NAME: GARBA MARIAM MATRICOLA: 1871871

1.0 APPROACH

In this project, four chinese datasets (AS,CITYU,MSR and PKU) were used to train the deep learning model for chinese word segmentation task.
First, the AS and CITYU were converted to simplified chinese with the HanziConv tool since they are in traditional chinese format. MSR and PKU are already in simplified chinese format.
These datasets were combined and a preprocessing technique was performed on them two generate an input file with no spaces between the chinese characters and a label file in BIES format. The BIES format is a way to encode the output of a word segmenter model. There are four classes the model has to predict: B (Beginning),I (Inside),E (End) and S (Single).

A unigram vocabulary was also built from the combined dataset to index all chinese characters. These vocabulary is then used to encode the input file in a word to index format which would be fed into the model since the model can only interpret numbers and not words. This encoded input file was then used as the training set (X) for the model.
From the generated label file, the labels in BIES format were also converted to integers: B (0), I(1),E(2), and S(3) and one hot encoded. This one hot encoded labels was used as the output of the training set (Y).

Since each sequence in X has variable lengths, all sequences in X and Y were padded with zeros at the end with the length of the the largest sequence in X. It is this padded sequence of X and Y that would be fed into the BiLSTM model that would be described shortly. This preprocessing step can be seen in Figure 1.

In the BiLSTM model, the characters are embedded using the default keras embedding with a variable input shape, 256 hidden units and 4 time distributed dense layers with softmax activation for the output of the model. Dropout regularization was applied to the input and LSTM model on order to aid the model with generalization on new instances. Stochastic Gradient Descent optimizer with momentum was used to compute the loss of the model and also varying the changes between the Accuracy and Precision metrics during training for the training set and development set. Masking was also activated during the training so that the padded zeros in X and Y won't be evaluated when computing the model loss. To speed up the learning process, the sequences were processed in batches and the model with the highest precision throughout the epoch is saved as the best model.

Motivations of the hyperparameters that was used were gotten from Ji Ma, Kuzman Ganchev and David Weiss, EMNLP 2018. Grid search couldn't be performed during the training because there wasn't enough computing resources.

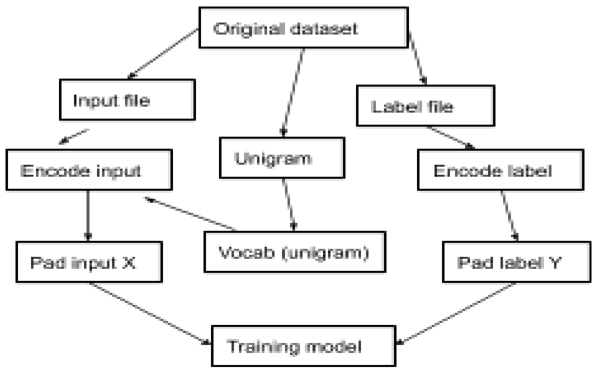
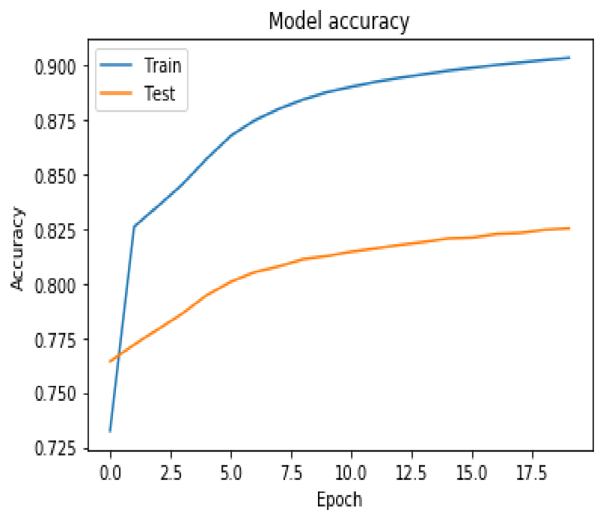


