NLP Assignment 1

Francesco Pieroni, Rooshan Saleem Butt, Umberto Altieri

Abstract

In this document we describe the results we have achieved while tackling the POS-tagging problem described in assignment 1 notebook. Starting from the most basic ones, we have built a total of nine models with different architectures, and using f1-score as metric, we reached a value of 0.86 on the test set.

1 Task description

A POS tag is a label assigned to every word in a text to indicate its part of speech. In automatic POS tagging it is required to have a tool capable of assigning POS labels to the words in a text which is often very long. Following the guidelines contained in the notebook, we approached the problem using the embedding model "GloVe" and developing some neural networks based on a simple one with a Bidirectional LSTM and a Dense layer.

2 Analysis of the data

All the data have been extracted from the text files inside the archive available at this link, and then split in training, validation and test sets according to their document numbers. Then wee decided to split each document in sentences, which will be the input of the models. Taking a look at the distribution of data, we could see that some labels are much more common than others, but we decided not to alter the distribution, mainly because some labels (e.g. "NN", "VB", ..) are more frequent than others in English texts. Starting from GloVe vocabulary, we checked OOV words and got a huge number (2346) of train set's words OOV, but applying some preprocessing techniques such as lower-casing and character replacement, we were able to cut the number down to 318. Then we built the embedding matrix using GloVe embeddings of dimension 50 and handling OOV words in the following way:

- Compound words: since many OOV words are compound, we assigned them an embedding equal to the one of their longest non-OOV word. For instance, "forest-products" will be associated with the embedding of "products".
- Compound words with no subwords in the vocabulary or non-compound words: they are assigned with a random embedding.

In addition, we encoded all the sentences and related labels using word and label mapping from word to indices, and finally standardized their length applying post-0-padding, otherwise inputs would have a different size.

3 Description of the models

Using keras layers, we created the baseline model with a non-trainable Embedding layer, a BiLSTM and a Dense layer. After that, we tried the indicated variations, namely: using a GRU instead of the LSTM, adding an additional LSTM layer and adding an additional dense layer (Note that all of these variations have been kept separated, so not mixed). After that, we realized that the best model by f1-score on validation set was the baseline one, so we tried to improve it with several changes, such as using GloVe embedding dimension=300, increasing the number of LSTM hidden units from 100 (value used in the baseline models) to 200, adding a dropout layer and/or dropout on the LSTM layer. In addition, all the models have been trained using categorical crossentropy as loss, because the labels were one-hot-encoded.

4 Results

We have computed the f1-score on the validation set and accordingly determined the baseline model as the best one, even though all the values are very similar (see Table 1).

	Baseline	GRU	Add. LSTM	Add. Dense
f1-score	0.73387	0.73238	0.73069	0.72315

Table 1: f1-scores of base models on validation test

In order to improve the best model, we have made some modifications to it and reached an f1-score of 0.8112 using GloVe embedding dimension of 300, 200 LSTM hdden units, a Dropout layer and dropout in the LSTM layer. Similar results were obtained without increasing the number of LSTM hidden units, which makes these two the best models. Evaluating their f1-score on the test set, we obtained a value of 0.8561 for the first one and 0.86881 for the other. Probably this is due to the fact that the second model is not over-fitting, while the first one a little bit. The over-fitting problem is the main reason for the smaller f1-scores of the base models, which is why we introduced Dropout.

5 Error analysis

It can be seen from the confusion matrix of Figure 1 that NNPS, PDT, RBR, JJR are the classes most misclassified in the test set. NNPS are often classified as NNP and NNS, probably because of lowercasing and OOV handling. Furthermore, PDT are often confused with adjectives (JJ), determinatives (DT) and surprisingly verbs (VB). Most probably, PDT support is just too small. Finally, RBR and JJR are both comparatives and have a small support, which makes their classification unreliable, nearly random.

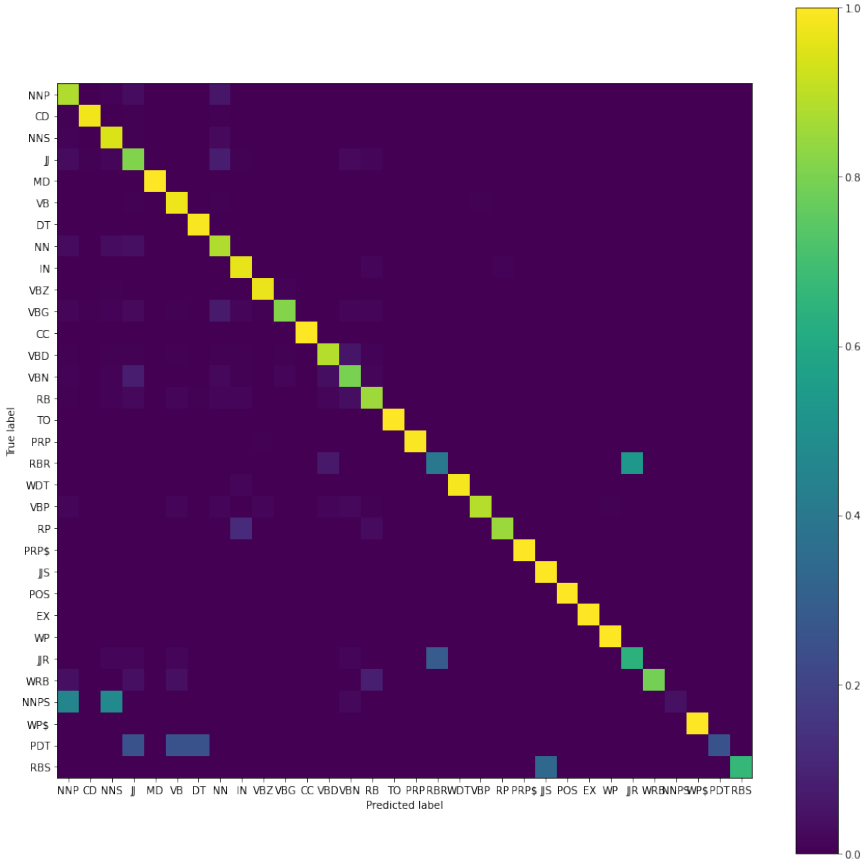

Figure 1: Confusion Matrix