Sapienza University of Rome

Natural Language Processing
Chinese Word Segmentation

Ibis Prevedello

April 22, 2019

1 Introduction

The goal of this project is to train a model based on Bidirectional LSTM to separate chinese words in a sentence [1].

The dataset used for the training was the concatenation of four different datasets: AS (Traditional Chinese), CITYU (Traditional Chinese), MSR (Simplified Chinese) and PKU (Simplified Chinese).

The training was done using a Google Compute Engine instance running a Tesla K80 GPU.

2 Dataset Preparation and Augmentation

Using all the four datasets the total number of samples is 867.444 with a testing set of 500 samples.

For the augmentation, first the dataset is converted to Simplified Chinese, then a dictionary is created using all unigrams formed of the characters that appear more than 20 times in the dataset. For the less frequent unigrams, it is replaced by a `<UNK>` flag, that represents a unknown symbol. This was necessary because the network also needs to learn what to do when it sees a totally new character.

All the sentences are then associated to the id of the respective character in the dictionary and the labels are converted to the BIES format, where B represent the beginning of the word, I inside, E end and S for single characters, which are then converted to categorical.

3 Network Training

The basic network is composed by two Bidirectional LSTM, but for the one of the training it was used a three-layer Bidirectional LSTM.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 626, 64)           152320
_________________________________________________________________
bidirectional (Bidirectional (None, 626, 512)          657408
_________________________________________________________________
bidirectional_1 (Bidirection (None, 626, 512)          1574912
_________________________________________________________________
dense (Dense)                (None, 626, 5)            2565
=================================================================
Total params: 2,387,205
Trainable params: 2,387,205
Non-trainable params: 0
_________________________________________________________________
```

The network was trained using different parameters, for this purpose a grid search function is implemented, where a dictionary with the name of the parameters and a list of values to be tried is passed.

The parameters tried were the following: epoch, batch size, merge mode for the LSTM, LSTM layers, embedding size. However the way the code is implemented is possible to define all the variables to the grid search, such as: hidden layer size, dropout, recurrent dropout, learning rate, and so on.

4 Results

Below is presented the list of all the different parameters that were used for the training.

```
Loss: 0.5546 - Accuracy: 0.7830
 Parameters: {'lstmLayers': 2, 'epochs': 3, 'batchSize': 32,
 'mergeMode': 'concat', 'embedding_size': 64}
Loss: 0.5617 - Accuracy: 0.7829
 Parameters: {'lstmLayers': 2, 'epochs': 3, 'batchSize': 32,
 'mergeMode': 'sum', 'embedding_size': 64}
Loss: 0.4826 - Accuracy: 0.8178
 Parameters: {'lstmLayers': 2, 'epochs': 6, 'batchSize': 32,
 'mergeMode': 'concat', 'embedding_size': 64}
Loss: 0.4674 - Accuracy: 0.8227
 Parameters: {'lstmLayers': 2, 'epochs': 6, 'batchSize': 32,
 'mergeMode': 'sum', 'embedding_size': 64}
```

