DUTIR at the BioCreative VI Precision Medicine Track: Document Triage for Identifying PPIs Affected by Genetic Mutations

Ling Luo, Zhihao Yang*, Hongfei Lin, Jian Wang
College of Computer Science and Technology, Dalian University of Technology, Dalian, China 116024

Abstract—The precision medicine initiative promises to identify individualized treatment depending on a patients’ genetic profile and their related responses. In order to help health professionals and researchers in the precision medicine endeavor, the BioCreative VI Precision Medicine Track focuses on mining protein-protein interactions (PPIs) affected by genetic mutations from the scientific published literature. This paper describes our method used to create our submissions to the document triage subtask. In our method, several deep learning models for document triage are used, and the ensemble of them performs better than any individual model. Our best submission achieves an F-score of 0.6909 on the test set.

Keywords—Document Triage; Precision Medicine; Deep Learning; Ensemble

I. INTRODUCTION

The precision medicine initiative (PMI) promises to identify individualized treatment depending on a patients’ genetic profile and their related responses. In order to help health professionals and researchers in the precision medicine endeavor, one goal is to leverage the knowledge available in the scientific published literature and extract clinically useful information that links genes, mutations, and diseases to specialized treatments (1). Therefore, automatic text mining technique has received much attention.

Despite previous studies in protein-protein interaction (2,3) and mutation extraction (4), no one has investigated how to combine these efforts in order to help assessing and curating the clinical significance of genetic variants, an essential step towards precision medicine. Thus, the precision medicine (PM) task in BioCreative VI aims to bring together the biomedical text mining community in a new BioCreative challenge task focusing on identifying and extracting from the biomedical literature protein-protein interactions (PPIs) changed by genetic mutations. This challenge consists of two subtasks. The first subtask is Document Triage, which focuses on identifying relevant PubMed citations describing genetic mutations affecting PPIs. The second subtask is Relation Extraction. Participants in this task will be expected to build automated methods that are capable of extracting experimentally verified PPIs affected by the presence of a genetic mutation.

For the challenge, we participated in the first subtask (Document Triage Task) and our submissions to the subtask are created by the deep learning models. We presented an ensemble system that combines the results from five individual neural network models to identify relevant articles describing genetic mutations affecting PPIs from biomedical literature. The overview of our system architecture is shown in Fig. 1. Firstly, some preprocessing steps including text sentence splitting, tokenization and lowercasing are performed. Secondly, a word embedding is learned with large amounts of unlabeled data with the fastText tool (5). Moreover, the additional feature embeddings (POS and NER embedding) are introduced into the model. Then with the embeddings as input, five models are trained by the annotated training set. Finally, the results from the five models are combined by the weighted majority voting. The process is described in details in the following sections.

II. FEATURES

Currently word embedding is widely used in the field of NLP, especially based on the deep learning methods (6). In our method, we used it as the features of our baselines. Moreover, to investigate the effects of other features (such as part of speech (POS) and named entity recognition (NER)), these features are added into the baselines as additional features. All feature embeddings are parameters of the neural network model, and they can be optimized when the model is trained. Details of each of features are presented as follows.

A. Word Embedding

Word embedding, also known as distributed word representation, can capture both the semantic and syntactic information of words from a large unlabeled corpus and has attracted considerable attention from many researchers. Compared with the bag-of-words (BOW) representation, word embedding is low-dimensional and dense. In recent years, several tools, such as word2vec (7) and fastText (5), have been widely used in the field of NLP. To achieve a high-quality word embedding, we downloaded a total of 1,322,107 MEDLINE abstracts from the PubMed website with the query string “protein” as the unlabeled data. Then the data and the training dataset (a total of 4,082 abstracts) provided in the PM document triage task were used to train 50-dimensional word embedding by the fastText tool as pre-trained word embedding.
B. Additional Features

Due to the complexity of the natural language, some linguistic features are often employed in traditional machine learning methods. We also explored the effect of additional features (such as POS and NER). The POS information of each word were generated by the Stanford CoreNLP tool (8). In addition, NER tags information (including gene, chemical, disease and mutation entities) generated by the PubTator tagger (9) was also used as a feature. The dimensions of the POS and NER embedding are both five and they were initialized randomly.

III. DESCRIPTION OF THE MODELS

In this section we describe in details the five models used in our system, which include 1) LSTM, 2) CNN, 3) BiLSTM-CNN, 4) RCNN and 5) HieLSTM.

