CS 388 Natural Language Processing Spring 2018

Homework 2

Prof. Raymond J. Mooney *Xinrui Hua*

1 Introduction

The goal of this homework is Part Of Speech Tagging, which needs to annotate each word in a sentence with a specific part-of-speech marker.

One way to do this is to use the BiLSTM model. The BiLSTM model is a bidirectional LSTM model, which contains a forward LSTM cell and a backward LSTM cell. The input of both the cells are the same and the output of the BiLSTM model is the concatenation of both cells' output. The LSTM cell is a memory cell that contains a forget gate, which can remember things from the previous cell, an input gate, which deals with the new input and an output gate, which can output all the states of the cell. The whole network architecture is that, we embed the words tensor and put it as the input into both forward and backward cell. Then both the cells output the hidden states and we concatenate them into a tensor. Finally we treat the tensor as an input of a Softmax layer and get the probabilities of different POS tags.

2 Orthographic Features

In this section I will discuss the four features I used in homework.

(1) The first feature is whether the word is capitalized. If the word is capitalized, it's likely that the word may be a Proper Noun, such as Austin, Bob and so on.

The method I use to determine whether the word is capitalized is to check $word == word.capitalize()$.

(2) The second feature is whether the word contains a common English suffix. I use the following list of suffix: 'age', 'al', 'ance', 'ence', 'dom', 'ee', 'er', 'or', 'hood', 'ism', 'ist', 'ity', 'ty', 'ment', 'ness', 'ry', 'ship', 'sion', 'tion', 'xion', 'able', 'ible', 'en', 'ese', 'ful', 'ic', 'ish', 'ive', 'ian', 'less', 'ly', 'ous', 'ate', 'en', 'ify', 'ise', 'ize'.

So I just need a loop to check whether the word ends with the suffix or not. Actually, these suffices can also be separated into several categories. For example, 'ee' is probably the suffix of a noun, and 'en', 'ify', 'ise', 'ize' is probably the suffix of a verb, 'less', 'ous' may be the suffix of an adjective. However, I'm not a expertise in English so I just choose a boolean feature that contains a suffix or not.

(3) The third is whether the word contains a hyphen. There are many types of words containing a hyphen. For example, sugar-free is a compound adjective, spot-check is a compound verb and break-in is a noun. Hence I only check the word to see whether it has a hyphen.

(4) The last one is whether the word begins with a number. There can be many numbers in the sentences, such as the age and the year. And the number can be presented in two ways. One way is in the form of $0 - 9$. The other is in English. Thus we should check both of them.

3 Implement

First of all, I add one output value for the function *get_processed_data*, which is the tensor for Orthographic Features.

The next thing is that I also add one input tensor for the model, which can be seen as $input\_features$. Thus the model will initialize with a tensor with shape $[128, 100, 4]$. And for each training, validating and testing, $feed\_dict$ will have three tensors.

3.1 Input

There are two ways to add the Orthographic Features into the LSTM model. The first is to add them into LSTM input tensors. Actually, after the input words tensor gets embedding, the shape of the input tensor is $[128, 100, 300]$. Hence we just concatenate the input tensor with the Orthographic Features tensor. And at last we get an input tensor with shape $[128, 100, 304]$. Thus in this case, we change the hidden states of the model from 300 to 304. And finally the output of both the forward and backward cells is a tensor with shape $[128, 100, 608]$.

3.2 Output

The other way to add Orthographic Features is to add them into the input of the softmax layer. Actually the output from the forward and backward cells is a tensor with shape $[128, 100, 600]$. And we concatenate it with Orthographic Features tensor to get a input tensor of shape $[128, 100, 608]$ for the softmax layer.

3.3 OOV Accuracy

I imitate the way to calculate accuracy and use the mask to get the OOV words. We know that the value of OOV words is equal to the length of vocabulary. Thus we use the mask of value $input\_dim - 2$ to get all the OOV words and the length of them. Then the accuracy is just the ratio of the correct POS tags over the OOV words and the length of all OOV words, and that is the function *compute_accuracy*.

4 Result

I set the number of epochs to be 6 and the batch size to be 128. The all training contains more than 700 batches. The following table 1 shows the results. The brief result is that the input model has the highest accuracy, lowest loss and longest training time, while the baseline model has the lowest accuracy, highest loss and shortest training time. And we can see from the figure that after about 500 batches, the promotion of the training is nearly zero.

Then I will discuss the two questions.

1. How does adding orthographic features affect the accuracy (both overall and OOV) and runtime of the BiLSTM and why?

	baseline	input	output
Validation Accuracy	0.956	0.960	0.956
Validation Loss	0.154	0.129	0.147
Validation OOV Accuracy	0.575	0.693	0.612
Test Accuracy	0.956	0.960	0.958
Test Loss	0.144	0.126	0.140
Test OOV Accuracy	0.548	0.669	0.592
Training Time	1225.31	1254.15	1240.03

Table 1: Results

We can see from both input and output model that, adding orthographic features really increase both the overall and OOV accuracy of BiLSTM model. But in the same time the training time also becomes longer.

The reason for longer training time is that there are more parameters in the model. For input model, the number of hidden states increases to 304 and the length of input for softmax layer increases to 608. For the output model, the length of input for softmax layer increases to 604.