```
Loss: 0.5164 - Accuracy: 0.8053
 Parameters: {'lstmLayers': 2, 'epochs': 3, 'batchSize': 64,
 'mergeMode': 'concat', 'embedding_size': 64}
Loss: 0.5226 - Accuracy: 0.8007
 Parameters: {'lstmLayers': 2, 'epochs': 3, 'batchSize': 64,
 'mergeMode': 'sum', 'embedding_size': 64}
Loss: 0.4470 - Accuracy: 0.8367
 Parameters: {'lstmLayers': 2, 'epochs': 6, 'batchSize': 64,
 'mergeMode': 'concat', 'embedding_size': 64}
Loss: 0.4820 - Accuracy: 0.8294
 Parameters: {'lstmLayers': 2, 'epochs': 6, 'batchSize': 64,
 'mergeMode': 'sum', 'embedding_size': 64}
Loss: 0.4822 - Accuracy: 0.8407
 Parameters: {'lstmLayers': 2, 'epochs': 6, 'batchSize': 128,
 'mergeMode': 'sum', 'embedding_size': 64}
Loss: 0.4479 - Accuracy: 0.8350
 Parameters: {'lstmLayers': 2, 'epochs': 10, 'batchSize': 64,
 'mergeMode': 'sum', 'embedding_size': 64}
Loss: 0.5233 - Accuracy: 0.7996
 Parameters: {'lstmLayers': 2, 'epochs': 6, 'batchSize': 64,
 'mergeMode': 'sum', 'embedding_size': 64}
Loss: 0.4525 - Accuracy: 0.8425
 Parameters: {'lstmLayers': 3, 'epochs': 6, 'batchSize': 128,
 'mergeMode': 'sum', 'embedding_size': 64}
Loss: 0.4285 - Accuracy: 0.8462
 Parameters: {'lstmLayers': 3, 'epochs': 10, 'batchSize': 64,
 'mergeMode': 'sum', 'embedding_size': 64}
Loss: 0.4283 - Accuracy: 0.8517
 Parameters: {'lstmLayers': 3, 'epochs': 6, 'batchSize': 64,
 'mergeMode': 'sum', 'embedding_size': 64}
Loss: 0.5117 - Accuracy: 0.8104
 Parameters: {'lstmLayers': 3, 'epochs': 6, 'batchSize': 64,
 'mergeMode': 'sum', 'embedding_size': 16}
Loss: 0.4874 - Accuracy: 0.8258
 Parameters: {'lstmLayers': 3, 'epochs': 6, 'batchSize': 64,
 'mergeMode': 'sum', 'embedding_size': 32}
```

For the training, the best accuracy obtained was 0.8517 using 6 epochs, batch size of 64 samples, 'sum' as the merge mode, 3 LSTM layers and a embedding size of 64.

Analyzing this data, it is possible to notice that the merge mode for the Bidirectional Layers did not influence much on the result, being the number of epochs and the batch size the biggest factors.

In the images 1 and 2 is possible to visualize the training and the testing accuracy for the best result obtained.

5 Conclusion

The goal of this project was not to try to beat the state of the art algorithm [1], but implement a similar architecture, try different parameters and learn the effect of these changes to the accuracy.

After all the tests and parameters set tried the best accuracy achieved was 0.8612 on the testing set based on 9 epochs, batch size of 64 samples, 'sum' as the merge mode, 3 LSTM layers and a embedding size of 64.

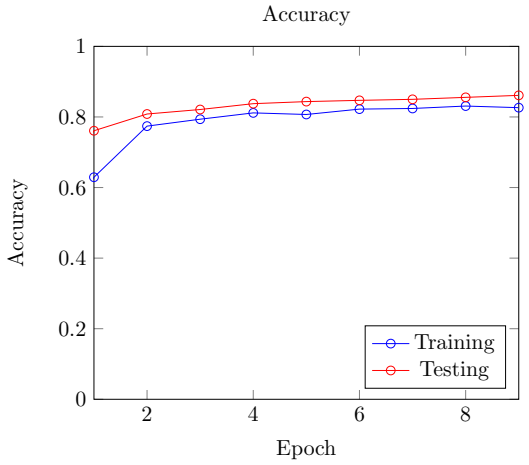

Figure 1: The plot shows how the training and the testing accuracy improves after each epoch.

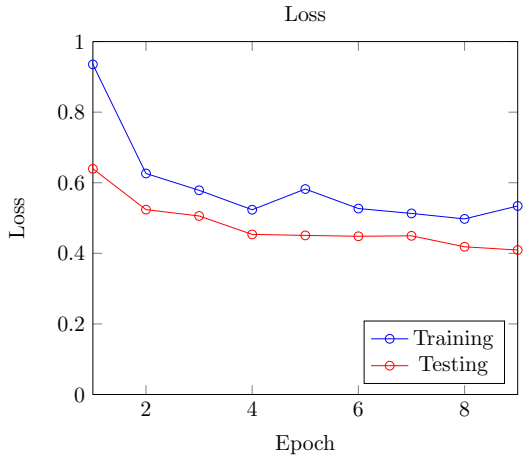

Figure 2: The plot shows how the training and the testing loss decrases after each epoch.

References

[1] J. Ma, K. Ganchev, and D. Weiss, "State-of-the-art chinese word segmentation with bi-lstms," *CoRR*, vol. abs/1808.06511, 2018.