental results on CNSE and CNSS datasets.

- **Term-based Methods**: This part of baselines include BM25 [29], LDA [3] and SimNet [18]. Both BM25 (best match 25) and SimNet are based on the bag-of-words model, using text similarity to determine whether two texts match.
- **Representation-based Methods**: In the representation-based models, we selected DSSM [12], C-DSSM [30], and ARC-I [9] as baselines. They all convert text into representation vectors to achieve text matching, and the difference mainly lies in the neural network used to encode text.
- **Interaction-based Methods**: For the interaction-based models, we selected ARC-II [10], MatchPyramid [25], BERT [7], and Match-Ignition [24] as baselines. Match-Ignition plugs PageRank algorithm into Transformers model to filter out noise at different granularities, achieving the state-of-the-art performance on the CNSE and CNSS datasets.
- **Hybrid Methods**: This category of models combine both representation-based and interaction-based methods. Among them, CIG [18] proposes a novel approach that abstracts text pairs into a graph structure by introducing "concept" nodes, extracts concept-level matching vectors using a siamese network architecture based on representation learning, and then fully interacts these matching vectors using graph convolutional neural networks.

In addition to the aforementioned methods, we also introduced several hierarchical approaches, such as GRU-HAN [41] and BERT-HAN [41]; as well as two methods using pre-trained models specifically designed for long texts, i.e., Longformer [1] and TransformerXL [6], as baselines for comparison on four English datasets.

4.3 Results and Analysis

In this section, we first show the performance of our approch and a series of baselines on the datasets we use. Furthermore, We will carefully analyze the factors that affects the performance of our model on CNSE and CNSS datasets, due to the limited prior work on these two Chinese datasets, this paper focuses on studying them in depth.

4.3.1 Model performance

The main experimental results of FBC along with baseline models are presented in Table 3 and Table 4, our model achieves the best

		AAN Dataset		OC Dataset		S2ORC Dataset		PAN Dataset	
	Model	**Acc**	**F1**	**Acc**	**F1**	**Acc**	**F1**	**Acc**	**F1**
I	BM25 [29]	67.60	68.00	80.32	80.38	76.47	76.53	61.59	62.47
II	RE2 [39]	87.81	88.04	94.53	94.57	95.27	95.34	61.97	58.30
	BERT-Finetune [7]	88.10	88.02	94.87	94.87	96.32	96.29	59.11	69.66
III	GRU-HAN [41]	68.01	67.23	84.46	82.26	82.36	83.28	75.70	75.88
	GRU-HAN-CDA [41]	75.08	75.18	89.79	89.92	91.59	91.61	75.77	76.71
	BERT-HAN [41]	73.36	73.51	86.31	86.81	90.67	90.76	87.57	87.36
	BERT-HAN-CDA [41]	82.03	82.08	90.60	90.81	91.92	92.07	86.23	86.19
IV	TransformerXL-Finetune [6]	83.85	83.24	91.61	91.79	92.50	92.39	58.25	69.07
	Longformer-Finetune [1]	88.06	88.41	94.76	94.74	96.31	96.29	56.61	69.74
V	Match-Ignition [24]	89.62	89.64	95.70	95.71	96.97	96.97	89.37	89.42
VI	**FBC**(ours)	**90.71**	**90.77**	**95.75**	**95.72**	**97.06**	**97.05**	**90.02**	**89.61**

Table 4: Experimental results on four English datasets, i.e., AAN, OC, S2ORC and PAN dataset.

performance on all the six datasets over all baseline methods. We list the observations as follows:

We first compared our model with BERT-Finetune [7], as PLM-based models greatly improve the quality of text encoding compared to traditional term-based methods and early representation methods. From Table 3-III and Table 4-II, it struggles with long-form text tasks (e.g. PAN) while our model demonstrates significant superiority over it.

Next, we compared our model with models based on hierarchical attention mechanisms like GRU-HAN [41] and BERT-HAN [41], and those PLM models specifically tuned for long-form texts like Longformer [1] and transformerXL [6]. Our model outperformed these methods on all four English datasets as shown in Table 4-III and IV, indicating that our designed key sentence extraction module effectively filtered out a large amount of noise.

