Assignment 1: POS Tagging

Serban Cristian Tudosie
serban.tudosie@studio.unibo.it

Francesco Vannoni
francesco.vannoni2@studio.unibo.it

Mirko Del Moro
mirko.delmoro@studio.unibo.it

Abstract - Recurrent Neural Networks have been shown to be very effective for tagging sequential data, e.g. speech utterances or handwritten documents. In this study, we propose to use several Neural Network for part-of-speech (POS) tagging task. The different combination of models exploits one or more RNN layers starting from the GloVe embeddings of the sentences words.

1 OUR APPROACH

1.1 Dataset Cleaning

The dataset provided by NLTK contains 200 dependency treebank files. We have prepared the dataset to train the models on sentences. By using regular expressions we have cleaned the file from unwanted blank spaces, tabs, and new lines. We split the dataset into train, validation, and test. The first 100 documents from the dataset are used as the train set, the following fifty as the validation set, and the remaining as the test set. Analyzing the dataset we have found that the average length of the sentences is around 25, and the variance is small. Therefore we look for outliers, and we decide to remove from the training and validation set those having a length higher than 80.
Sentences are then tokenized by assigning to each word an identifying integer. Since we have variable length sentences we pad each sequence and we reserve the 0 token for the padding.

1.2 GloVe Embeddings and OOV Handling

After that we have computed the embedding matrix for sentence terms. To get the embedding we have used GloVe vocabulary but we have to handle the dataset's terms not contained in GloVe. Out of vocabulary embedding is affected only by the words of the split it belongs to, while it is independent from the words of the other splits.
We have computed OOV terms embedding by averaging on the embeddings of all their previous and next word from the sentences of the split the words belong to. If all the neighbours are OOV terms the average can't be computed so the embedding elements are picked from a uniform distribution. We have decided to assign the same kind of embedding also to those terms having only one neighbour contained in the vocabulary, otherwise the two embeddings would have been the same. This process must be done in order for train, validation and test. The result embeddings are concatenated to the embedding matrix and the terms are added to the vocabulary. This allows to have an embedding matrix where: if a word is present in both train and test set and not in GloVe vocabulary, the embedding is computed as OOV when the training set is processed, and it is considered as a vocabulary word when the handling of the test set takes place.

1.3 Training and Evaluation

The embedding matrix is used in order to train four models with different architectures shown in the next section (Section 2). Those models are made out of combinations of layers but all of them contain at least one RNN layer.
We have trained the models for 10 epochs with a batch size of 32 and we evaluate them on the validation set by using the f1 macro scores in order to select the two best models. Then we have performed Bayesian Optimization hyperparameter tuning on the selected ones. In particular we have tried several combinations of number of units and l2 regularizer value. Lastly we have trained the best models for 50 epochs with a batch size of 32 using "Early Stopping" in order to prevent overfitting. Then we have evaluated them on the test by using the f1 score not considering the punctuation tags.

2 MODELS

We have tried four model architectures in order to obtain POS tagging:

1. two layers architecture: a Bidirectional LSTM layer and a Dense layer on top (baseline)
2. two layers architecture: a GRU layer and a Dense layer on top (gru)
3. three layers architecture: a Bidirectional LSTM layer, a LSTM layer and a Dense layer on top (two lstm)
4. three layers architecture: a Bidirectional LSTM layer, two Dense layer on top (two dense)

The inputs are the tokenized sentences of the training set, the outputs are the POS labels. The activation function of the top Dense layer is "softmax"; in the last model (where there are two dense layers) the other dense layers has

"Relu" as activation function. As shown in Section 1 the l2 regularizer value and the number of RNN units is tuned through a Bayesian Optimization hyperparameter tuning.
The optimizer chosen is Nadam with an initial learning rate equal to 0.01; the loss function is "categorical crossentropy". In order to prevent overfitting we have exploited "Early Stopping" monitoring the validation accuracy.

3 RESULTS EVALUATION

In this section we are going to evaluate the findings obtained. The left histogram shows the scores obtained with the models trained for 10 epochs on the validation set by computing F1 macro metric. As we can see in the Figure 1, baseline and gru are the models with the highest scores on the validation set. We have trained gru and baseline for an higher number of epoch (50) and we have used the resulting models on the test set.

