# Part-of-Speech Tagging with LSTMs

Pandian Raju (pandian@cs.utexas.edu, UT ID: pr22695)

March 2017

## 1 Introduction

Long Short Term Memory (LSTM) is a recurrent neural network (RNN) architecture extensively used in the field of deep learning to learn and classify natural language and for other machine learning problems. As a part of this project, a basic model of bidirectional LSTM is given which attempts to assign the Part-of-Speech (POS) tags to a set of sentences, given a training set. The work in this project aims at a) adding a metric to calculate the prediction accuracy for Out-of-Vocabulary (OOV) words and b) feeding orthographic features of the words to the LSTM along with the words themselves and analyze the accuracy and OOV accuracy obtained. In doing so, the orthographic features are added to two different layers: at the input of LSTM and at the output classification layer and the accuracies obtained for these two models are compared and analyzed.

## 2 Background

- As a part of preprocessing, all the sentences are clipped to a length of 100 meaning that if a sentence has more than 100 words, only first 100 words are fed to the LSTM along with the words themselves.
- During training phase, a unique feature id is assigned to each distinct word and during validation/test phase, the feature id for the word is looked up from the training phase and a unique OOV id is assigned to words which did not appear in the training phase.
- A default embedding is used to convert the input dimension of approximately around 41000 (number of distinct words in the training set) to hidden state size of 300 for computational tractability.
- Tensorflow framework is used to implement the bidirectional LSTM.

## 3 Methods

### 3.1 OOV Accuracy

- The OOV accuracy is obtained similar to how the average accuracy is obtained with the only difference being the mask used to get the accuracy.
- OOV Accuracy of a particular batch is defined as the ratio of number of correct POS predictions of OOV words in that batch to the total number of OOV words in that batch.
- The mask is defined to be 1 for those tokens with feature id len(vocabulary), which is the default id assigned to OOV words during test/validation phase and 0 otherwise.

### 3.2 Orthographic features

- Orthographic features are additional features for each word like capitalization, starting with a number, containing a hyphen, having some common suffixes or prefixes. These features are extracted for each word and fed to the LSTM.

- To extract these features, a set of prefixes and suffixes are defined and during preprocessing phase, each word is checked against these features and the feature ids corresponding to them are returned.
- These features can then be fed into LSTM either in the input layer or the output classification layer. A set of around 100 orthographic features are used as exhaustive suffixes/prefixes and around 60 features are used as common suffixes/prefixes. Examples of some prefixes or suffixes include 'anti', 'extra', 'ir', 'il', 'ity', 'al', 'ment'etc.

### 3.3 Adding to the input layer

- At the input layer of LSTM, the orthographic features are converted to one-hot vectors and appended to the embedded input of dimension 300. Separate one-hot vectors are appended for both prefix and suffix since a word can contain both of them independently.
- The hidden state size then becomes 300 + dimension (prefix-orthographic-feature) + dimension (suffix-orthographic-feature). The remaining flow remains the same once the orthographic features are appended to the input.
- Additionally, experiments are conducted by using the feature ids as integers (instead of converting to one-hot vectors) and by embedding the orthographic features to orthographic hidden state size of size 10 for each prefix and suffix.

### 3.4 Adding to the output layer

- The same method is followed as for the input layer except that instead of appending the one-hot orthographic vectors to the LSTM input, they are appended to the LSTM output just before softmax is applied and classification is done (where the LSTM output is converted into one-hot vectors of POS tags).
- Similar to input layer, experiments are conducted using integer values, one-hot vectors and embedded orthographic features.

## 4 Results

- Tables 1 and 2 summarize the average accuracy and OOV accuracy obtained during validation and testing phases for different models respectively.

Table 1: Validation accuraices for different models

| **Validation** | **BASELINE** | | **INPUT** | | **OUTPUT** | |
|---|---|---|---|---|---|---|
| | **Average accuracy** | **OOV accuracy** | **Average accuracy** | **OOV accuracy** | **Average accuracy** | **OOV accuracy** |
| **ONE HOT** | 95.5 | 57.7 | 96.2 | 73.7 | 95.6 | 60.3 |
| **INT VALUES** | 95.5 | 57.7 | 95.7 | 61.6 | 95.7 | 60 |
| **EMBEDDED** | 95.5 | 57.7 | 96.3 | 75 | 95.8 | 65.3 |

- It can be seen that compared to the baseline model, adding orthographic features at input and output layer improve the overall accuracy by a small amount but improve the OOV accuracy by substantial amount.
- The experiment using just integer values instead of one-hot vectors improves the accuracies only by a small margin because the LSTM can distinguish between features well if they are fed as one-hot vectors whereas while passed as integer values, there will be only one dimension for the orthographic features which is not very effective in the classification.