We finally compare our model with Match-Ignition [24], which introduces text noise filtering mechanisms of different granularities to tackle the noise problem. The comparison between FBC and Match-Ignition demonstrates the superiority of our model. FBC employs an entity-driven key sentence extraction method, which also effectively addresses the noise issue in long text matching. On the other hand, it integrates two encoding methods, Cross-Encoder and Bi-Encoder, allowing for more comprehensive document interactions and capturing more matching signals, thus improving the final matching performance.

4.3.2 *Analysis on Key Sentence Extraction*

The quality of key sentence extraction can significantly impact model performance, thus we investigated the effectiveness of different key phrase extraction methods. The results are shown in Table 5.

	CNSE Dataset		CNSS Dataset	
Model	**Acc**	**F1**	**Acc**	**F1**
TextRank + Match-Ignition [24]	86.32	84.55	91.28	91.39
TF-IDF + Match-Ignition [24]	85.56	83.58	91.86	92.05
Entity + Match-Ignition [24]	86.46	84.86	91.52	91.55
TextRank + FBC	86.20	84.34	92.00	93.08
TF-IDF + FBC	86.14	84.45	92.43	92.66
Entity + FBC (ours)	**86.96**	**85.35**	**93.39**	**93.60**

Table 5: Results of different key sentences extracting methods with Match-Ignition and our model on CNSE and CNSS datasets.

Table 5 shows that the entity-driven algorithm outperforms TF-IDF and TextRank algorithms in both models, providing strong evidence for the superiority of the entity-driven key sentence extraction algorithm. This also suggests that entity information in sentences plays a greater role in determining sentence relevance compared to word frequency information. Furthermore, under same key sentence extraction method, FBC performs better than Match-Ignition, which indirectly reflects the superiority of our model.

4.3.3 *Analysis on pooling layer in Bi-Encoder*

In this paper we employ a pooling layer in our Bi-Encoder to separately capture title matching features and text content matching features. We conduct three sets of experiments using **Mean-Pooling**, convolutional neural network (**CNN + max-pooling**), and recurrent neural network (**Bi-LSTM**), in order to verify the impact of different pooling methods on model matching performance. The experimental results are shown in Table 6.

	CNSE Dataset		CNSS Dataset	
Types of pooling layer	**Acc**	**F1**	**Acc**	**F1**
Mean-Pooling	**86.96**	**85.35**	**93.39**	**93.60**
CNN-Pooling	86.01	84.51	91.74	92.03
RNN-Pooling	85.32	83.64	91.36	91.57

Table 6: Results of different pooling layer on CNSE and CNSS datasets.

It can be observed that the simple mean pooling achieves the best results. This implies that the text features obtained after Cross-Encoder are already sufficient to express the semantic information of the text, and that simple mean pooling can be used to construct local semantic features effectively. Although classical network models such as CNN and RNN can also encode local text semantics well, the greater difference between the feature vectors of the encoded text and the global vectors obtained through Cross-Encoder limits the improvement in matching performance of the model.

4.3.4 *Model Stability Analysis*

Due to the sparsity of matching signals between long documents, current methods for matching long texts are more prone to errors when predicting the matching relationship between two documents describing the same event. Therefore, this subsection mainly analyzes the prediction performance of our FBC model on document pairs describing the same event and different events, compared to Match-Ignition. Experimental results are shown in Table 7, where **Acc_pos** represents the prediction accuracy of the model on document pairs describing the same event and **Acc_neg** represents the opposite.

	CNSE Dataset		CNSS Dataset	
Model	**Acc_pos**	**Acc_neg**	**Acc_pos**	**Acc_neg**
Match-Ignition [24]	79.91	90.23	91.19	92.59
FBC(ours)	**85.36**	**88.17**	**93.16**	**92.89**

Table 7: Model Stability Analysis of Match-Ignition and FBC (ours) on CNSE and CNSS datasets.

