Introduction

This report is about the process that led me to make a word segmenter for Chinese language. During the report I will explain every choice I have made to get the results.

Preprocess of the dataset

For the first preprocessing part I have looped through the sentences, labelling every sentence, adding every word and every two words to two dictionary (one for the char and one for the bigram) with an increasing identifier, including an Unknown value in case a word of the input file was not present in the embeddings and removing any whitespace from the sentences creating a new file.

For the second part of the preprocessing I have looped through the sentences without whitespaces and through labels to create the train, dev and test set for both chars and bigrams input, setting the padding to the length of the longest sentence. After this process we get a 2Dimensional instances sets and a 3Dimensional classes sets.

Model

As regards the model I have used a multi input layer with two input adding to each of them the embeddings, then concatenating them and adding a Bidirectional LSTM layer and a last Dense layer with softmax activation.

To improve the model performances I have added a recurrent dropout which has reduced the problem of overfitting.

I have tuned the batch size and the number of hidden size of the RNN and found out that 64 of batch size and 100 hidden size of RNN works better for the limited amount of testing I have done.

Testing

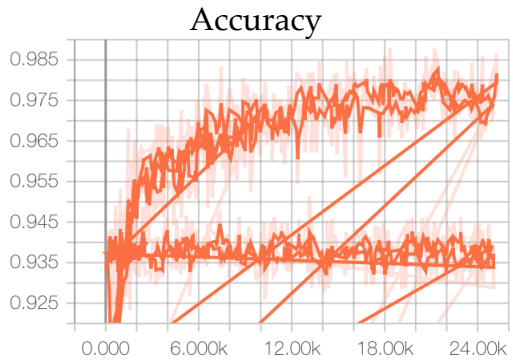
As regards the testing I have firstly loaded the vocabularies used in the training and checked it against the input file and replaced any unknown char or unknown bigram with a 'U' so that the model can handle the unknown words and then used my own trained model to predict the labels of the input file. As regards the length, in case a sentence was longer than the input of the model I would split it in subarrays of length equals to the input of the model and stores the indices. In the prediction if an array is a subarray the algorithm will not write a newline

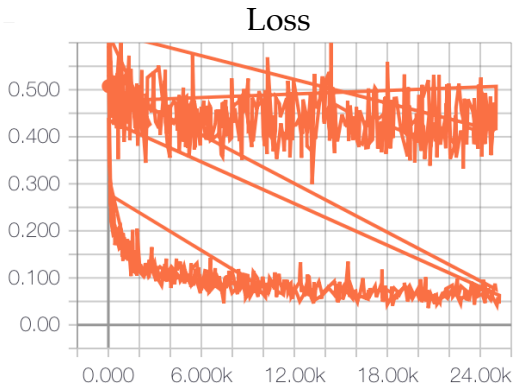
Results

Using the score function on a subset of the as_training.utf8 file I got

Precision: 0.9848808876061602

Graphs and model





On the accuracy graph it is possible to see the validation accuracy which is around 97% after the recurrent dropout.

```
Model only with char
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 188)         1122924
_________________________________________________________________
bidirectional (Bidirectional (None, None, 200)         231200
_________________________________________________________________
bidirectional_1 (Bidirection (None, None, 200)         240800
_________________________________________________________________
dense (Dense)                (None, None, 4)           804
=================================================================
Total params: 1,595,728
Trainable params: 1,595,728
Non-trainable params: 0
_________________________________________________________________
```

```
Model using the bigrams
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, 164)          0
__________________________________________________________________________________________________
input_2 (InputLayer)            (None, 164)          0
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 164, 164)     925616      input_1[0][0]
__________________________________________________________________________________________________
embedding_2 (Embedding)         (None, 164, 164)     91313396    input_2[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 164, 328)     0           embedding_1[0][0]
                                                                 embedding_2[0][0]
__________________________________________________________________________________________________
bidirectional_1 (Bidirectional) (None, 164, 200)     343200      concatenate_1[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 164, 4)       804         bidirectional_1[0][0]
==================================================================================================
Total params: 92,583,016
Trainable params: 92,583,016
Non-trainable params: 0
__________________________________
```

Hyper parameters used for training

Batch size	64
Epochs	3
RNN hidden size	100
RNN dropout rate	0.2
Learning rate	1E-03
Decay	1e-3 / 200
Input	164