COMP5046 A2 Report

1 Data preprocessing

First, the raw dataset is read by pandas and each sample is split by comma.

Second, documents are split by "-*docstart*-". Words are split by space.

Third, I convert word into corresponding index and add the sign <START> in the beginning of each sample and <END> at the end. Each sample is a sentence.

2 Input Embedding

2.1 Input embedding training

Baseline word embedding is using random generated word2vec. Training the baseline word embedding with baseline model (BiLSTM with CRF), the f1-score is 0.9234. Then I tried three different ways of word embedding, which are pre-trained word2vec, POS tagging, Tf-idf.

First, Word2vec is produced by using pre-trained glove.840B.300d.txt, so is Word2vec embedding is 300 dimensions.

Second, Pos-tag is produced by NLTK ackage, which generates a 50 dimension vector randomly. Also, pos tagging are converted to index value. And we also add <PAD> token to the POS vocabulary and label vocabulary.

Third, Tf-idf is calculated by three steps: 1. It gets the documents by splitting the data with "-*docstart*-". 2. Tf-idf value is calculated with *Sklearn*. 3. The Tf-idf value is cut into array by the sentence level.

Also, some other processing is done in this step. Out-of-vocabulary words are initialized with random gaussian samples. Using padding to fix the length of input sentences.

The dimension of word embedding matrix is 351 because our word2vec dimension is 300, pos-tag is 50 dimension, and TF-IDF is 1 dimension.

2.2 Input embedding selection

To get the best input vector, all cases of concatenating three features are searched. The results are shown as the following table. After training with the baseline Bi-LSTM mode, I find the best input embedding is using all three features achieved the f1-score 0.9648 on validation set, which is greatly high than baseline f1-score 0.9243.

So, in my NER model, each input vector is generated by concatenating three different features embeddings, word2vec, pos, and TF-IDF.

Baseline+Ablation Study on Embedding				
word2vec	**pos**	**tfidf**	**RESULT**	**Selection**
FALSE	FALSE	FALSE	**0.9234**	
FALSE	TRUE	FALSE	**0.9302**	
FALSE	FALSE	TRUE	**0.9261**	
FALSE	TRUE	TRUE	**0.9337**	
TRUE	FALSE	FALSE	**0.9611**	
TRUE	TRUE	FALSE	**0.9631**	
TRUE	FALSE	TRUE	**0.9615**	
TRUE	**TRUE**	**TRUE**	**0.9648**	**Selected**

3 NER model

The Baseline NER model is single layer Bi-Lstm with CRF. Firstly, using the Bi-LSTM model with the input of best feature vector, we can get the probability of each name entity. **Bi**-LSTM can explore the information from both past and future, however it will lose the sequential dependency information. This is why CRF is often used for labeling or parsing of sequential data for named entity recognition. The diagram below shows how it works (cited from COMP5046 Lab09 slice).

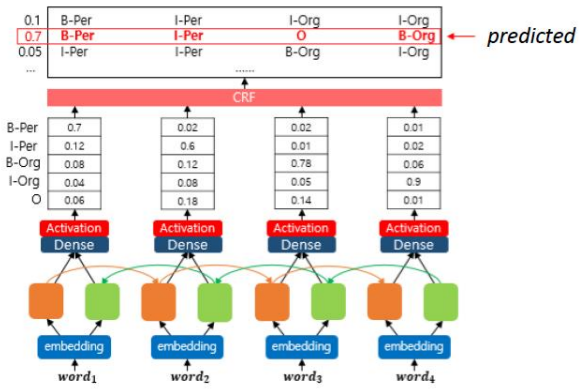

Figure 1: NER Baseline model

Comparing RNN with LSTM, LSTM has the following advantages. First, RNN has no cell state; LSTM memorizes information through cell state. Second, The RNN activation function is only tanh; LSTM introduces the sigmoid function through the input gate, forget gate, and output gate and combines the tanh function to add a summation operation to reduce the possibility of gradient disappearance and gradient

explosion. Third, LSTM can deal with both short-term dependence and long-term dependence, but RNN can only deal with short-term dependence.

3.1 Single or Multiple Layers in the Seq2Seq model

Our best model the 3 layer Bi-LSTM model. From the first table below, we can see that when layers are greater than 4, model results on validation set obviously decrease, which means too many LSTM layers may cause overfitting. So we need to find out the optimal layer number.

Number of layers should be turned together with attention strategy because they are influencing each other. When introducing self-attention, best Bi-LSTM layers is 1. After introducing two attention strategy, the best layer number is 3.