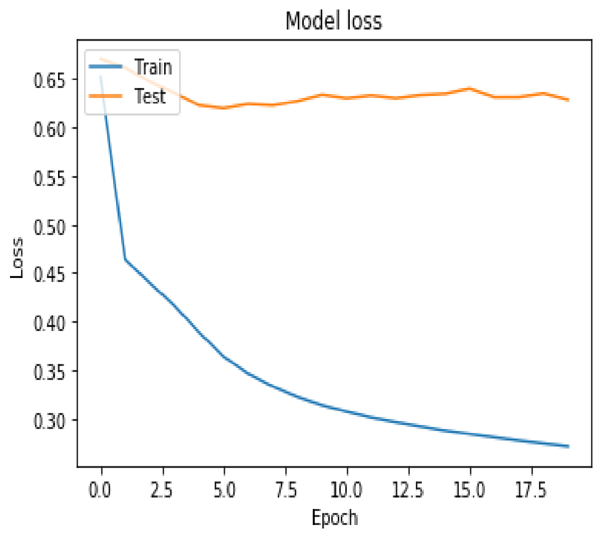
Figure 1 (Preprocessing step)

2.0 RESULT ANALYSIS

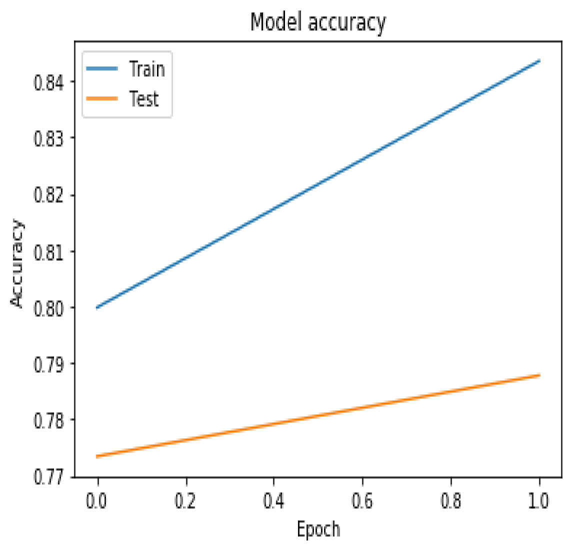
The table below shows the hyperparameters value that was used for **SGD and RMSProp optimizer.** The chart that follows also shows the loss and accuracy of the training. The first two charts is for SGD (20 epochs) and the last two chart is for RMSProp (2 epochs).

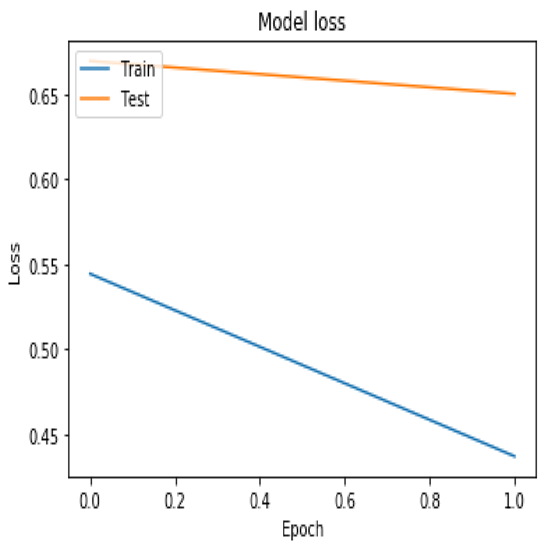
Hyperparameters	Values (SGD)	Values (RMSProp)
Char embedding size	64	64
Learning rate	0.035	0.001
Batch size	128	256
Momentum	0.95	-
Input and RNN Dropout	0.6, 0.4	0.6, 0.35
RNN hidden size	256	256





The chart that follows also shows the loss,precision and accuracy of the training - for R





After changing the values of the hyperparameters of this two optimizers, SGD accuracy performed better than RMSProp at the second epoch. (RMSProp could not be trained for more than two epochs due to resources and time constraints).

Scoring the result of the SGD model on the gold set gives a precision of about 50%. I would like to point out that although the models did not converge,but perhaps with more training and efficient resources, this precision would be higher than this.