Table 2: Testing accuracies for different models

| Testing | BASELINE | | INPUT | | OUTPUT | |
|---|---|---|---|---|---|---|
| | Average accuracy | OOV accuracy | Average accuracy | OOV accuracy | Average accuracy | OOV accuracy |
| **ONE HOT** | 95.7 | 56.1 | 96.4 | 74.7 | 95.8 | 60 |
| **INT VALUES** | 95.7 | 56.1 | 95.9 | 60.7 | 95.8 | 57.4 |
| **EMBEDDED** | 95.7 | 56.1 | 96.5 | 75.2 | 96 | 64.8 |

- There is very little difference between the accuracies obtained by using one-hot vectors or by using the embedded form of the orthographic features. Since the dimension of the orthographic features is very small (around 40-50), embedding them doesn't provide any advantages since it is mostly used for computational tractability to transform very high dimension space to a lower one.
- The increase in OOV accuracy over the baseline model is significant while adding the orthographic features to the input than to the output layer. The reason is because adding to the input layer actually adds the orthographic features to the LSTM RNN and those features are taken into account during all the backpropagation process and the LSTM can be better trained in this case. But in case of adding them to the output layer, the features are just appended to the LSTM output which then is converted to one-hot POS tag vectors. The amount of learning (by error backpropagation) involving the orthographic features is less in this case compared to adding them to the input layer.

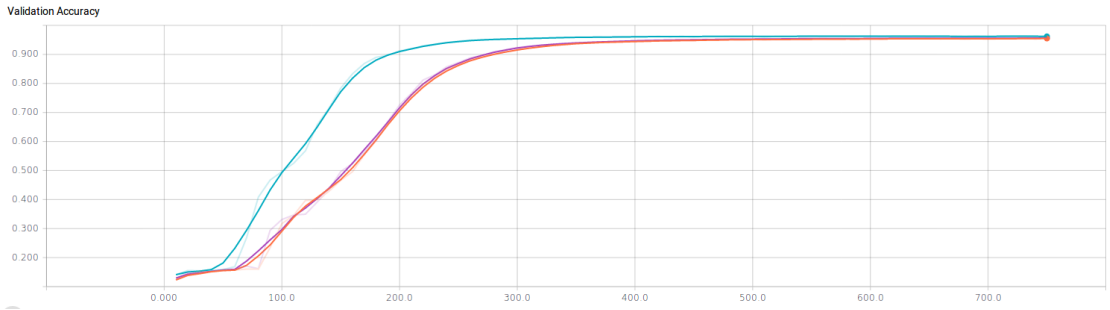


Figure 1: Validation accuracies for Baseline(Red), Input(Cyan) and Output(Violet)

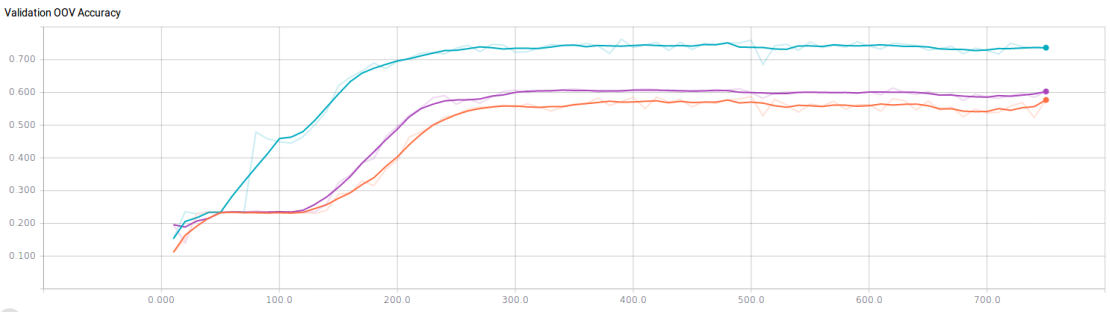


Figure 2: Validation OOV accuracies for Baseline(Red), Input(Cyan) and Output(Violet)

- Adding the orthographic features improves the OOV accuracy by a big margin but not the overall accuracy because the OOV words do not have any features associated with them without the orthographic features and hence adding these features helps in classifying them much better. On the other hand, the other words which appeared in training set already

have features associated with them (word one-hot vector) and orthographic features are just addition to them and hence the increase in accuracy is not significant.

