Chinese Word Segmentation

1st NLP homework

24 April 2019

Student: Anxhelo Diko
Matricola: 1867167
Course: Compute Science LM-18

Professor: Roberto Navigli

Introduction

Replication of State of the art Chinese word segmentation. The task is tough and one of the biggest problems is dealing with out of vocabulary words. We are offerd four datasets in chinese, two of them in simplified chinese and two in traditional chinese (these datasets needs to be hanzi converted)

Preprocessing

In the preprocessing I firs create the unigrams and bigrams. Then I make vocabularies out of them . I use the vocabularies to create feature vectors for all data which takes part to training, testing and prediction. For testing and prediction the unknown instances are replaced with 0 ("UNK":0). The next step is Padding and I use the size of 30. These steps were for X-data. The Y-data is first generated from the original file by giving a label to each char based on the location then it is label encoded. The next step for it is padding and after that one hot encoding representation of each label. In general the preprocessing has as goal to produce 3 output files unigram_features, bigrams_features and the one hot encoding version of y. Unigrams and bigrams feature vectors are used for the input layer and the one hot encoding values as target.

Dataset

The datasets I use are as, msr and pku. I first joined the as and msr to create a dataset with 800.000+ samples for training (as is hanzi converted). Fore dev and test I used both msr_gold and pku_gold.

Model

I choosed to use the stacked BiLSTM model. The architecture consist of two separated embedding layers(plus one input layer for each), one for bigrams and one for unigrams, these two layers are then concatenated with keras.layers.concatenate to be used as input for the first LSTM layer. The output of the first LSTM is then passed to the second LSTM and then to the time distributed dense layer which takes an input of 4 values(one hot encoding size=4) and uses a softmax activation function. The optimizer used for this model is adam and for feeding the data to the model I used a batch generator function to avoid memory problems and it went out that with this method my model ran faster.

Hyperparameters

Hyperparameters makse the difference almost in every deep learning task. The model I built for this homework had a lot of hyperparameters which needed to be tuned. For tunning them I tried to do a GridSearch using the package telos but not for all of them. The most crucial hyperparameters which made the difference on my model were embedding size, learning rate and the number of epochs.

Evaluation

My best model had 91~92% training acc. , 90~91% val. Acc., 0.105 training loss and 0.22 val. Loss. It`s score on the score.py file was in avarage 89% depending on the used dataset.

