Legal Feature Enhanced Semantic Matching Network for Similar Case Matching

Zhilong Hong, Qifei Zhou, Rong Zhang, Weiping Li and Tong Mo
School of Software and Microelectronics
Peking University, Beijing, China
{hongzhilong, qifeizhou, rzhangpku}@pku.edu.cn, {wpli, motong}@ss.pku.edu.cn

***Abstract*—Similar case matching (SCM) aims to determine whether legal case documents are similar or not. In fact, SCM is an extension of the semantic text matching. Various deep learning models are proposed to solve the semantic text matching problems. However, the main difference between the case documents may be subtle, and the length of documents can be quite long. Moreover, the case documents are written in structural format and contain plenty of legal terms. To address these challenges, we propose a novel model in this paper. Accordingly, the legal feature vector is introduced into the semantic text matching model, and BERT is adopted as the encoding layer to capture long-range dependencies in the case documents. We conduct several experiments to evaluate the performance of our proposed model. The results show that our model outperforms other existing methods on the public dataset CAIL2019-SCM.**

***Index Terms*—Similar case matching, BERT, Attention mechanism, Law intelligence**

I. Introduction

Similar case matching (SCM) mainly aims to discriminate whether legal case documents are similar or not. Justice means that similar cases with similar facts should always result in similar verdicts. This applies to jurisdictions with common law legal systems like the United States, as well as jurisdictions with civil law legal systems like China.

To this end, judges and lawyers spend much time finding and judging similar cases for reference. With similar case matching techniques, they can narrow down the size of the candidate set and accelerate the lookup. What is more, SCM can help with other techniques, including legal judgment predictions, question and answering systems, and information retrieval systems.

In addition to the header and footer, the legal case document in China often consists of five parts as the following:

- Information of parties involved in the current lawsuit
- Records of previous lawsuits
- Fact descriptions
- Court views that include analysis of fact descriptions
- Verdicts, including the final decision and the related law articles

The fact description part contains details of the case, corresponding evidence, and the verification of evidence made by judges. When we say two cases are similar, we are focusing on the similarity between the facts of two case, because the facts contain more information than any other part. Especially for civil cases, the amount of money, the rate of interest, and the dates in the facts have a significant impact on the final decision. These factors need to be emphasized during the matching.

Cases are divided into three categories: civil case, criminal case, and administrative case. Meanwhile, each category has hundreds of causes. For example, the civil case in China has 424 tertiary causes and 367 quaternary causes. It is worth noting that cases with different causes vary in the description and structure. When two cases have different causes, they cannot be similar in any aspect. Accordingly, similar case matching can only be applied to cases with the same causes.

There is existing literature focused on calculating the similarity among texts of different cases. Brüninghaus [2] and Saravanan [16] tried to extract features from legal case documents and compare the extracted features instead of the full texts. Kumar [11] and Raghav [15] converted the document into embeddings through the vector space model and calculated the similarity of two embeddings. However, these methods cannot make use of contextual information so that they require a large amount of human efforts to work. SCM is a certain application of semantic text matching, which has many subtasks, including natural language inference (NLI), information retrieval (IR), and question answering (QA). NLI aims to judge whether a premise can be inferred from the hypothesis. Both SCM and NLI focus on the similarity of texts. The meaning of the text pair in NLI can be either related or identical, while the meaning of cases in SCM is not. In fact, cases may differ in the involved parties and facts. There are two types of deep learning framework in NLI. One type is based on the siamese network [1], where sentence pairs are encoded individually with the same encoder like CNN [8] and LSTM [14]. Then the similarity is calculated in accordance with the embedding. However, these methods have drawbacks for interactions in low-level semantics, which may lead to the loss of important information. In order to resolve this shortcoming, the matching aggregation network is proposed, where more interactions are added in the word level and the phrase level. Chen et al. [3] proposed a state-of-the-art model ESIM to capture more local information between the text pair prior to performing the global comparison. Based on the ESIM model, there are more models proposed so far [22], [26]. After Transformer [21] architecture launches, researchers conducted

The source code can be obtained from https://github.com/thesharing/lfesm.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

more investigations based on it [5], [19].

Compared with NLI, SCM is still confronted with three major challenges that make it non-trivial. These challenges can be briefly described as the following:

1) The difference between case documents may be subtle, which makes it hard to decide whether two documents are similar or not. The difference between the amount of money, the interest rate, and the count of items results in different final verdicts, which is defined by legal professionals. For example, in private lending cases, there are three situations about the interest rate. When no interest is promised while signing the lending contract, the judge does not support the claim of interest. On the other hand, when the interest rate is higher than 24%, the excessive part is not supported by the court. Finally, when the interest rate exceeds 36%, the borrower can request to return the excessive part. The cases tend to be dissimilar when the interest rates in the two cases are in different situations.
2) Challenges are originating from the long length of the documents. In fact, the majority of case documents contain more than 512 characters, which makes it hard to capture the information in the inter-sentence level.
3) The case document is written in a structured format containing a large number of legal terms. This makes it easier to capture the key point of the document. Moreover, it is necessary to utilize the legal knowledge along with the semantic information.

In order to address these issues, we propose the legal feature enhanced semantic matching network (LFESM) in this paper. We introduce legal feature attributes and combine them with the semantic text matching model. More specifically, similar to the facial recognition [17], a triplet (A, B, C) containing three cases is considered as the input. Case A is the anchor, and our aim is to compare case B and case C with A and decide which one is more similar to A. In this regard, an example is illustrated in Fig. 1. The amount of money, interest rate, evidence and collateral is common between case *A* and *B*, so it is concluded that case *B* is more similar to case *A* than case *C* is.