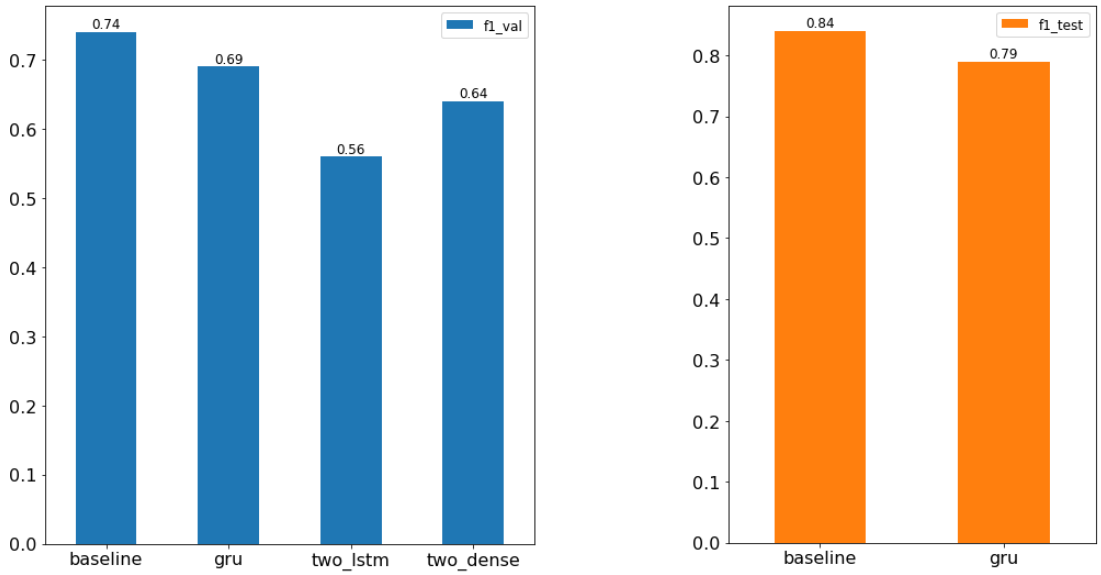
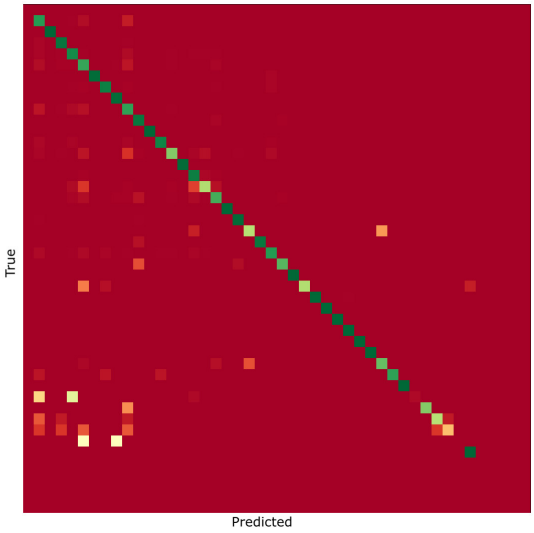


Figure 1: Models F1 score on validation set (left histogram). Best Models F1 score on test set (right histogram)

Analyzing epoch by epoch the scores on the training set we can observe that the gru is faster to converge. Gru has less parameters to train than two_dense and probably by training on an higher number of epochs than 10 it could overtake gru. However analyzing loss and accuracy during training and validation, we have observed that it is unstable and tends to overfit, so this is why we have chosen gru and baseline for hyperparameter tuning. As expected the best one is baseline, the one having an higher number of parameters. Both models did not train for fifty epochs because Early stopping interrupted the train after respectively 36 and 44 epochs.

4 ERROR ANALYSIS



As shown in the previous section we have reached higher performances but there is still a percentage of errors. Therefore we have tried to find where the errors are. The figure on the left shows the heatmap of the confusion matrix computed on the test set. If an element is on the diagonal it means it is a correct prediction. The darker the green, the higher is the percentage of correct prediction. In the lower part a lot of rows are completely red because the corresponding POS tags are not present in the test set.
Above that we can see some mistakes. In particular there are three tags that the model is not able to classify: "nnps", "-rrb-" and "pdt". We observed that these categories rarely appear in the training set, so maybe this is one of the reason of the mistakes. In addition to that, "nnps" is often classified as "nns" or as "nnp" so the model is able to tag it as a noun but it confuses proper with common nouns and plural with singular.
"pdt" is classified as "jj" or as "dt", tags that usually are followed by a noun. Lastly "-rrb-" is well classified only in the 33% of the cases, sometimes it is confused with "-lrb-", so the model is able to recognize a parenthesis but it can't spot the difference between left and right parenthesis.