A. LSTM

A recurrent neural network model, namely long-short term memory (LSTM) (10), is used in our ensemble system. For the \( f \)-th word in the article, an LSTM takes as input \( x_f \), and produces \( h_f \) based on the following formulas:

\[
\begin{align*}
    i_f &= \sigma(W_i \cdot x_f + W_{ih} \cdot h_{f-1} + W_{ic} \cdot c_{f-1} + b_i) \\
    c_f &= \left(1 - i_f\right) \odot c_{f-1} + i_f \odot \tanh(W_c \cdot x_f + W_{hc} \cdot h_{f-1} + b_c) \\
    o_f &= \sigma(W_o \cdot x_f + W_{oh} \cdot h_{f-1} + W_{oc} \cdot c_f + b_o) \\
    h_f &= o_f \odot \tanh(c_f)
\end{align*}
\]

where \( \sigma \) is the element-wise sigmoid function, and \( \odot \) is the element-wise product. \( \{W_i\} \) is the weight matrix set. \( \{b\} \) is the bias vector set.

Then the sequence of vectors \( h_{1:m} \) output from all LSTM cells are combined into a single vector that represents the article by the max pooling layer. At last, a softmax function is used on this feature vector to compute the predictive probabilities of the article types.

B. CNN

In the convolutional neural network (CNN) model, a convolution operation is applied to produce local features. Given an input sequence \( X = [x_1, x_2, \ldots, x_n] \), a fixed size \( k \) window approach is used to capture each element’s context information. Then a matrix operation, as shown in formula (5), is applied to each successive window in the sequence:

\[
C = \text{ReLU}(W_{\text{conv}} \cdot X_{i:i+k-1} + b)
\]

where \( W_{\text{conv}} \) is the transformation matrix that is the same across all windows in the article, ReLU is the rectified linear unit function (11), and \( C \) is the convolutional layer result.

In our CNN model, two consecutive convolutional layers (window size \( k=3 \)) are stacked to extract convolutional features. Then a max pooling operation and a fully connected hidden layer are used after convolutional features. Finally, a softmax function is used to classify articles.

C. BiLSTM-CNN

In general, CNN is capable of extracting local information and LSTM can capture long-distance dependency. So we combined the two neural network architectures into a model, namely BiLSTM-CNN. The model consist two parts, a bidirectional LSTM (BiLSTM) layer and a convolution layer. Firstly, an article is represented as a sequence of embeddings by the embedding layer. Next, the embeddings are given as input to a BiLSTM layer. In the BiLSTM layer, a forward LSTM computes a representation of the sequence from left to right, and another backward LSTM computes a representation of the same sequence in reverse. These two distinct networks use different parameters, and then the representation of a word is obtained by concatenating its left and right context representations. Then a tanh layer on top of the BiLSTM is used to learn higher features. Next the features are fed into a convolution layer and a max pooling operation is used to extract global features from the convolution layer. Finally, a softmax function is used to classify articles.

D. RCNN

Similar with the BiLSTM-CNN model, Lai et al. proposed a recurrent convolutional neural network (RCNN) for
document classification (12). In the model, the recurrent structure to capture the contextual information to the greatest extent possible when learning word representations of articles, which may introduce considerably less noise compared to a traditional window-based neural network. Moreover, the model can reserve a large range of the word ordering when learning representations of articles. A max pooling layer is also employed that automatically judges which features in articles play key roles.

E. Hie-LSTM

Recently, Yang et al. proposed a hierarchical attention network (HAN) for document classification (13). The model is designed to capture two basic insights about document structure. First, since documents have a hierarchical structure (words form sentences, sentences form a document), a document representation is constructed by first building representations of sentences and then those are aggregated into a document representation. Second, it is observed that different words and sentences in a document are differentially informative.

Our implementation of the model (Hie-LSTM) is similar to HAN, which has a hierarchical structure of documents using two LSTMs instead of two bidirectional GRUs in HAN. And the max-pooling layer is used in our model instead of the original attention layer in HAN.

IV. MODEL ENSEMBLE

In our method, the results of the above mentioned models are combined via the weighted majority voting (14). In the method, only the class labels are available from the classifier outputs. We define the decision of the $r^{th}$ classifier as $d_{r,j} \in \{0,1\}$. The ensemble result then chooses class $J$ that receives the largest total vote:

$$\sum_{r=1}^{R} w_r d_{r,j} = \max_{j=1}^{C} \sum_{r=1}^{R} w_r d_{r,j}$$

(6)

where $w_r$ is the weight of classifier $r$, $\sum_{r=1}^{R} w_r = 1$ and $d_{r,j}$ is 1 or 0 depending on whether classifier $r$ chooses $j$ or not.

