CS 388 Natural Language Processing
Homework 2: Part-of-Speech Tagging with LSTMs

Lihang Liu
EID: ll32632

1 Introduction

Bidirectional Long Short Term Memory networks (BiLSTMs) is a powerful tool for sequential labelling. In this project, we are going to apply BiLSTMs to the task of Part-Of-Speech (POS) tagging using data from the Penn Treebank. More importantly, we utilized orthographic features of English words and discussed different ways of incorporating into the BiLSTM model.

2 Background

In this section, we will introduce the basic concept of BiLSTM, POS tagging and orthography.

2.1 BiLSTM

Bidirectional Long Short Term Memory networks (BiLSTMs) is a specific type of Long Short Term Memory Network. BiLSTMs are introduced to increase the amount of input information available to the network by combining features from both directions of the current state. BiLSTM is especially useful when the context of the input is available and help reduce the ambiguity of the current state. For example, in the application of handwriting recognition, the information before and after the current state help in determining the right word.

2.2 POS tagging

POS tagging is a basic task of Natural Language Processing, which help the semantic analysis, word sense disambiguity, etc. Give a input sentence, the goal of POS tagging is to assign a label of part-of-speech to a word given its definition and context.

2.3 Orthography

English orthography is the writing conventions used to represent spoken English by connecting its spelling to its meaning. For example, whether it's capitalized, contains a common English suffix, contains a hyphen, or starts with a number.

The reason why orthography will help the task of POS tagging is the relation from the orthography to the meaning. For example, a word ends with "tion" should be a noun with high probability. We will discuss more in the Experiments part.

3 Methodology

In this section, we will discuss the main task of this project: how to extract orthographic features, how to incorporate into the BiLSTMs model and how to calculate the OOV (out of vocabulary) accuracy.

3.1 Orthographic Features

3.1.1 Definition

As stated in the Background section, the orthographic features can be used to improve the performance of POS tagging. In this project, we utilize 5 kinds of orthographic features:

- Whether the word is capitalized.
- Whether the word contains a hyphen.
- Whether the word starts with a number.
- Whether the word starts with a specific type of English prefix.
- Whether the word ends with a specific type of English suffix.

Here we collect 23 common English prefix as follows:

> 'anti', 'de', 'dis', 'en', 'em', 'fore', 'in', 'im', 'il', 'ir', 'inter', 'mid', 'mis', 'non', 'over', 'pre', 're', 'semi', 'sub', 'super', 'trans', 'un', 'under'

As well as 30 common English suffix as follows:

> 'able', 'ible', 'al', 'ial', 'ed', 'en', 'er', 'est', 'ful', 'ic', 'ing', 'ion', 'tion', 'ation', 'ition', 'ity', 'ty', 'ive', 'ative', 'itive', 'less', 'ly', 'ment', 'ness', 'ous', 'eous', 'ious', 'es', 's', 'y'

3.1.2 Encoding

We have discussed 4 kinds of orthographic features in the previous section. The first 3 kinds of orthographic features are easy to encode: use one bit for each orthographic feature, 1 stands for present and 0 stands for absent.

While there are several ways to deal with the prefix\suffix orthographic features: 1. encode the prefix\suffix feature as an integer which presents its index; 2. encode the prefix\suffix feature as one bit which presents whether the word contains any of the common prefix\suffix; 3. encode the prefix\suffix feature as an one-hot vector, of which each bit represents whether the corresponding type of prefix\suffix is present or not. The defect of method 1 is that it's not easy for the network to learn that the integer value stands for an discrete index rather than a continuous value of some property. The defect of method 2 is that the encoding is so lossy that many helpful information are discarded. As a result, we utilize the method 3 to encode the prefix\suffix features. To summarize, we now have an orthographic feature of length $3 + 23 + 30 = 56$.

It's worth to mention that as the number of orthographic features (or other features) increases, the computational cost also increases. One solution is to embed the input features to shorter features, following the idea of word embedding. This is not used in our implement as the current orthographic features are not the bottleneck of the network.