From Table 7, the prediction accuracy of Match-Ignition declines sharply when facing document pairs describing the same event, especially on the CNSE dataset. This phenomenon may be caused by various factors, such as annotation errors in the dataset, considerable variation in document structure, or overly long extraction length of key sentences by the model. On the contrary, our FBC model's accuracy for predicting documents describing the same event does not decrease significantly. This indicates that compared with Match-Ignition, the FBC model not only has better performance but also has better stability.

4.3.5 Ablation study

To demonstrate the effect of each module, we present an ablation study by removing each module from the framework. We conducted two groups of experiments. In the first group, we mainly validated the role of Key sentence extraction module, Bi-Encoder and Cross-Encoder, and the experimental results are shown in Table 8-**II**, where:

- "*w/o*-KSE": remove the **K**ey **S**entence **E**xtraction module and directly input the original text into the model for training and prediction.
- "*w/o*-Bi-Encoder": remove the Bi-Encoder layer, and the global matching vector obtained from the Cross-Encoder layer will be used directly to obtain the matching result through the classifier.
- "*w/o*-Cross-Encoder": remove the Cross-Encoder layer, and the Bi-Encoder will be directly used to construct the matching vector based on the title and text content.

As shown in the model architecture presented in Figure 3, the classification layer of the FBC model takes three matching vectors as inputs, namely, the global matching vector obtained from the Cross-Encoder layer and the title-text content matching vectors obtained through vector Hadamard product and vector difference in the Bi-Encoder matching layer. Additionally, to enhance the robustness of the model, the R-Drop function is employed in the loss function. So in the second group, we designed the following experiments to verify the effectiveness of these four modules, and the experimental results are shown in Table 8-**III**, where:

- "*w/o*-rdrop": remove the r-drop regularization and use cross entropy loss function.
- "*w/o*-cls": drop the global feature vector `[CLS]` and only use the two feature vectors output from the Bi-Encoder for classification.
- "*w/o*-mul": remove the Hadamard product operation and only keep the subtraction operation.
- "*w/o*-sub": contrary to "*w/o*-mul", remove the subtraction operation and only keep the Hadamard product operation in the fusion layer of Bi-encoder.

As shown in Table 8-**II**, removing key sentence extraction led to a decrease in model performance on both datasets. This demonstrates that our key phrase extraction module is beneficial in addressing the input length limitations of PLMs, while also being capable of filtering out noise and extracting crucial information from long-form texts. Besides, it can be observed that the removal of the Cross-Encoder layer results in a sharp decrease in model performance, with accuracy and F1 score both dropping below 80%. While the removal of the Bi-Encoder matching layer only makes a slight model performance decrease. This is mainly because the Cross-Encoder layer provides a more comprehensive interaction between texts compared to the Bi-Encoder, allowing for more fine-grained matching signals between texts, and thus having a greater impact on the matching results. In our FBC model, the Bi-Encoder primarily serves to enrich the matching signals and can be considered as a model optimization layer, thus having a smaller impact on the matching results.

		CNSE Dataset		CNSS Dataset	
	Model	**Acc**	**F1**	**Acc**	**F1**
I	Full-Model	**86.96**	**85.35**	**93.39**	**93.60**
II	*w/o*-KSE	85.39	84.10	92.58	92.13
	w/o-Bi-Encoder	84.76	83.26	92.26	92.01
	w/o-Cross-Encoder	78.92	77.00	77.38	78.63
III	*w/o*-rdrop	85.82	84.02	92.45	92.63
	w/o-cls	85.30	83.41	92.91	93.07
	w/o-mul	85.17	84.34	92.20	92.32
	w/o-sub	85.10	84.65	91.65	91.70

Table 8: Ablation study results on the CNSE and CNSS datasets.

From Table 8-**III**, it can be concluded that using R-Drop loss function can improve model performance. Besides, the global matching vector (`[CLS]`) generated by the encoding layer and the title-text content matching vectors obtained through vector Hadamard product (**mul**) and vector subtraction (**sub**) in the matching layer all contribute to the model performance to varying degrees. This indicates that these three matching vectors contain matching information at different levels, and concatenating them leads to a feature vector with more matching signals. This result is also intuitive, as the global features generated by the encoding layer are based on the interactions between words and contain word-level matching information, while the feature vectors generated by the matching layer using the title and text content are based on local matching signals and contain coarser-grained matching information.