We found the best setting of weights via brute force grid search, quantizing the coefficient values in the interval $[0, 1]$ at increments of 0.1. The search was evaluated on our development set to avoid overfitting.

TABLE I. PERFORMANCE OF INDIVIDUAL MODELS ON OUR DEVELOPMENT SET

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>0.7157</td>
<td>0.8650</td>
<td>0.7833</td>
<td>0.8088</td>
</tr>
<tr>
<td>CNN</td>
<td>0.6784</td>
<td>0.8282</td>
<td>0.7459</td>
<td>0.7745</td>
</tr>
<tr>
<td>BiLSTM-CNN</td>
<td>0.6908</td>
<td>0.8773</td>
<td>0.7730</td>
<td>0.7941</td>
</tr>
<tr>
<td>RCNN</td>
<td>0.6878</td>
<td>0.8650</td>
<td>0.7663</td>
<td>0.7892</td>
</tr>
<tr>
<td>Hie-LSTM</td>
<td>0.7181</td>
<td>0.8282</td>
<td>0.7692</td>
<td>0.8015</td>
</tr>
</tbody>
</table>

TABLE II. THE EFFECT OF ADDITIONAL FEATURES ON PERFORMANCE ON OUR DEVELOPMENT SET

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>0.7157</td>
<td>0.8650</td>
<td>0.7833</td>
<td>0.8088</td>
</tr>
<tr>
<td>+POS</td>
<td>0.6441</td>
<td>0.8773</td>
<td>0.7429</td>
<td>0.7574</td>
</tr>
<tr>
<td>+NER</td>
<td>0.6460</td>
<td>0.8957</td>
<td>0.7506</td>
<td>0.7623</td>
</tr>
<tr>
<td>+POS+NER</td>
<td>0.6307</td>
<td>0.9325</td>
<td>0.7525</td>
<td>0.7549</td>
</tr>
<tr>
<td>CNN</td>
<td>0.6784</td>
<td>0.8282</td>
<td>0.7459</td>
<td>0.7745</td>
</tr>
<tr>
<td>+POS</td>
<td>0.6650</td>
<td>0.8160</td>
<td>0.7328</td>
<td>0.7623</td>
</tr>
<tr>
<td>+NER</td>
<td>0.6250</td>
<td>0.9203</td>
<td>0.7444</td>
<td>0.7476</td>
</tr>
<tr>
<td>+POS+NER</td>
<td>0.6717</td>
<td>0.8160</td>
<td>0.7368</td>
<td>0.7672</td>
</tr>
<tr>
<td>BiLSTM-CNN</td>
<td>0.6908</td>
<td>0.8773</td>
<td>0.7730</td>
<td>0.7941</td>
</tr>
<tr>
<td>+POS</td>
<td>0.7167</td>
<td>0.7914</td>
<td>0.7522</td>
<td>0.7917</td>
</tr>
<tr>
<td>+NER</td>
<td>0.6904</td>
<td>0.8344</td>
<td>0.7556</td>
<td>0.7843</td>
</tr>
<tr>
<td>+POS+NER</td>
<td>0.6636</td>
<td>0.8712</td>
<td>0.7533</td>
<td>0.7721</td>
</tr>
</tbody>
</table>

V. RESULTS

The organizers of the BioCreative VI PM document triage tasks provided a corpus including the training and test sets. The training set consists of 4,082 annotated PubMed articles (title and abstract) as relevant or not relevant, and the test set consists of 1,500 unannotated articles. In our experiments, we randomly selected the 10% of the training set as the development set to tune the hyper-parameters. And the document triage performance was measured with an F-score which attributes equal importance to precision and recall (F1 score). In addition, an Accuracy (Acc) measure is also used to evaluate the performance.

A. Training settings

In our method, the parameters of the model in the word embedding are initialized with pre-trained word embeddings and other parameters are initialized at random from a uniform distribution. Then all parameters of models (except for the CNN using Adadelta (15)) are optimized using RMSprop (16) to minimize categorical cross-entropy. Our models were implemented using open-source deep learning library keras (https://keras.io) and trained on a NVIDIA Tesla K40 GPU.

B. Performance of Individual Models

Table I reports the results of each individual model on our development set. We found that the CNN model performed the worst, with BiLSTM-CNN slightly better than RCNN and Hie-LSTM. The most competitive model is the LSTM model.

C. The Effect of Additional Features on Performance

We also investigated the effect of two additional features (POS and NER embeddings mentioned in the section II.B) on the performance of the models (LSTM, CNN and BiLSTM-CNN) and Table II shows the results of different combinations of these features.