At the first step, LFESM extracts the legal feature attributes from the documents and converts them into one-dimensional vectors. After that, we use the BERT model as the encoder to encode three sentences individually. Then we concatenate the legal feature vector and word embeddings together. After the concatenation, the model is divided into two siamese parts, while in each part (A, B) and (A, C) interact with each other. Furthermore, we calculate the inter-attention to capture the mutual matching information and correlations between each pair. Next, we apply a Bi-LSTM to aggregate the matching information and convert them into a fixed-length vector with pooling. Finally, we feed the subtraction of two parts to the final classifier to determine the overall result.

To investigate the effectiveness of our model on handling the similar case matching task, we conduct experiments on the public dataset CAIL2019-SCM [23]. Experiment results show

The defendant Han intended to borrow *300,000 yuan* from the plaintiff Zhang and *the outsider Sun*, and got the *encumbrance* in November 2011. The owner of the encumbrance was Zhang and Sun, the amount of debt was 300,000 yuan. Later, since Sun had no funds, Han borrowed 300,000 yuan from Zhang, and they signed the *Mortgage Loan Contract* on April 28, 2012. The monthly interest rate is *2.2%*, the repayment period is *6 months*, and the *collateral* is ** building of the defendants. On the same day, Han issued a *debit receipt*… **A**

On May 19, 2012, the defendant Lu borrowed *300,000 yuan* from the plaintiff Liu due to the urgently need of funds for the project. They both agreed on the *monthly interest of 9,000 yuan* and signed the *IOU* ("I owe you"). The IOU stated that: "Today, Liu borrowed 300,000 yuan in cash. If I can't repay the debt, the house of new traffic police will be taken as the *collateral*. The monthly interest is 9,000 yuan. Lu." After that, the Lu failed to repay the money… **B**

On May 11, 2015, the defendant Zhao borrowed *6,000 yuan* from the plaintiff Liu. In April 2016, Zhao *repaid 1,000 yuan* to the plaintiff, and left 5,000 yuan unpaid. On December 22, 2016, Zhao wrote *a promissory note* to state that Liu borrowed Zhao 5,000 yuan in cash. Zhao hasn't repaid until now, and Liu sued to the court… **C**

Fig. 1. A case document triplet (A, B, C) sampled from CAIL2019-SCM. The original text is written in Simplified Chinese, and we translate them into English. Differences of feature attributes among three cases are marked with italics.

that our method can effectively make correct decisions and get the state-of-the-art score beyond other models on this dataset. The main contributions of this paper can be summarized as the following:

1) We propose a similar case matching model called LFESM, which achieves the state-of-the-art performance on the public dataset CAIL2019-SCM.
2) We adopt BERT as the encoding layer to enhance the model's ability of handling long-range dependencies in the sequence. Furthermore, we use siamese matching aggregation architecture to distinguish the similarity of triplet (A, B, C) effectively.
3) LFESM captures the key points of the documents by incorporating the regex-preprocessed legal feature attributes into the model.

II. Related Work

A. Similar Case Matching (SCM)

Similar case matching (SCM) aims to measure the similarity among legal case documents. Brüninghaus and Ashley [2] tried to extract features and attributes from the case documents and compared the extracted features instead of the full text. Saravanan et al. [16] constructed a case ontology and extracted features based on the ontology model. Kumar et al. [11] calculated the cosine similarity of all terms in the case documents with the vector space model. Raghav et al. [15] combined paragraph similarity with citation information to find the most similar case document in the candidate set.

Former studies are mainly focused on conventional methods so that they cannot utilize the context information. In fact, similar case matching is an extension of semantic text matching,

which can be applied as a foundation of diverse tasks in natural language processing (NLP), such as natural language inference, information retrieval, and question answering. Therefore, we can adopt the main ideas from related fields we mentioned above.

B. Natural Language Inference (NLI)

Natural language inference (NLI) is concerned with determining whether a natural language hypothesis h can be inferred from a premise p or not. For example, the premise "*Apple Inc. was founded by Steve Jobs.*" can be inferred from the hypothesis "*Steve Jobs established Apple Inc. in Cupertino.*". On the one hand, similar case matching has something in common with NLI. In fact, they are both concerned about the similarity in corpus and semantics. On the other hand, they are different from several aspects. More specifically, NLI requires the premise to express similar meaning as the hypothesis, while there are no identical cases with the same facts. The extent of differences determine whether two cases are similar or not.

With the development of deep learning in natural language processing, there are two types of deep learning framework for NLI. The first framework is based on siamese architecture [1]. In the architecture, sentences are encoded individually into sentence vectors in the same vector space with the same encoder. Then the final decision is made based on the similarity of the independent sentence vectors. The encoder is usually based on CNN [8], LSTM [14], or a self-attention network [18]. Siamese network shares parameters with the same encoder, which makes the model smaller and easier to train. Also, the output embedding can be used for other tasks, such as visualization, sentence clustering, and transfer learning. However, there is no explicit low-level semantic interaction in the procedure of encoding, which may lose some useful information.

The second framework is the matching aggregation framework, where more fine-grained interactions are added in smaller units like the word level and the phrase level. Yin et al. [25] introduced the comparison of word embedding and phrase embedding of different length in addition to the comparison of sentence vector. Chen et al. [3] came up with the ESIM model. It employs local inference based on attention mechanism and then composites them to capture local information between two sequences. Motivated by ESIM, Wang et al. [22] proposed BiMPM with more complex matching operations to enhance the model. Moreover, Zhang et al. [26] proposed the MFAE model based on ESIM, which fuses two sentences in eight ways to generate final multi-fusion emphasis and representations. After Transformer [21] emerges, Duan et al. [5] and Tan et al. [19] tried to adopt self-attention architectures onto NLI tasks.