5 Conclusion

We presented a novel framework in this paper for long-form text matching that incorporates both Bi-Encoder and Cross-Encoder, enabling more semantic interactions between the documents to compare. Furthermore, we present an entity-driven key sentence extraction method to improve the efficacy of text matching. We have conducted extensive experiments with several benchmark datasets to assess the effectiveness of this method. The results of our experiments indicate that our method achieves the best performance and is more robust on match document pairs that describe the same event.

Acknowledgements

We thank the anoymous reviewers for their helpful comments. This work was supported in part by the National Key Research and Development Program of China (No.2021YFF1201200), the Natural Science Foundation of China (No.62006251) and the Natural Science Foundation of Hunan Province (No.2021JJ40783), the Science and Technology Major Project of Changsha (No.kh2202004). This work was carried out in part using computing resources at the High-Performance Computing Center of Central South University.

References

[1] Iz Beltagy, Matthew E. Peters, and Arman Cohan, 'Longformer: The long-document transformer', *arXiv: Computation and Language*, (2020).

[2] Chandra Bhagavatula, Sergey Feldman, Russell Power, and Waleed Ammar, 'Content-based citation recommendation', *north american chapter of the association for computational linguistics*, (2018).

[3] David M Blei, Andrew Y Ng, and Michael I Jordan, 'Latent dirichlet allocation', *Journal of machine Learning research*, **3**(Jan), 993–1022, (2003).

[4] Yaokai Cheng, Ruoyu Chen, Xiaoguang Yuan, Yuting Yang, Shan Jiang, and Bo Yang, 'Overview of long-form document matching: Survey of existing models and their challenges', in *Journal of Physics: Conference Series*, volume 2171, p. 012059. IOP Publishing, (2022).

[5] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu, 'Convolutional neural networks for soft-matching n-grams in ad-hoc search', in *Proceedings of the eleventh ACM international conference on web search and data mining*, pp. 126–134, (2018).

[6] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov, 'Transformer-xl: Attentive language models beyond a fixed-length context', *arXiv: Learning*, (2019).

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, 'Bert: Pre-training of deep bidirectional transformers for language understanding', *arXiv preprint arXiv:1810.04805*, (2018).

[8] J. Guo, Y. Fan, Q. Ai, and W. B. Croft, 'A deep relevance matching model for ad-hoc retrieval', in *Acm International*, pp. 55–64, (2016).

[9] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen, 'Convolutional neural network architectures for matching natural language sentences', *Advances in neural information processing systems*, **27**, (2014).

[10] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen, 'Convolutional neural network architectures for matching natural language sentences', *Advances in neural information processing systems*, **27**, (2014).

[11] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck, 'Learning deep structured semantic models for web search using clickthrough data', in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pp. 2333–2338, (2013).

[12] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck, 'Learning deep structured semantic models for web search using clickthrough data', in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pp. 2333–2338, (2013).

[13] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard De Melo, 'Pacrr: A position-aware neural ir model for relevance matching', *arXiv preprint arXiv:1704.03940*, (2017).

[14] Akshita Jha, Vineeth Rakesh, Jaideep Chandrashekar, Adithya Samavedhi, and Chandan K Reddy, 'Supervised contrastive learning for interpretable long-form document matching', *ACM Transactions on Knowledge Discovery from Data (TKDD)*, (2022).

[15] Jyun-Yu Jiang, Mingyang Zhang, Cheng Li, Michael Bendersky, Nadav Golbandi, and Marc Najork, 'Semantic text matching for long-form documents', in *The world wide web conference*, pp. 795–806, (2019).

[16] Diederik P Kingma and Jimmy Ba, 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980*, (2014).

[17] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut, 'Albert: A lite bert for self-supervised learning of language representations', *arXiv preprint arXiv:1909.11942*, (2019).

[18] Bang Liu, Di Niu, Haojie Wei, Jinghong Lin, Yancheng He, Kunfeng Lai, and Yu Xu, 'Matching article pairs with graphical decomposition and convolutions', *meeting of the association for computational linguistics*, (2018).