The results show that F-scores generally decrease when these additional features are added into the models. When only POS feature is added, the models perform the worst. One
plausible reason is that POS information is not necessary for the biomedical document triage task. Moreover, the noise may be introduced into the model by the errors of the POS and NER tools.

D. Performance of Different Model Combinations

Finally, the results of the previous models are combined into an ensemble. The results of our submitted runs on our development set and official results of the runs on the test set are shown in Table III. Our best submission achieves an F-score of 0.6909 on the test set. On our development set, although F-scores generally decrease when the additional features are added into the models, they can help boost the performance of the ensemble system. When the results of LSTM, BiLSTM-CNN, He-LSTM, CNN with additional features and BiLSTM-CNN with additional features are combined, the highest F-score of 0.8101 is achieved on our development set. Moreover, the weights of models show that LSTM, BiLSTM-CNN and He-LSTM models have more contribution for the overall ensemble performance.

ACKNOWLEDGMENT

This work is supported by the grants from the Natural Science Foundation of China (No. 61272373, 61572102 and 61572098), Trans-Century Training Program Foundation for the Talents by the Ministry of Education of China (NCET-13-0084), and the Major State Research Development Program of China (No. 2016YFC0901902).

REFERENCES


<table>
<thead>
<tr>
<th>Model</th>
<th>Weighted</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>ACC</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>1.0</td>
<td>0.7157</td>
<td>0.8650</td>
<td>0.7833</td>
<td>0.8088</td>
<td>0.5890</td>
<td>0.8068</td>
<td>0.6809</td>
<td>0.6687</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.2</td>
<td>0.7157</td>
<td>0.8650</td>
<td>0.7833</td>
<td>0.8088</td>
<td>0.5890</td>
<td>0.8068</td>
<td>0.6809</td>
<td>0.6687</td>
</tr>
<tr>
<td>CNN</td>
<td>0.1</td>
<td>0.7157</td>
<td>0.8650</td>
<td>0.7833</td>
<td>0.8088</td>
<td>0.5890</td>
<td>0.8068</td>
<td>0.6809</td>
<td>0.6687</td>
</tr>
<tr>
<td>BiLSTM-CNN</td>
<td>0.3</td>
<td>0.7398</td>
<td>0.8896</td>
<td>0.8078</td>
<td>0.8309</td>
<td>0.6112</td>
<td>0.7945</td>
<td>0.6909</td>
<td>0.7284</td>
</tr>
<tr>
<td>RCNN</td>
<td>0.1</td>
<td>0.7157</td>
<td>0.8650</td>
<td>0.7833</td>
<td>0.8088</td>
<td>0.5890</td>
<td>0.8068</td>
<td>0.6809</td>
<td>0.6687</td>
</tr>
<tr>
<td>Hie-LSTM</td>
<td>0.3</td>
<td>0.7157</td>
<td>0.8650</td>
<td>0.7833</td>
<td>0.8088</td>
<td>0.5890</td>
<td>0.8068</td>
<td>0.6809</td>
<td>0.6687</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.3</td>
<td>0.7157</td>
<td>0.8650</td>
<td>0.7833</td>
<td>0.8088</td>
<td>0.5890</td>
<td>0.8068</td>
<td>0.6809</td>
<td>0.6687</td>
</tr>
<tr>
<td>CNN</td>
<td>0.1</td>
<td>0.7157</td>
<td>0.8650</td>
<td>0.7833</td>
<td>0.8088</td>
<td>0.5890</td>
<td>0.8068</td>
<td>0.6809</td>
<td>0.6687</td>
</tr>
<tr>
<td>BiLSTM-CNN</td>
<td>0.3</td>
<td>0.7436</td>
<td>0.8896</td>
<td>0.8101</td>
<td>0.8333</td>
<td>0.5882</td>
<td>0.8219</td>
<td>0.6857</td>
<td>0.7103</td>
</tr>
<tr>
<td>BiLSTM-CNN*</td>
<td>0.1</td>
<td>0.7157</td>
<td>0.8650</td>
<td>0.7833</td>
<td>0.8088</td>
<td>0.5890</td>
<td>0.8068</td>
<td>0.6809</td>
<td>0.6687</td>
</tr>
<tr>
<td>Hie-LSTM</td>
<td>0.2</td>
<td>0.7157</td>
<td>0.8650</td>
<td>0.7833</td>
<td>0.8088</td>
<td>0.5890</td>
<td>0.8068</td>
<td>0.6809</td>
<td>0.6687</td>
</tr>
</tbody>
</table>

*means the model using the additional POS and NER embedding