C. Legal Intelligence

In recent years, many researchers have focused on applying NLP techniques in the legal area. Except for similar case matching, there are multiple fields in this regard, including legal judgement prediction (LJP) [9], [27], legal question answering [6], finding applicable law articles [12], extracting information from case documents [20] and interpreting verdicts [24]. All work above attempts to integrate existing related models and methods to the legal field and adapt them to the characteristics of law texts, so that they can better assist legal professionals. Based on the main idea of natural language inference, we propose a model to combine legal feature attributes with the semantic text matching model.

III. Method

In this section, we will first give the definition of the SCM task. Then we describe the architecture of LFESM and details of each layer. After that, we will introduce how to generate the legal feature vector. Finally, we show the output layer and the loss function of our model. The overall view of our model is shown in Fig. 2.

A. Task Definition

SCM is mainly focused on calculation of similarity among the case documents. Similar to facial recognition [17], we take a triplet (A, B, C) as the input, which contains three case documents. Case A is the anchor. Case B is positive, case C is negative, or vice versa. The reason we use triplet is to focus on the similarity between the anchor and the positive, as well as the dissimilarity between the anchor and the negative. The triplet is denoted as $\mathbf{a} = \{a_1, a_2, \ldots, a_{\ell_a}\}$, $\mathbf{b} = \{b_1, b_2, \ldots, b_{\ell_b}\}$ and $\mathbf{c} = \{c_1, c_2, \ldots, c_{\ell_c}\}$. The a_i, b_j or c_k is a character in the case documents. Moreover, ℓ_a, ℓ_b and ℓ_c denote the length of $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ respectively. The goal of SCM is to predict the label $y \in \{0, 1\}$. When $y = 0$, then $sim(\mathbf{a}, \mathbf{b}) > sim(\mathbf{a}, \mathbf{c})$. Otherwise, when $y = 1$, then $sim(\mathbf{a}, \mathbf{b}) < sim(\mathbf{a}, \mathbf{c})$.

B. Model Architecture

1) Input Encoding: As we mentioned in Section I, the most important and informative part of a case document is the fact description part. Since the input of the model is the sequences consisted of characters, LFESM initially extracts legal feature attributes from the sequences and converts them into feature vectors $\mathbf{f}$. We will describe the procedure in Section III-C. After that, LFESM uses the full BERT model as the encoding layer to generate the embeddings of the input sequence.

It should be indicated that BERT [4] is a multilayer bidirectional Transformers encoder, where the Transformers [21] are stacked layer by layer. The Transformer is formed by stacked layers with self-attention layers and point-wised, fully connected layers. Without recurrence used in RNN and LSTM, Transformer can still capture the long-range dependence existing in the text more effectively with attention mechanism and positional encoding.

Here BERT learns word representation and its corresponding context as the following:

$$\bar{\mathbf{a}}_i = \mathrm{BERT}(\mathbf{a}, i), \forall i \in [1, \ldots, \ell_a] \tag{1}$$

$$\bar{\mathbf{b}}_j = \mathrm{BERT}(\mathbf{b}, j), \forall j \in [1, \ldots, \ell_b] \tag{2}$$

$$\bar{\mathbf{c}}_k = \mathrm{BERT}(\mathbf{c}, k), \forall i \in [1, \ldots, \ell_c] \tag{3}$$

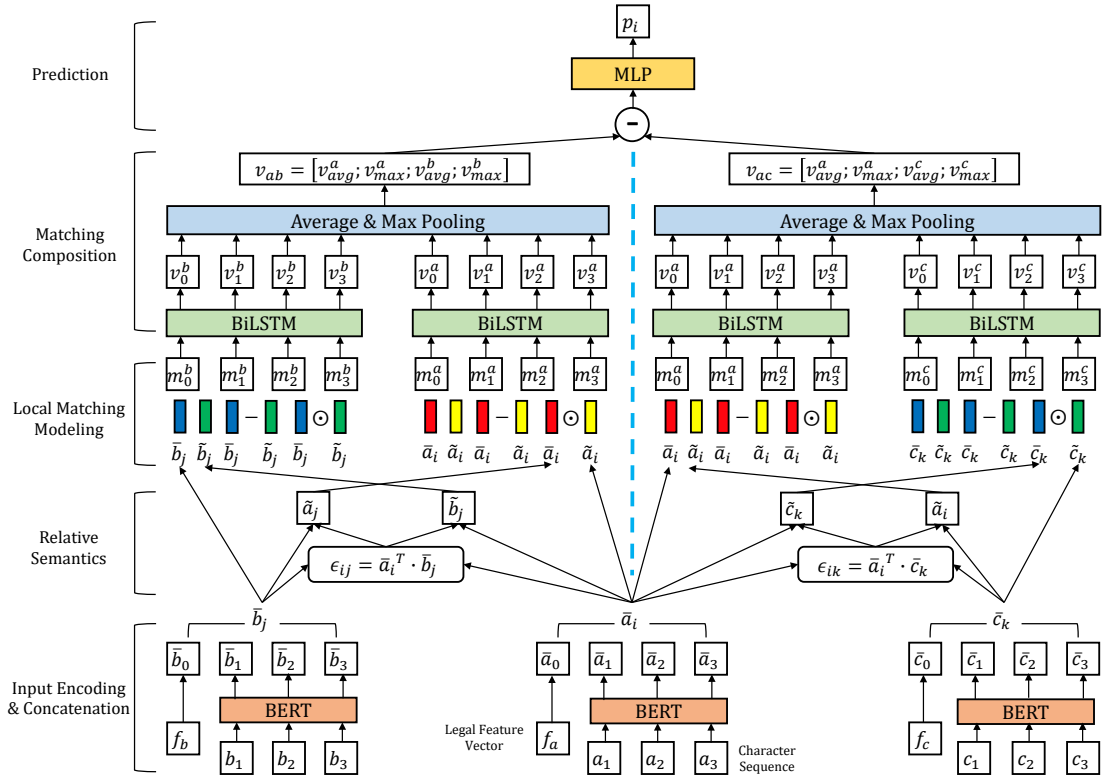

Fig. 2. Overview of LFESM.