Number of layers of Bi-LSTM (+ self-attention)	Learning rate	epoch	F1-score
1	1.00E-03	20	0.9713
2	1.00E-03	20	0.9708
3	1.00E-03	20	0.9668
4	1.00E-03	20	0.8346
5	1.00E-03	20	0.8346
10	1.00E-03	20	0.8247

Number of layers of Bi-LSTM (+ self-attention + general attention)	Learning rate	epoch	F1-score
1	1.00E-03	20	0.9609
2	1.00E-03	20	0.9702
3	1.00E-03	20	0.9716
4	1.00E-03	20	0.9701
5	1.00E-03	20	0.9682

3.2 Attention

I Used the structure of seq2seq to build the NER model. As is shown in the following diagram, the first B-LSTM model who takes in the input vector can be viewed as encoder and the second Bi-LSTM who takes in the output from the encoder or attention can be viewed as decoder.

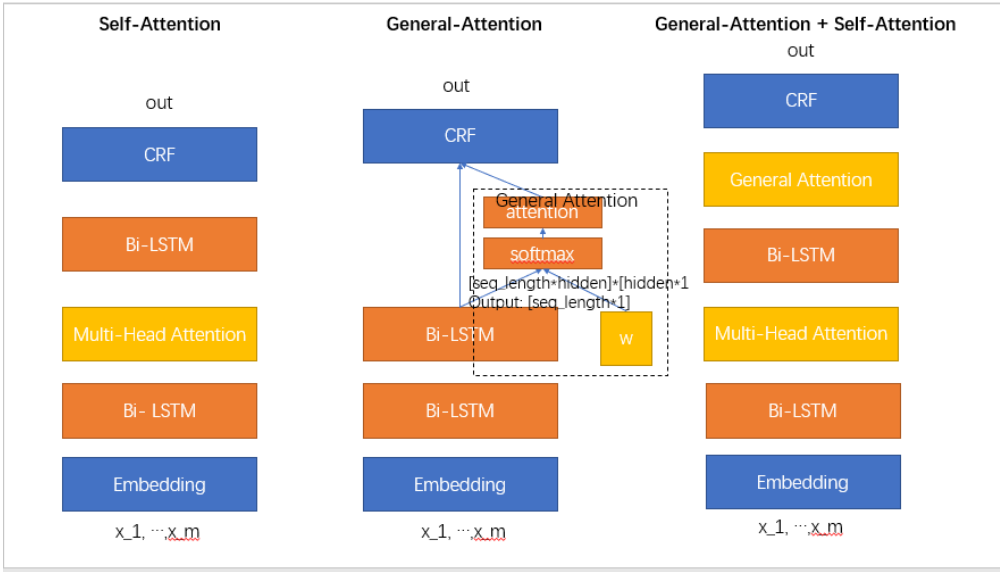

Figure 2: different attention strategy

3.2.1 Self-attention

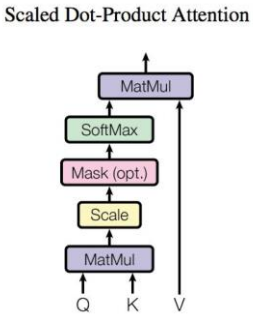

Figure 3: Scaled Dot-Product Attention

For self-attention, Q(Query), K(Key) and V(Value) are three matrixes from the input. Firstly, we make dot product of Q and K. To avoid the value from dot product is too large, we scale this value by dividing it to the $\sqrt{d_k}$,where d_k is the dimension of the source hidden state. Then we take softmax to the attention score to achieve its probability of each entity, which also called attention weight. Then multiply with V, we can achieve the weighted sum.

The equation of self-attention:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

3.2.2 Multi-head attention

Multi head attention strategy is commonly used in transformer, which can concatenate the attention score from different Q, K, V. First, a linear transformation is applied to the input matrixes Q, K, V. Then, different heads are concatenated. Afterwards, it is linearly transformed again.

3.3.3 General attention

The second strategy is using general attention score. Instead of placing the attention between encoder and decoder, in this experiment I placed it after the decoder.

$score(B, H) = B^T W_o H$, W_o is trainable weight matrix

3.3.4 Self-Attention + General Attention

As is shown in Figure 2, this assignment also tried on using self-attention and general attention together. According to our experiment result, this attention strategy is the best, which achieved the f1-score 0.9716 (best model).

As the f1-score for baseline model is 0.9648, using attention makes our model better than baseline model. Attention calculates the similarity between any two entities in a series. The series is on document level. It gives weights on important information based on document level, which makes it better than RNN. When RNN try to capture long-term dependency, it will be limited by the series length.

4 Evaluation

4.1 Evaluation setup

4.11 Setup environment

Hardware:
CPU: CORE i7 9th Gen

GPU: RTX2070

Software:
python 3.7

numpy = 1.18.1

torch=1.3.1

4.12 Hyper-parameter

Word embedding dimension: 351

Batch-size:32

Learning rate:{1e-5,5e-5,1e-4,5e-4,1e-3}

Num of heads of self-attention:{1,2,3,4,5,6,10,20}

Bi-LSTM layer num:{1,2,3,4,5}

Num of training epoch:20

Optimizer: Adam

4.2 Evaluation result

The model performance is evaluated by f1-score.