3.1.3 Incorporate into BiLSTMs

There are two ways to incorporate the orthographic features into the BiLSTMs model. One way is to concatenate the orthographic features with the word embedding output, and then feed forward to the LSTM cells. Another way is to concatenate the orthographic features with the LSTM output, and then feed forward to the full-connected layers followed by POS prediction.

3.2 OOV Accuracy

One common problem of Natural Language Processing is the unknown words encountered in the test set. In the case of POS tagging, when the network gets an unknown word, the word embedding can't provide useful input features for the network to learn from. However, the network can use the context to infer the correct POS tag. Also, after we add the orthographic features, they will also help in determining the POS tag.

To compute the OOV accuracy, we firstly get a binary mask where 1 stands for the present of an unknown word and 0 stands for the present of a known word. Then apply the binary mask to the final softmax prediction. So only unknown words are considered for the OOV accuracy. Finally, we divide the aggregated accuracy by the total number of unknown words. In the implementation, we reuse the $compute\_accuracy$ function of the $Model$ class. The total number of unknown words can be calculated by simply summing over the binary mask.

4 Experiments

4.1 Dataset

We conduct our experiments on the Penn Treebank dataset, which consists of around 21,000 sentences and a set of 69 POS tags. To evaluate the performance, we apply a 80:10:10 split for train, dev and test, respectively. We preprocessed the dataset by get rid of all sentences with length greater than 100.

4.2 Experimental Setup

We use the BiLSTMs as the network architecture and the Adam Optimizer as the default optimizer. The learning rate is set to 0.0005. The batch size is set to 128.

4.3 Evaluation metrics

To evaluate the performance, we use two metrics: the accuracy for all words and the accuracy for only the OOV words.

4.4 Comparisons

To evaluate the benefit of orthographic features, we compare the following 3 methods:

- **Baseline**. The original BiLSTMs model.
- **Input**. The BiLSTMs model with the orthographic features concatenated with the embedding layer.
- **Ouput**. The BiLSTMs model with the orthographic features concatenated with the LSTM output layer.

4.5 Results

The experimental results are shown in Tab 1. The runtime is based on GPU environment.

	Val Overall Accuracy	Val OOV Accuracy	Test Overall Accuracy	Test OOV Accuracy	Runtime
Baseline	95.4%	57.4%	95.4%	54.7%	398s
Input	96.4%	78.8%	96.4%	79.1%	454s
Output	95.4%	57.3%	95.5%	57.0%	422s

Table 1: Comparison on the Overall accuracy and OOV accuracy of 3 methods.

4.6 Discussions

Overall, adding orthographic features improves the performance on overall accuracy and even more on OOV accuracy. Because the orthographic features provide extra features besides a pure index for each word. This is especially helpful when we encounter an unknown word. For example, when the word ends with a suffix "tion", it's high possible that it's a noun. When the words is capitalized, it may be a proper noun. These orthographic information will help in determining its POS tag. However, the runtime is longer since the added orthographic features enlarge the feature dimension of the model, which requires more computation.

If we compare the method Input and Output, the improvement from method Input on the overall accuracy is larger: 1.0% versus 0.1%, and even larger on the OOV accuracy: 24.4% versus 2.3%. This is a huge difference. It's because the method Input concatenates the orthographic features to the embedded layer before the LSTM model, so the LSTM model can extract higher layer of features and interact with word embedding features. While, the method Output concatenates the orthographic features after the LSTM model, which doesn't have such advantage. However, the runtime of the method Input seems a bit longer. This is because for method Input, the dimension of the hidden state of LSTM cells increases. While for method Output, it increases the dimension of the final full-connected layers. The computation of LSTM cells are more costly than that of the full-connected layers, so the method Input runs longer.

5 Conclusion

In this project, we reproduced a BiLSTM model for the task of POS tagging. To improve the performance, we utilized orthographic features of each word and incorporated into the BiLSTM model. The experimental results show that concatenating orthographic features with the embedded layer is beneficial for the overall accuracy, especially for the OOV accuracy.