where $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ are the case texts. Moreover, $\bar{\mathbf{a}}_i, \bar{\mathbf{b}}_j, \bar{\mathbf{c}}_k \in \mathbb{R}^h$ are the hidden state vectors of each character generated by the last Transformer layer of BERT.

2) Concatenation of Inputs: After the input sequence is encoded, we need to concatenate the feature vector $\mathbf{f}_a$, $\mathbf{f}_b$ and $\mathbf{f}_c$ with the embedding vectors $\bar{\mathbf{a}}$, $\bar{\mathbf{b}}$ and $\bar{\mathbf{c}}$. Since the dimension of $\mathbf{f}_a$ is equal to the dimension of $\bar{\mathbf{a}}_i$, we just put $\mathbf{f}_a$ in front of $\bar{\mathbf{a}}_1$, so that the input vector can be rewritten as $[\mathbf{f}_a, \bar{\mathbf{a}}_1, \bar{\mathbf{a}}_2, \ldots, \bar{\mathbf{a}}_{\ell_a}]$. For convenience, we denote $\bar{\mathbf{a}}_0 = \mathbf{f}_a$. Then the concatenated vector is:

$$\bar{\mathbf{a}}_i, \forall i \in [0, \ldots, \ell_a] \tag{4}$$

It is observed that the length of the input vector is $\ell_a + 1$. The same operation can be applied to $\bar{\mathbf{b}}$ and $\bar{\mathbf{c}}$.

3) Inter-sentence Attention Weight: From here the model splits into two siamese parts. In the *left* part, $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$ interact with each other, while in the *right* part, $\bar{\mathbf{a}}$ and $\bar{\mathbf{c}}$ interact with each other. In the following sections we take the $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$ as the example.

The attention mechanism simulates people's attention when they are reading a sentence. For each embedding vector $\bar{\mathbf{a}}_i$, which denotes a character or the feature vector, inter-sentence attention in sequence $\bar{\mathbf{a}}$ is calculated to softly align the character to the contents of $\bar{\mathbf{b}}$. We denote the attention weight as ϵ_{ij}. If the absolute value of ϵ_{ij} is high, we can say $\bar{\mathbf{a}}_i$ and $\bar{\mathbf{b}}_j$ are strongly correlated with each other.

In this layer we compute the attention weights as the similarity of the input tuple $\langle \bar{\mathbf{a}}_i, \bar{\mathbf{b}}_j \rangle$ between $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$:

$$\epsilon_{ij} = \bar{\mathbf{a}}_i^\top \cdot \bar{\mathbf{b}}_j \tag{5}$$

4) Relative Semantics: After obtaining the attention weight, we can calculate the relative semantic vector using the *softmax* function. The *softmax* function will normalize the weights. In other words, prior to applying *softmax*, the value of weights could be negative or greater than one. After applying *softmax*, all the weights are mapped to the interval $(0, 1)$, and the sum of weights in one dimension is 1. The normalized attention weight is described as the following:

$$\alpha_{ij}^a = \frac{\exp(\epsilon_{ij})}{\sum_{k=0}^{\ell_b} \exp(\epsilon_{ik})} \tag{6}$$

$$\alpha_{ij}^b = \frac{\exp(\epsilon_{ij})}{\sum_{k=0}^{\ell_a} \exp(\epsilon_{kj})} \tag{7}$$

Then, the relevant semantics of $\bar{\mathbf{a}}_i$ and $\bar{\mathbf{b}}_j$ can be composed using α_{ij}^a and α_{ij}^b correspondingly:

$$\tilde{\mathbf{a}}_i = \sum_{j=0}^{\ell_b} \alpha_{ij}^a \bar{\mathbf{b}}_j, \forall i \in [0, \ldots, \ell_a] \tag{8}$$

$$\tilde{\mathbf{b}}_j = \sum_{i=0}^{\ell_a} \alpha_{ij}^b \bar{\mathbf{a}}_i, \forall j \in [0, \ldots, \ell_b] \tag{9}$$

where $\tilde{\mathbf{a}}_i$ is the weighted sum of $\{\bar{\mathbf{b}}_j\}_{j=0}^{\ell_b}$. This means the content in $\{\bar{\mathbf{b}}_j\}_{j=0}^{\ell_b}$ that is relevant to $\bar{\mathbf{a}}_i$ is selected to form

the relative semantics $\tilde{\mathbf{a}}_i$. We calculate $\tilde{\mathbf{b}}_j$ in the same way as (9) shows.