[19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov, 'Roberta: A robustly optimized bert pretraining approach', *arXiv preprint arXiv:1907.11692*, (2019).

[20] Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Dan S Weld, 'S2orc: The semantic scholar open research corpus', *arXiv preprint arXiv:1911.02782*, (2019).

[21] Zhengdong Lu and Hang Li, 'A deep architecture for matching short texts', *Advances in neural information processing systems*, **26**, (2013).

[22] Bhaskar Mitra, Fernando Diaz, and Nick Craswell, 'Learning to match using local and distributed representations of text for web search', in *Proceedings of the 26th international conference on world wide web*, pp. 1291–1299, (2017).

[23] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward, 'Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval', *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, **24**(4), 694–707, (2016).

[24] Liang Pang, Yanyan Lan, and Xueqi Cheng, 'Match-ignition: Plugging pagerank into transformer for long-form text matching', *conference on information and knowledge management*, (2021).

[25] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng, 'Text matching as image recognition', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, (2016).

[26] Martin Potthast, Matthias Hagen, Tim Gollub, Martin Tippmann, Johannes Kiesel, Paolo Rosso, Efstathios Stamatatos, and Benno Stein, 'Overview of the 5th international competition on plagiarism detection', in *CLEF Conference on Multilingual and Multimodal Information Access Evaluation*, pp. 301–331. CELCT, (2013).

[27] Dragomir R. Radev, Pradeep Muthukrishnan, Vahed Qazvinian, and Amjad Abu-Jbara, 'The acl anthology network corpus', *ELRA*, **47**, 919–944, (2009).

[28] Nils Reimers and Iryna Gurevych, 'Sentence-bert: Sentence embeddings using siamese bert-networks', *arXiv preprint arXiv:1908.10084*, (2019).

[29] Stephen Robertson, Hugo Zaragoza, et al., 'The probabilistic relevance framework: Bm25 and beyond', *Foundations and Trends® in Information Retrieval*, **3**(4), 333–389, (2009).

[30] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil, 'Learning semantic representations using convolutional neural networks for web search', in *Proceedings of the 23rd international conference on world wide web*, pp. 373–374, (2014).

[31] Bhargav Srinivasa-Desikan, *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*, Packt Publishing Ltd, 2018.

[32] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng, 'A deep architecture for semantic matching with multiple positional sentence representations', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, (2016).

[33] Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng, 'Match-srnn: Modeling the recursive matching structure with spatial rnn', *arXiv preprint arXiv:1604.04378*, (2016).

[34] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman, 'Glue: A multi-task benchmark and analysis platform for natural language understanding', *arXiv preprint arXiv:1804.07461*, (2018).

[35] Zhiguo Wang, Wael Hamza, and Radu Florian, 'Bilateral multi-perspective matching for natural language sentences', *arXiv preprint arXiv:1702.03814*, (2017).

[36] Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, Tie-Yan Liu, et al., 'R-drop: Regularized dropout for neural networks', *Advances in Neural Information Processing Systems*, **34**, 10890–10905, (2021).

[37] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power, 'End-to-end neural ad-hoc ranking with kernel pooling', in *Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval*, pp. 55–64, (2017).

[38] Liu Yang, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork, 'Beyond 512 tokens: Siamese multi-depth transformer-based hierarchical encoder for long-form document matching', *conference on information and knowledge management*, (2020).

[39] Runqi Yang, Jianhai Zhang, Xing Gao, Feng Ji, and Haiqing Chen, 'Simple and effective text matching with richer alignment features', *arXiv preprint arXiv:1908.00300*, (2019).

[40] Shunxiang Zhang, Zhaoya Hu, Guangli Zhu, Ming Jin, and Kuan-Ching Li, 'Sentiment classification model for chinese micro-blog comments based on key sentences extraction', *Soft Computing*, **25**(1), 463–476, (2021).

[41] Xuhui Zhou, Nikolaos Pappas, and Noah A. Smith, 'Multilevel text alignment with cross-document attention', *empirical methods in natural language processing*, (2020).