Livia Biggi – 1793434

Chinese Word Segmentation using Bi-LSTM

Dataset

The dataset used was the result of the concatenation of the four training datasets (AS, CITYU, PKU and MSR) – with AS and CITYU having been converted to simplified Chinese beforehand. Given the size of the resulting dataset and the number of duplicate sentences in it, this has been further reduced to only include unique sentences consisting in no more than one hundred words (reducing the size of the dataset by 7 and 0.5%, respectively).

Of this, only 60% of the sentences (taken uniformly at random) were kept, and further split into the train, validation, and test sets – accounting for 70, 20 and 10% of the total size. The train and validation sets have been used during the training of the model, whilst the test has been used to verify the precision of the model predictions.

Pre-processing

Given the input file (the concatenation of all four datasets) in simplified Chinese, the first step of the pre-processing part consisted in converting all digits, punctuation and Latin letters to half-width, following the approach used by Ma et al. (2018). Then, all words were stripped of (potential) punctuation, thereby eliminating the risk of considering punctuation marks – such as parentheses – part of the word itself. Only the most used marks in Chinese were used to check the punctuation; for instance 、 and 「 were kept, whilst non-language-specific symbols were ignored (e.g. ▲ and →).

The labels were generated according to the word length and saved into a separate file. Finally, white-space was removed to create the input file to the model.

Model and Training

Separate vocabularies of characters' unigrams and bigrams were created, and all sentences were padded according to the length of the longest sentence in the trainset. The gold file in the BIES format was transformed into one-hot encoded labels.

The Bidirectional LSTM model used in the project was developed using Keras. It takes two inputs: the unigrams and bigrams of characters at each position, and generates their embeddings, which are then fed to the Bidirectional LSTM comprising two stacks. The output of the LSTM layer is then passed to a Dense layer with a Softmax activation function. The same model has been tried with different optimizers: SGD (with momentum = 0.95), Adam and Adagrad. Although SGD is used in the paper by Ma et al., (2018), Adagrad was chosen because the model generates its embeddings, and since Adagrad uses a different learning rate for each parameter, rarer words will be updated with a higher learning rate than frequent words. Furthermore, it also proved much faster to train the same model using Adagrad, rather than SGD. Details and comparisons of previous models, as well as their hyperparameters can be found in the following pages.

Evaluation

The evaluation of the model has been performed on the test set created during the pre-processing phase, consisting of just under 50,000 sentences. The model predictions are performed depending on the sentences' lengths in the test set: if at least one of them is longer than the maximum length allowed in the model (256), then the indices of the sentence(s) that exceed the maximum length are saved, and the sentences split into a number of sub-sentences. After generating unigram and bigram features according to the vocabulary used in the model, all sentences are padded, and the prediction on the whole file is performed. Then, the sub-sentences are converted into the BIES format, concatenated together to form the input sentence, and the padding is removed. The final precision of the model on the test set is 0.935.

Tables and Figures

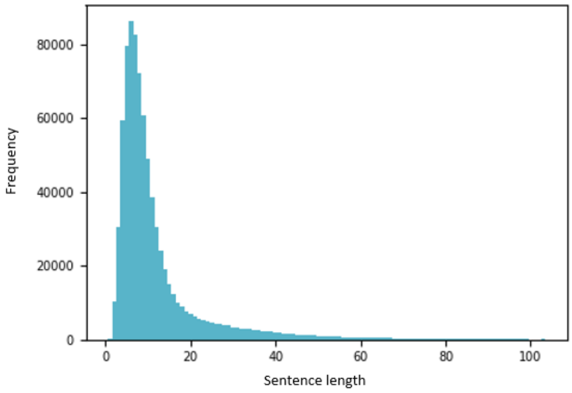
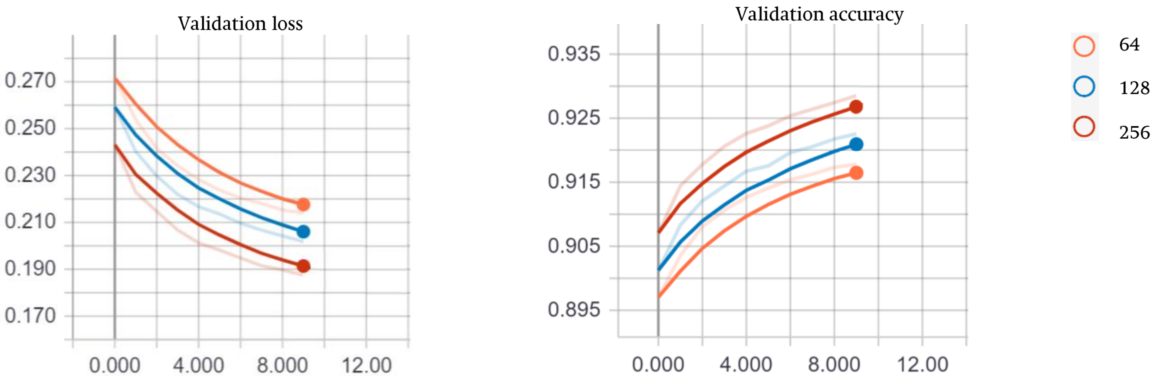


Figure 1: Distribution of the length of the sentences in the dataset

Embedding size	[64, 128, **256**]
Batch size	[128, **256**]
Learning rate	[0.005, **0.01**, 0.025]
Optimizer	[SGD+momentum, Adam, **Adagrad**]
Hidden size	[128, **256**]
Input dropout rate	[**0.2**, 0.4]