5) Local Matching Modeling: Here we get the concentrated vector $\bar{\mathbf{a}}$, $\bar{\mathbf{b}}$, as well as their relative semantic vector $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$. In this layer we will calculate the local matching information with them. First we compute the difference and the element-wise product for the pair $\langle \bar{\mathbf{a}}, \tilde{\mathbf{a}} \rangle$ and $\langle \bar{\mathbf{b}}, \tilde{\mathbf{b}} \rangle$. Next all of these vectors and results will be concatenated together:

$$\mathbf{m}^a = [\bar{\mathbf{a}}; \tilde{\mathbf{a}}; \bar{\mathbf{a}} - \tilde{\mathbf{a}}; \bar{\mathbf{a}} \odot \tilde{\mathbf{a}}] \tag{10}$$

$$\mathbf{m}^b = [\bar{\mathbf{b}}; \tilde{\mathbf{b}}; \bar{\mathbf{b}} - \tilde{\mathbf{b}}; \bar{\mathbf{b}} \odot \tilde{\mathbf{b}}] \tag{11}$$

where $\odot$ is the element-wise product between two vectors. The concatenation of $\bar{\mathbf{a}}$, $\tilde{\mathbf{a}}$, $\bar{\mathbf{a}} - \tilde{\mathbf{a}}$ and $\bar{\mathbf{a}} \odot \tilde{\mathbf{a}}$ is considered as a higher level of interaction, while the interaction in relevant semantics is word-level. In this way we can collect the local matching information and further make a composition of it.

6) Matching Composition: In the matching composition layer, we use bidirectional LSTM (BiLSTM) to compose the local matching information. LSTM [7] is a sequential encoder based on RNN. RNN treats the input as a temporal sequence. Moreover, the hidden state of the current time step is affected by the hidden state of the last time step and the current input token. LSTM employs three gates together with a memory cell to control the data flow, which makes it capable of capturing long-distance dependencies in the sequence.

The BiLSTM runs a forward and backward LSTM at the same time and captures information in both directions. Here the local matching information is composed as:

$$\mathbf{v}_i^a = \mathrm{BiLSTM}\left(f\left(\mathbf{m}^a\right), i\right), \forall i \in [0, \ldots, \ell_a] \tag{12}$$

$$\mathbf{v}_j^b = \mathrm{BiLSTM}\left(f\left(\mathbf{m}^b\right), j\right), \forall j \in [0, \ldots, \ell_b] \tag{13}$$

where $\mathbf{m}^a$ and $\mathbf{m}^b$ is the concatenated vector yielded from (10) and (11). The f is a fully connected layer with ReLU activation to reduce the overall parameter size and avoid overfitting:

$$f(\mathbf{m}^a) = \mathrm{ReLU}\left(\mathbf{W}_1 \mathbf{m}^a + \mathbf{q}_1\right) \tag{14}$$

where $\mathbf{W}_1 \in \mathbb{R}^{h \times 4h}$, and h is the dimension of the embedding vector $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$. Then we compute both average and max pooling of $\mathbf{v}_a$ and $\mathbf{v}_b$, concatenate them to form the composition vector $\mathbf{v}_{ab}$.

The average pooling is:

$$\mathbf{v}_{\mathrm{avg}}^a = \sum_{i=0}^{\ell_a} \frac{\mathbf{v}_i^a}{\ell_a}, \quad \mathbf{v}_{\mathrm{avg}}^b = \sum_{j=0}^{\ell_b} \frac{\mathbf{v}_j^b}{\ell_b} \tag{15}$$

The max polling can be mathematically expressed as:

$$\mathbf{v}_{\max}^a = \max_{0 \le i \le l_a} \mathbf{v}_i^a, \quad \mathbf{v}_{\max}^b = \max_{0 \le j \le l_b} \mathbf{v}_j^b \tag{16}$$

Moreover, the concatenation is:

$$\mathbf{v}_{ab} = \left[\mathbf{v}_{\mathrm{avg}}^a; \mathbf{v}_{\max}^a; \mathbf{v}_{\mathrm{avg}}^b; \mathbf{v}_{\max}^b\right] \tag{17}$$

From here the siamese parts are merged together. As we mentioned in Section III-B3, the model splits into two siamese parts. In the *left* part the input vector $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$ interact with each other and produce the composition vector $\mathbf{v}_{ab}$. The *right* part produces $\mathbf{v}_{ac}$. Then $\mathbf{v}_{ab}$ and $\mathbf{v}_{ac}$ is sent to the next layer.

7) Prediction: The last layer is the prediction layer. First we calculate the difference between $\mathbf{v}_{ab}$ and $\mathbf{v}_{ac}$ to encode their distance. Then we put the difference into the final multilayer perceptron (MLP) classifier and make the prediction:

$$p_i = \mathrm{MLP}\left(\mathbf{v}_{ab} - \mathbf{v}_{ac}\right) \tag{18}$$

where p_i denotes the probability of the prediction label y_i and $y_i \in \{0, 1\}$. The entire model is trained end-to-end, and the loss function we use is cross entropy:

$$\mathcal{L} = -\sum_{i=0}^{1} \left[y_i \log p_i + (1 - y_i) \log\left(1 - p_i\right)\right] \tag{19}$$

C. Legal Feature Vector

As we mentioned in Section III-B1, before the documents are sent into the encoding layer, LFESM extracts the legal feature attributes and converts them into the feature vector. In the following sections, we will describe the legal feature attributes and vectorization.