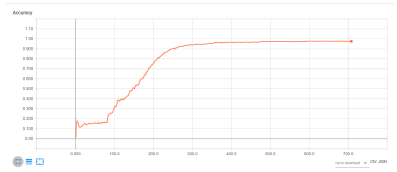
The reason for the promotion of accuracy is that these orthographic features is helpful for the POS tagging task, especially for OOV words. For OOV words, after adding orthographic features, we become more confident to give POS tags because now we have more information of the OOV words. For example, the OOV word with a suffix -ly is likely to be an adverb, which is very helpful for us to determine the POS tags. And another example is the numbers. It's obvious that many numbers in test set are OOV words. So if we have the knowledge that it's a number, we can reduce the number of possible POS tags. And thus also for overall accuracy, with the increase of OOV accuracy and more knowledge on words, the accuracy is higher.

2. How does changing the approach to adding these features (input vs. classification layer) affect the accuracy (both overall and OOV) and runtime of the BiLSTM and why?

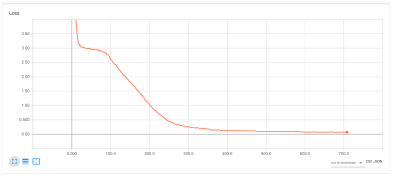
   The runtime of output model is shorter because there are fewer parameters in output model.

   The overall and OOV accuracy of input model is higher. I think one reason is that there are more parameters in input model. And another reason is that, adding features in input is benefit for POS tagging of the whole sentence. For example, if we find that the current word is a number, we may guess that this is a number for age or day or year, which helps us to figure out POS tags for neighbour words. But if we add them in output, the features can only help us determine the POS tags for this word.

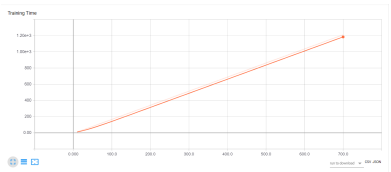
The figures are attached here. And all the figures can be seen in trace directory.



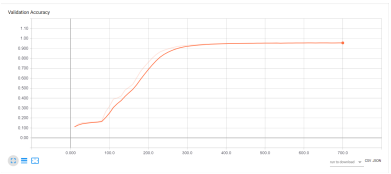
(1) baseline accuracy



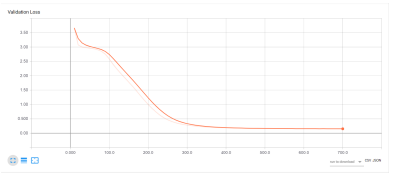
(2) baseline loss



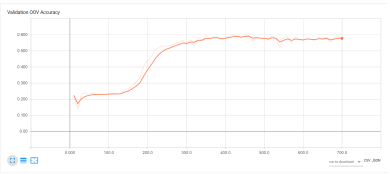
(3) baseline time



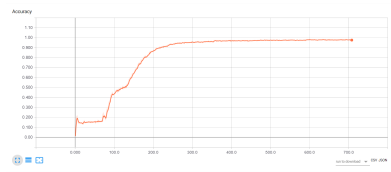
(4) baseline val accuracy



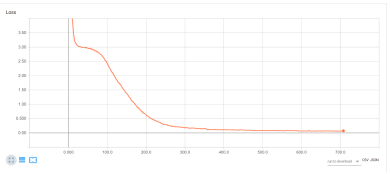
(5) baseline val loss



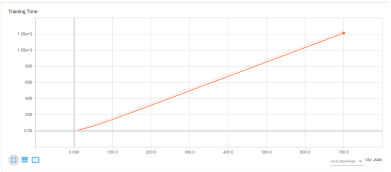
(6) baseline val oov accuracy



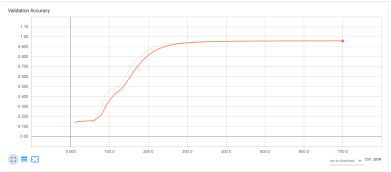
(7) input accuracy



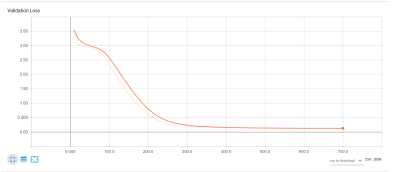
(8) input loss



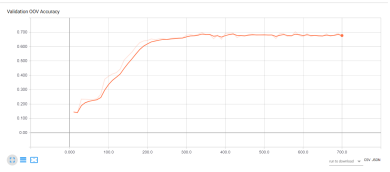
(9) input time



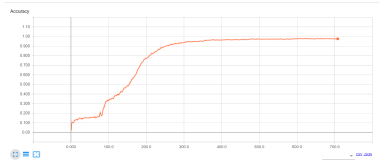
(10) input val accuracy



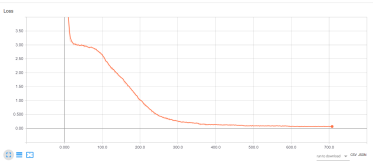
(11) input val loss



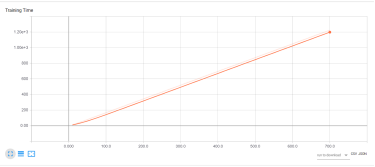
(12) input val oov accuracy



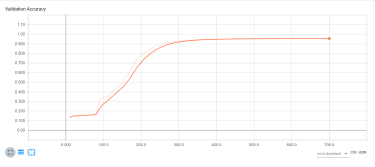
(13) output accuracy



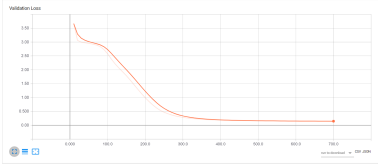
(14) output loss



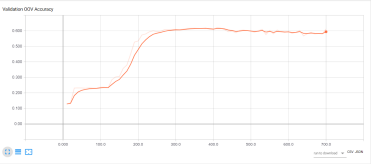
(15) output time



(16) output val accuracy



(17) output val loss



(18) output val oov accuracy