- Figures 1 and 2 show the tensorboard plots for validation accuracy and validation OOV accuracy for all 3 models. It can be seen that for overall accuracy, input model attains a local maximum within lesser number of epochs but the final accuracies are very close for all the three models.
- The OOV accuracies are clearly better for input model than the other two since the orthographic features are actually fed to the LSTM and the features are learnt. Baseline model has the lowest OOV accuracy among the three since it doesn't include the orthographic features at all.

## 4.1 Run-time performance


Figure 3: Number of batches versus the training time for different models

- Figure 3 shows the training time for the different models for different number of batches trained.
- It's clear that the training time increases linearly with the number of batches (or number of epochs) trained on irrespective of the type of the model.
- Adding the orthographic features to the input layer takes more training time compared to the other two. It's because adding the orthographic features effectively increases the hidden state size and hence directly increases the amount of learning (error backpropagation) and so processing happening in the LSTM cell.
- Training time for output model is only slightly higher than the baseline model because in this case the hidden state size is still same and only in the final fully connected layer where the output of LSTM is converted to POS tag one-hot vectors, the orthographic features are added.

## 4.2 Underfitting and overfitting

- Figure 4 shows the effect of the number of epochs the BiLSTM is trained for on the test accuracies obtained.
- It can be observed that the accuracies are very low for 1 epoch and gets to a reasonable local peak for 2 epochs. Hence if the BiLSTM is trained only for very few epochs (1 in this case), the test accuracies will be very low and this is because of underfitting.

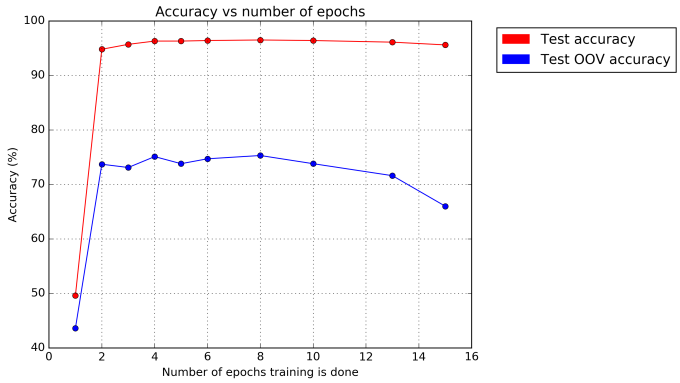


Figure 4: Test accuracies versus the number of epochs

- The other case is overfitting where too much of training causes the accuracies to drop. In this case, beyond 8 epochs, the OOV accuracy starts to decrease steadily with each additional epoch trained for. Hence, an optimal value should be chosen for the number of epochs (4 or 8 epochs seems to be an optimal value for this experiment).

### 4.3 Orthographic features

- Table 3 shows the effect of number and type of orthographic features used on the accuracies obtained.

Table 3: Effect of number of orthographic features on accuracy

| **Variant** | **Test overall accuracy** | **Test OOV accuracy** |
|---|---|---|
| Only suffix (∼50 features) | 96.5 | 75.5 |
| Only prefix (∼50 features) | 96.3 | 71.4 |
| 60 features | 96.3 | 73.1 |
| 100 features | 96.4 | 75.8 |

- It can be seen that suffixes are more helpful in predicting the POS tags of OOV words than the prefixes.
- As the number of suffixes or prefixes used increases, the test OOV accuracy also increases. Hence to achieve good accuracies, one needs to capture as many orthographic features as possible. But adding too many very infrequent features might become noise and lead to overfitting.

## 5 Conclusion

Bidirectional LSTM is used to predict the POS tags for the words. New orthographic features are defined and added to the LSTM and its effect on the accuracies obtained are discussed. Adding the features to the input layer proves to be more helpful than adding to the output layer, but at the cost of training time. Also, the run-time performance of the models are analyzed and optimal number of epochs to train the model is also discussed. With all these parameters and the models, one can choose the right set of parameters to maximize the test accuracies obtained.