1) Legal Feature Attributes: As the difference between case documents is subtle, the difference between some attributes like the amount of money, interest rate, and count of items should be emphasized. Therefore, we need to extract a set of specific legal feature attributes according to the cause of cases. Usually, similar or related causes can share the same set of feature attributes. Here we take the cause *Private Lending* as an example. In private lending cases, there are several factors that have impacts on the final verdicts:

- The property of plaintiffs and defendants, whether they are a natural person or a legal person.
- The count of plaintiffs and defendants.
- The type of guarantee, including no guarantee, collateral, mortgage, and guarantor.
- The type of method to calculate the interest rate, including no interest, simple interest, compound interest, and others.
- The converted annual interest rate: we convert both the monthly and daily interest rate to the annual one and only take the highest value appeared in the context. Moreover, we divide the interest rate into three intervals according to the law, including $[0\%, 24\%]$, $(24\%, 36\%]$, and $(36\%, \infty)$.
- The borrowing delivery method, including no lending, cash, bill, bank transfer, transfer via mobile apps like Alipay or WeChat, and other methods.
- The repayment form, including unpaid, partial paid, and others.
- The evidence, including receipts, bank slips, contracts, repayment commitments, mortgages or collaterals, the chat history of SMS or WeChat, and the voice record.

All of these attributes consist of the combination of legal terms and numbers, which have certain patterns. Therefore, we can extract these attributes using regex repression. The attributes extracted from the documents are either numeric or nominal. For the example illustrated in Fig. 1, we can see that case *A* and case *B* have common attributes such as the interest

rate, the evidence, and the collateral. In contrast, case *A* and case *C* are different in the interest rate, the amount of money, and the evidence. Therefore, it is observed that case *B* is more similar to case *A* than case *C* is.

2) Vectorization: After extracting these attributes, we further vectorize them into the legal feature vectors. For each nominal attribute, it will be converted into vector using one-hot encoding, which means that the embedding only consists of 0 and 1. Only the bit at the specified category is 1, while other bits are all 0. For example, the converted vector of the *guarantee* attribute in the case *A* from Fig. 1 is $[1, 0, 0, 0]$ since there is no guarantee in it. Moreover, the numeric attributes will remain their original value. Then, the vectors are concatenated together. For *private lending* cases, the dimension of the concatenated vector $\mathbf{p}$ is 18. To concatenate the feature vector with the semantic word embedding we mentioned in Section III-B2, we use a fully connected layer to transform the dimension:

$$\mathbf{f}_a = \mathbf{W}_2\mathbf{p}_a + \mathbf{q}_2 \tag{20}$$

where $\mathbf{W}_2 \in \mathbb{R}^{d\times h}$, d is the dimension of $\mathbf{p}$, and h is the dimension of the vector $\bar{\mathbf{a}}$. Moreover, the same vectorization operation is applied on the attributes of *B* and *C*.

Overall, LFESM is implemented based on the combination of the semantic matching model and legal feature vectors. In the following section, we will prove the significance of enhancements in our model.

IV. Experiment

In this section, we carry out experiments on the public dataset and compare our method with several models to prove the effectiveness of our model. Also, we conduct ablation experiments to investigate the role of each enhancement in our method.

A. Data

CAIL2019-SCM is a public dataset that focuses on the similar case matching task. The dataset contains 8,138 triplets of legal documents. It is worth noting that all the documents are obtained from China Judgement Online website[1] and are correlated to private lending cases. The document in the triplets contains the fact description part, whose length is 500-800 characters. The similarity among the triplets is annotated by the legal professionals. Table I shows the amount of the dataset and its sample distribution. As we can see, the distribution of positive and negative samples is balanced.

TABLE I
THE AMOUNT OF DATA IN CAIL2019-SCM

Dataset	Positive[1]	Negative[2]	Total Amount
Train	2,596	2,506	5,102
Valid	837	663	1,500
Test	803	733	1,536

[1]Positive means $sim(\mathbf{a}, \mathbf{b}) > sim(\mathbf{a}, \mathbf{c})$.
[2]Negative means $sim(\mathbf{a}, \mathbf{b}) < sim(\mathbf{a}, \mathbf{c})$.

[1]http://wenshu.court.gov.cn/

B. Implementation Details

The training procedure of the BERT layer contains two steps: pre-training and fine-tuning. The pre-trained model is trained on the large-scale corpus and can provide parameters for downstream tasks. Fine-tuning is the extension of pre-training where learned layers are allowed to retrain or fine-tune on the downstream tasks. Since all the documents are written in Simplified Chinese and belong to the civil case, we use the pre-trained BERT model obtained from OpenCLaP [28]. The model is pre-trained on 26.54 million civil case documents. Since BERT can only handle sequences whose length is no longer than 512, we truncate the input sequence from the front. That is because the back part is more informative than the front part in the sequence.

We adopt the same hyperparameters as the BERT model. The hidden size h in the BERT layer and the BiLSTM layer is set to 768 in our model. Moreover, we apply dropout among each layer with the dropout rate at 0.1. The batch size is 3, and we trained the model for 6 epochs. The whole model is optimized using AdamW [13], which adopts weight decay instead of L_2 regularization and regularizes variables with large gradients more than L_2 regularization would. AdamW yields better training loss and generalization error than Adam does. The learning rate of the optimizer is 2e-5. In addition, we use NVIDIA Apex that enables mixed-precision training to accelerate the training procedure.

C. Results and Analysis

Table II shows the experimental results of several methods, which contain baselines and best scores obtained from CAIL2019-SCM, along with the scores of our baselines and our proposed model. Aside from the baselines in the original paper, we implemented three baselines based on the BERT, CNN, and LSTM. First, we use BERT to convert the character sequence into embedding vectors without fine-tuning. In CNN and LSTM baseline models, CNN [10] and LSTM [7] are adopted as the encoding layer to convert the embeddings to hidden states. A linear layer with *softmax* activation calculates the similarity. For all the methods, we use accuracy as our evaluation metrics.