4.22 Ablation Study - different embedding model

Baseline+Ablation Study on Embedding				
word2vec	**Pos-tag**	**tfidf**	**RESULT**	**Selection**
FALSE	FALSE	FALSE	**0.9234**	
FALSE	TRUE	FALSE	**0.9302**	
FALSE	FALSE	TRUE	**0.9261**	
FALSE	TRUE	TRUE	**0.9337**	
TRUE	FALSE	FALSE	**0.9611**	
TRUE	TRUE	FALSE	**0.9631**	
TRUE	FALSE	TRUE	**0.9615**	
TRUE	**TRUE**	**TRUE**	**0.9648**	**Selected**

First, I used a random vector as baseline input embedding, then try to find how the three embedding method performs. Introducing any of the three methods will be better than baseline.

Using gird search, it shows that word2vec is the most significant feature, pos-tag is the second significant feature and Tf-IDF is the third. For this assignment, Tf-IDF has only one hidden cell, which takes less weight. Also, for NER task, Tf-IDF is less closely correspond to the task than Pos-tag.

Generally, all three methods are selected because this way of concatenation achieved the highest score.

4.21 Performance Comparison

Using three input vectors, the f1-score on baseline model is 0.9648. From the following table we can see that introducing attention can improve model performance.

The best model, which is a 3 layer Bi-LSTM using 2-heads self-attention and general attention strategy, achieved the f1-score 0.9716. It is worth mentioning that Gradient clipping is used in the training process to prevent exploding gradients.

Ablation Study on Attention, Layer num and Hyper Parameters						
Attention Type	**Head-num**	**layer**	**Learning-rate**	**epoch**	**F1-score**	
Self-attention	1	2	5.00E-05	20	0.9382	
scaled-dot-product	1	2	5.00E-04	20	0.9648	
	1	2	1.00E-04	20	0.9483	
	1	2	5.00E-03	20	0.8024	
	1	2	1.00E-03	20	0.9708	
	1	1	1.00E-03	20	0.9713	
	1	3	1.00E-03	20	0.9668	
	1	4	1.00E-03	20	0.8346	
	1	5	1.00E-03	20	0.8346	
	1	10	1.00E-03	20	0.8247	
	2	1	1.00E-03	20	0.9715	
	3	1	1.00E-03	20	0.9701	
	4	1	1.00E-03	20	0.9705	
	5	1	1.00E-03	20	0.9715	
	6	1	1.00E-03	20	0.9702	
	10	1	1.00E-03	20	0.9699	
	20	1	1.00E-03	20	0.9694	
General attention	-	1	1.00E-03	20	0.9408	
	-	2	1.00E-03	20	0.9699	
	-	3	1.00E-03	20	0.9674	
	-	4	1.00E-03	20	0.9668	
	-	1	5.00E-05	20	0.832	
	-	2	5.00E-05	20	0.879	
	-	3	5.00E-05	20	0.8006	
	-	4	5.00E-05	20	0.8935	
	-	5	5.00E-05	20	0.8867	
	-	6	5.00E-05	20	0.7346	

Self-attention + general attention	2	1	1.00E-03	20	0.9609	
	2	2	1.00E-03	20	0.9702	
	2	3	1.00E-03	20	0.9716	Best model
	2	4	1.00E-03	20	0.9701	
	2	5	1.00E-03	20	0.9682	

4.2.3 Ablation Study - different attention strategy

Attention Type	Head_num	Layer	F1-score	
Self-Attention	2	1	**0.9715**	
General Attention	-	2	**0.9699**	
Self+General	2	3	**0.9716**	Best model

Combing self-attention with general attention is slightly better than just using self-attention and its best layer is 3. If keeps increase the number of layers, model will be overfit.

4.2.4 Ablation Study - different layer strategy

For self-attention attention strategy, the experiment result table shows how the number of layers and number of heads influence the model results. Considering we have only 600 (300*2) hidden cells, I find that the optimal number of head is 2, and for each head, the number of hidden cells is 300. If using less head, the effectiveness of attention will be weakened, but if using too much heads, each head will receive very limited information.

Also, for the different of layers, self-attention shows different results. The optimal number of layer is 1.

Finally, after combining scaled dot-product self-attention and general attention, I also applied the same way of hyperparameter turning. Using the best head number, which is 2, the best layer number is 3. After 3, the model will face overfitting.

5 Reference

[1] Wolf, Thomas, et al. "Huggingface's transformers: State-of-the-art natural language processing." *ArXiv, abs/1910.03771* (2019).