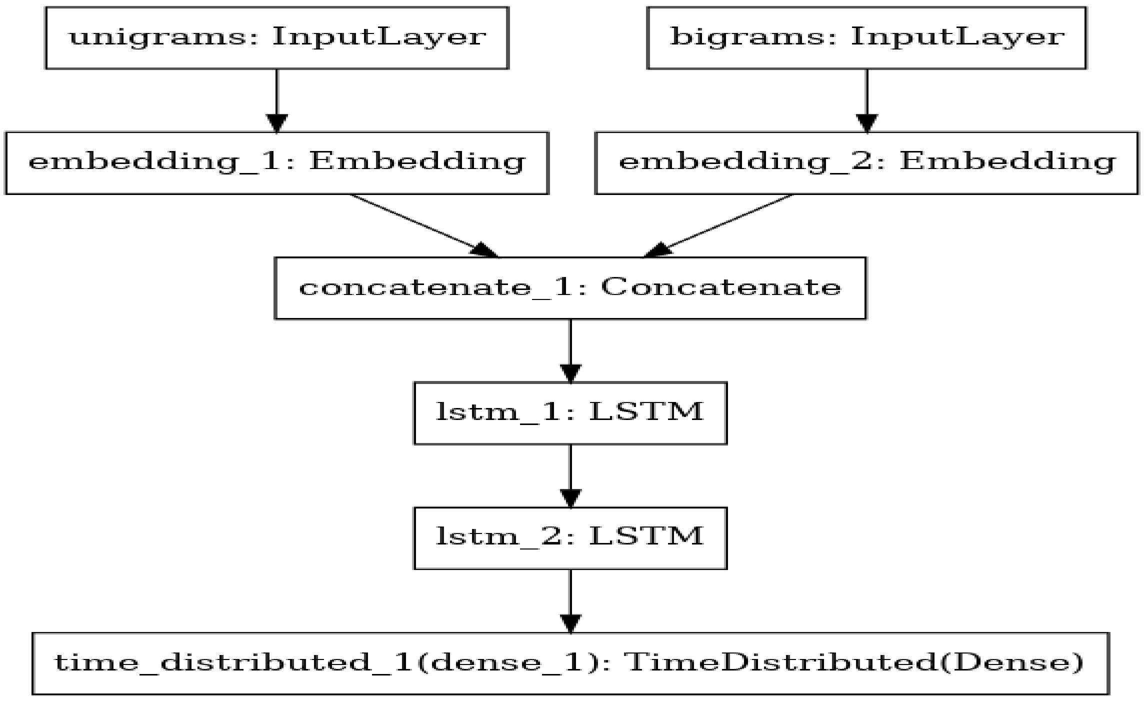


Illustration 1: Model architecture generated by Keras

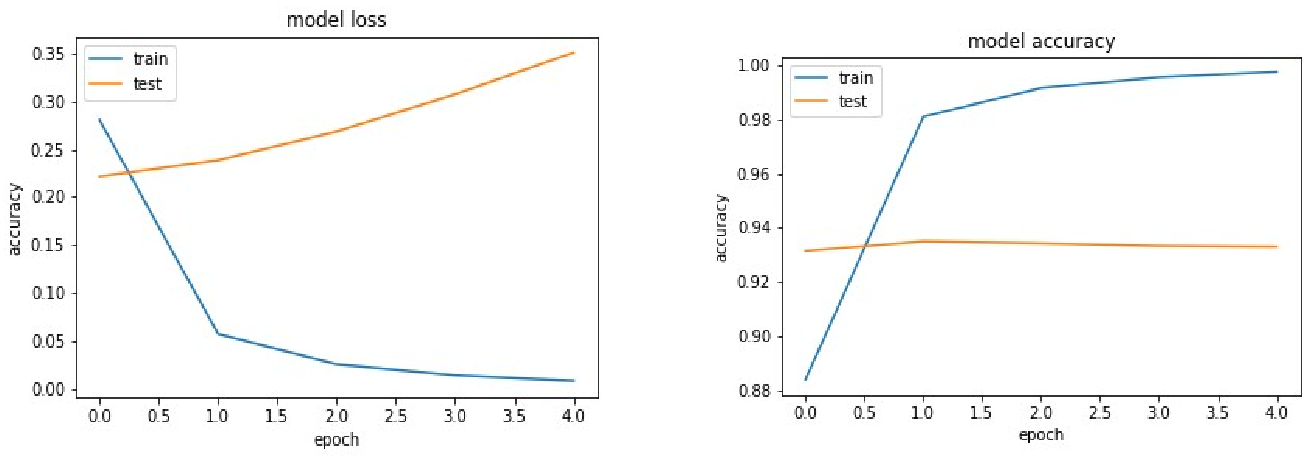
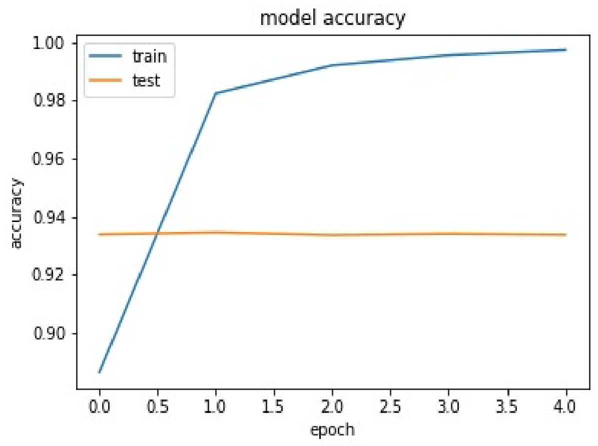


Illustration 2: Model overfitting with 5 epochs and 800.000 Data Samples batch size 64



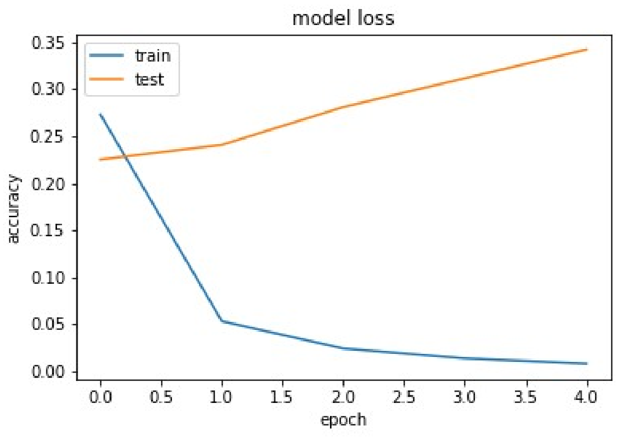


Illustration 3: Model overfitting with 5 epochs and 800.000 Data Samples batch size 128

The model have all my final parameters except the number of epochs. The two previous models were very poor in testing even though they had 93% accuracy on validation, on precision they were above 70%.

The final model metrics are those in the Evaluation Section.

Embedding Layer − Unigrams = 64	*Recurrent dropout = 0.25*
Embedding Layer − Bigrams = 32	*Batch size = 128*
Learning rate = 0.005	*Hidden layer size = 264*
LSTM dropout = 0.3	*Padding size = 30*

Table1: Hyperparameters list