TABLE II
EXPERIMENTAL RESULTS OF METHODS ON CAIL2019-SCM

	Method	Valid	Test
Baseline[1]	BERT	61.93	67.32
	LSTM	62.00	68.00
	CNN	62.27	69.53
Our Baseline	BERT	64.53	65.59
	LSTM	64.33	66.34
	CNN	64.73	67.25
Best Score[1]	11.2yuan	66.73	72.07
	backward	67.73	71.81
	AlphaCourt	70.07	72.66
Our Method	LFESM	**70.01**	**74.15**

[1]Obtained from CAIL2019-SCM.

As shown in Table II, our baseline performs better than theirs on the evaluation set, but not as good as theirs on the test

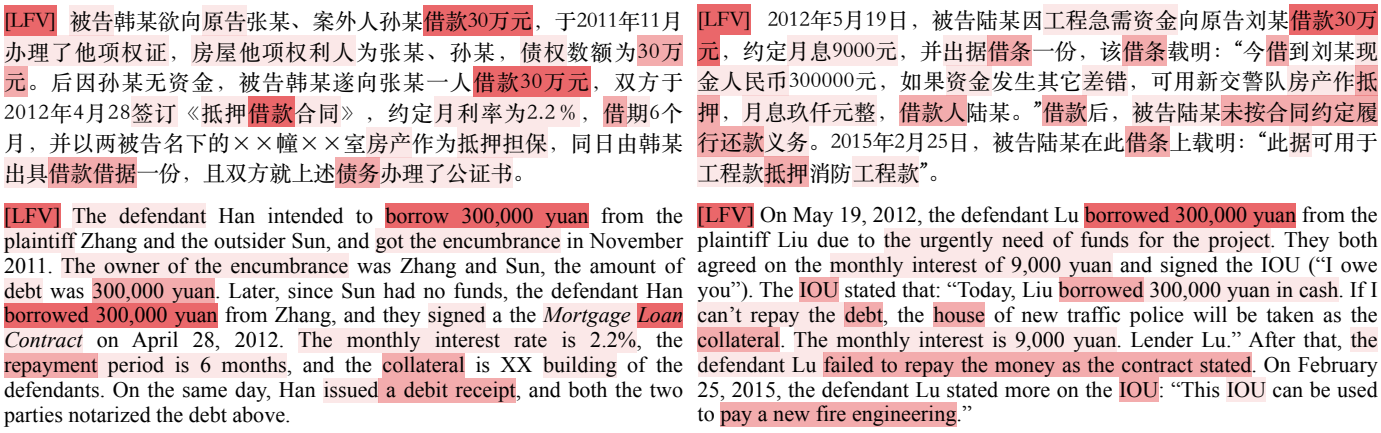

Fig. 3. Heatmap of attention weights in case *A* and case *B*. The original texts are written in Simplified Chinese, and we translate them into English. We calculate the sum of attention weights for each character and divide the sums into four categories. Words with deeper background color have higher weights.

set. This may be because we use different word embeddings and different hyperparameters from theirs. For the test set, the current best score is 72.66% achieved by *AlphaCourt*, and LFESM has improved 1.49% on the test set. Although our method is slightly behind the best score on the evaluation set, it still outperforms the second one by 2.28%. It demonstrates the effectiveness of our proposed method on the SCM task.

D. Ablation Experiment

To further explore the role of each module, we designed several ablation experiments to evaluate the performance of LFESM. First, we simply use BERT embedding and a linear layer to predict the similarity between A and B as well as A and C. Then we add the LSTM encoding layer into it. BERT embedding itself can express the semantics of each word. However, as they do not interact with the context during the training, it cannot utilize the contextual information. Therefore, the performance of the model improves when the model obtains more complicated interactions in the sequence. Due to the same reason, when we replace the BERT layer with BERT embedding, LFESM gets worse. The replacement means that we do not fine-tune the BERT model. Instead, we use the word embedding output from BERT directly.

TABLE III
RESULT OF ABLATION EXPERIMENTS

Method	Valid	Test
BERT Embedding	64.53	65.59
BERT Embedding + LSTM	64.33	66.34
LFESM w/o BERT layer[1]	66.33	69.79
LFESM w/o Siamese[2]	67.63	71.76
LFESM w/o LFV[3]	67.87	72.87
LFESM	**70.01**	**74.15**

[1]Replace the BERT layer with BERT embedding.
[2]Without siamese architecture.
[3]Without legal feature vector.

Whether to use siamese architecture or not is another key point. We tried to replace the siamese parts by changing the input from three to two and calculate the similarity of *A* and *B*, as well as *A* and *C* individually. If $sim\,(A,B) > sim\,(A,C)$, we select *B* as the result, and vice versa. If we remove the siamese architecture, the accuracy decreases 2.39%. So we can see that siamese architecture plays an important role in our method. Also, when we remove the legal feature vector (LFV), we can see the accuracy falls 1.28%. We will further illustrate the contribution of LFV in the following section.

E. Case Study

In this section, we select an example to illustrate that our method works. Fig. 3 is the heatmap of attention weights, in this figure there are two similar cases *A* and *B*. We calculate the sum of attention weights for each character and divide the sums into four categories. Words with deeper background color have higher weights. First, it worth noting that LFV has high attention weight, which means it does matter in our method. In addition, we can see words related to legal attributes have higher attention weights than others, including the amount of money and the collateral. This indicates that LFESM is focused on the words related to legal attributes more than other words.