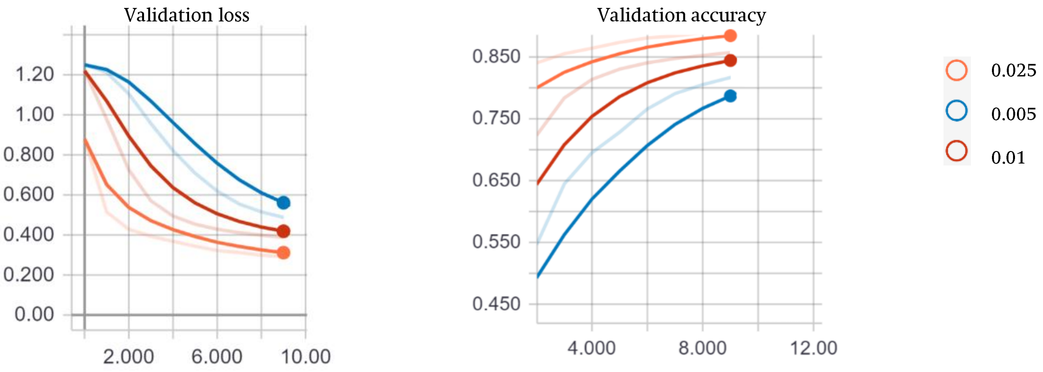
Table 1: Hyperparameters settings. The bold values are those used in the final model

Embedding size



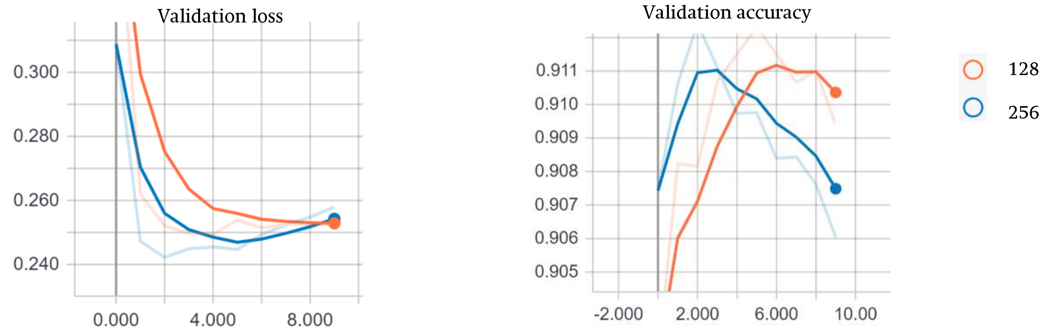
The plots show the differences in the validation loss and validation accuracy at training time for models with different embedding sizes (64, 128, 256), ceteris paribus. The same embedding size was used for both unigrams and bigrams.

Learning Rate



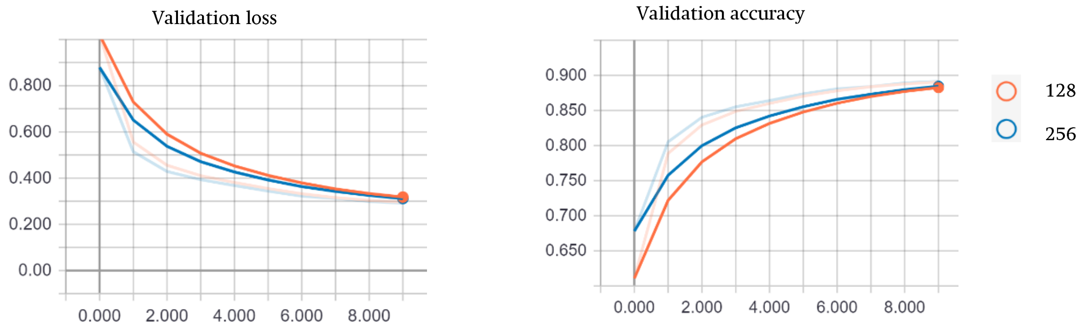
The plots show the differences in the validation loss and validation accuracy at training time for models with different learning rates – in the case of the SGD with momentum = 0.95; (0.005, 0.01, 0.025), ceteris paribus.

Batch Size



The plots show the differences in the validation loss and validation accuracy at training time for models with different batches – holding everything else the same. Although the models are both overfitting, it is noticeable that the one using batches of 256 reaches a high level of validation accuracy earlier. A batch size of 256 was then used in the final model to speed up the training phase.

Hidden Size



The plots show the differences in the validation loss and validation accuracy at training time for models with different hidden sizes, ceteris paribus. Even though the models show a similar performance, the one where a hidden size of 256 performed better across all epochs, and was thus chosen for the final model.

References

Huang, X., Jiang, J., Zhao, D., Feng, Y., Hong, Y., (2017): Natural Language Processing and Chinese Computing. *Proceedings of the 2017 NLPCC 6th CCF International Conference*

Ma, J., Ganchev, K., Weiss, D., (2018): State-of-the-art Chinese Word Segmentation with Bi-LSTMs. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4902–4908. Association for Computational Linguistics

Wong, K., Li, W., Xu, R., Zhang, Z., (2009): Introduction to Chinese Natural Language Processing. *Morgan & Claypool Publishers*

Zhou, H., Yu, Z., Zhang, Y., Huang, S., Dai, X., Chen, J., (2017): Word-Context Character Embedding for Chinese Word Segmentation. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 771-777. Association for Computational Linguistics