V. Conclusion

In this paper, we concentrate on the similar case matching (SCM) task and propose the legal feature enhanced semantic matching network (LFESM) to solve it. More specifically, SCM is focused on comparing the similarity of two documents, and the documents have three major characteristics. (1) The difference between the documents is subtle. (2) The length of the document is quite long. (3) Documents are structured and contain a large number of legal terms. LFESM adopts BERT as the encoding layer to enhance the ability of handling long sequences and introduces legal feature vectors to capture the legal knowledge. Experiments show that LFESM outperforms all state-of-the-art models on the test set of CAIL2019-SCM.

In the future, we will continue to work on the following topics. (1) We will explore more ways to extract legal feature

attributes and vectorize them. (2) Since the attributes have logical connections, it is our next challenge to find a way to extract more information by leveraging the connections. (3) BERT can only handle sequence no longer than 512. At present, we simply truncate from the front because the back is more informative than the front. However, we are looking for a new way to make use of the whole sentence.

ACKNOWLEDGMENT

We appreciate the reviewers' insightful comments. This work is supported by the the National Key Research and Development Program of China (2017YFB1400401).

REFERENCES

[1] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744, 1994.

[2] Stefanie Brüninghaus and Kevin D. Ashley. Improving the Representation of Legal Case Texts with Information Extraction Methods. 2001.

[3] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced LSTM for Natural Language Inference. In *Proceedings of the 55th Annual Meeting of the ACL (Volume 1: Long Papers)*, pages 1657–1668, 2017.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the NAACL: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[5] Chaoqun Duan, Lei Cui, Xinchi Chen, Furu Wei, Conghui Zhu, and Tiejun Zhao. Attention-Fused Deep Matching Network for Natural Language Inference. In *Proceedings of the 27th International Joint Conferences on Artificial Intelligence*, pages 4033–4040, 2018.

[6] Biralatei Fawei, Jeff Z. Pan, Martin Kollingbaum, and Adam Z. Wyner. A Methodology for a Criminal Law and Procedure Ontology for Legal Question Answering. In *Joint International Semantic Technology Conference*, pages 198–214. Springer, 2018.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[8] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050, 2014.

[9] Zikun Hu, Xiang Li, Cunchao Tu, Zhiyuan Liu, and Maosong Sun. Few-shot charge prediction with discriminative legal attributes. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 487–498, 2018.

[10] Yoon Kim. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on EMNLP*, pages 1746–1751, 2014.

[11] Sushanta Kumar, P. Krishna Reddy, V. Balakista Reddy, and Aditya Singh. Similarity analysis of legal judgments. In *Proceedings of the Fourth Annual ACM Bangalore Conference*, page 17. ACM, 2011.

[12] Yi-Hung Liu, Yen-Liang Chen, and Wu-Liang Ho. Predicting associated statutes for legal problems. *Information Processing & Management*, 51(1):194–211, 2015.

[13] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, September 2018.

[14] 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 (TASLP)*, 24(4):694–707, 2016.

[15] K. Raghav, P. Krishna Reddy, and V. Balakista Reddy. Analyzing the extraction of relevant legal judgments using paragraph-level and citation information. *AI4JCArtificial Intelligence for Justice*, page 30, 2016.

[16] M. Saravanan, Balaraman Ravindran, and S. Raman. Improving legal information retrieval using an ontological framework. *Artificial Intelligence and Law*, 17(2):101–124, 2009.

[17] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[18] T. Shen, T. Zhou, G. Long, J. Jiang, S. Wang, and C. Zhang. Reinforced self-attention network: A hybrid of hard and soft attention for sequence modeling. In *International Joint Conference on Artificial Intelligence*, 2018.

[19] Chuanqi Tan, Furu Wei, Wenhui Wang, Weifeng Lv, and Ming Zhou. Multiway attention networks for modeling sentence pairs. In *Proceedings of the 27th IJCAI*, pages 4411–4417. AAAI Press, 2018.

[20] Tom Vacek and Frank Schilder. A sequence approach to case outcome detection. In *Proceedings of the 16th Edition of the International Conference on Articial Intelligence and Law*, pages 209–215. ACM, 2017.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, \Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[22] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conferences on Artificial Intelligence*, pages 4144–4150. AAAI Press, 2017.

[23] Chaojun Xiao, Haoxi Zhong, Zhipeng Guo, Cunchao Tu, Zhiyuan Liu, Maosong Sun, Tianyang Zhang, Xianpei Han, Heng Wang, and Jianfeng Xu. CAIL2019-SCM: A Dataset of Similar Case Matching in Legal Domain. *arXiv preprint arXiv:1911.08962*, 2019.

[24] Hai Ye, Xin Jiang, Zhunchen Luo, and Wenhan Chao. Interpretable Charge Predictions for Criminal Cases: Learning to Generate Court Views from Fact Descriptions. In *Proceedings of the 2018 Conference of the NAACL: Human Language Technologies, Volume 1 (Long Papers)*, pages 1854–1864, 2018.

[25] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *Transactions of the Association for Computational Linguistics*, 4(1):259–272, 2016.

[26] Rong Zhang, Qifei Zhou, Bo Wu, Weiping Li, and Tong Mo. What Do Questions Exactly Ask? MFAE: Duplicate Question Identification with Multi-Fusion Asking Emphasis. In *Proceedings of the 2020 SIAM International Conference on Data Mining*. SIAM, 2020. in press.

[27] Haoxi Zhong, Zhipeng Guo, Cunchao Tu, Chaojun Xiao, Zhiyuan Liu, and Maosong Sun. Legal judgment prediction via topological learning. In *Proceedings of the 2018 Conference on EMNLP*, pages 3540–3549, 2018.

[28] Haoxi Zhong, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Open chinese language pre-trained model zoo. Technical report, 2019.