Part-of Speech (POS) tagging as Sequence Labelling using Recurrent Neural Architectures

Andrea Proia, Federico Spurio, Cristian Urbinati

December 2021

Abstract. This work aims to accomplish the POS tagging task using different RNN architectures on a corpus. In particular, we have evaluated the performance of Bi-LSTM and Bi-GRU combined in different ways. Based on the fact that the f1-score of all these models is similar, we have concluded that the majority of the work is carried by the recurrent layers. We obtained good result: most of the tags are predicted correctly and the majority of the misclassified are grammatically similar to the correct tags.

Introduction

The task consists on assign a POS tag to each word in a sentence based on its grammatical meaning in that context. We use GloVe pre-trained model to obtain a vector representation of the semantic relation between words.
The goal is to exploit some RNN architectures to learn common patterns and then use a Dense layer as classifier to predict the corresponding sequence of tags. We then choose the two best fine-tuned models using the f1-macro score.

1 Dataset preprocessing and data conversion

The dataset we deal with is the dependency treebank corpus composed of a list of documents, which already contains the associations between words and their relative POS tags. We decide to split each document into sentences and we load them into a dataframe, so that for each row we had the sentence words, the corresponding list of tags and to which set they belong to (train/valid/test).
We make the choice of splitting into sentences since, in most of the cases, the grammatical meaning is bounded to the sentence, thus it's not really useful to consider more than that. Moreover, we believe that shorter sequences were easy to process and therefore could lead to less training time. Then, we manage to cast all the words to lowercase so that they match with the pre-trained word embeddings.
Next step was to build the vocabulary and the embedding matrix, the former associates to each word an index, the latter stores at each row the embedding vector of the word matching that index. We manage out of voabulary (OOV) words associating them a random embedding vector.
Finally, we pad all the sequences of words to a maximum length by means of the string `-PAD-` encoded with 0 in the vocabulary. The same padding operation is done for the labels, which are then directly converted in a categorical format. The maximum length of the sequence is determined by training set, emulating a real scenario where we can't prevent to have longer sentences at test time. Such sentences are truncated.

2 Model definition and training

The baseline model is a Keras Sequential and contains in order: an Embedding layer to which we pass the embedding matrix (not trainable), a Bidirectional LSTM that returns the entire sequence of outputs that are then passed to a Dense layer wrapped by a Time Distributed layer. We also experiments using a GRU instead of the LSTM, adding an additional LSTM or another Dense layer.
For the train we use the categorical crossentropy loss and, as evaluation metric, the categorical accuracy. We use KerasTuner[1], a general-purpose hyperparameter optimization framework based on different tuning algorithms. In particular, we tested our models with

Bayesian Optimization in order to optimize: LSTM/GRU units, Dense units and the learning rate by minimizing the validation loss.
Even if the model does not quickly overfit, we decide to implement an early stopping as regularization technique, to stop the training as it is get worse, and do not to train the model for too many epochs.

3 Performance evaluation

In fig.1 we can see the training process of different models in relation to the number of epochs. We noticed that the model2 needs more epochs, this is unusual since the GRU should be faster than the LSTM. We have seen that this happens because the learning rate predicted by the KerasTuner for that model is lower with respect to the other (0.005 against 0.02 of the model1).

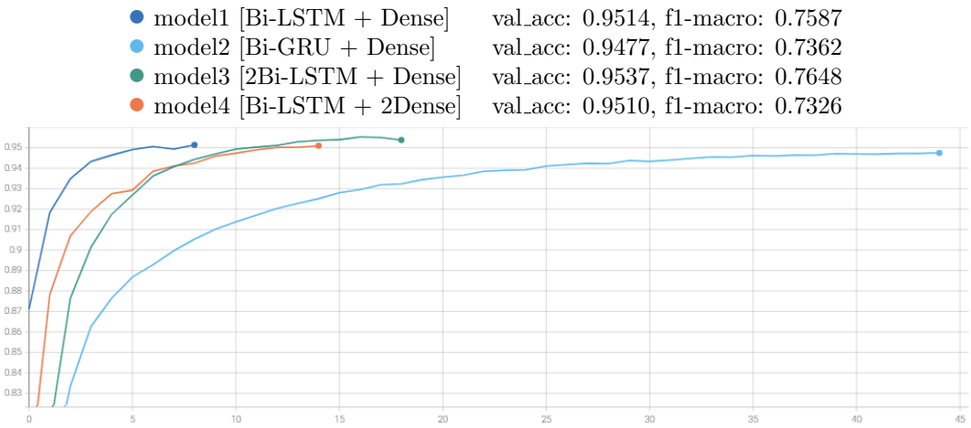

Figure 1: Categorical accuracy scores on validation set

We finally choose to evaluate on test set the two models with higher f1-macro score on the validation set, ignoring punctuation. Below, the accuracy and f1-macro score calculated on the test set:

Model	accuracy	f1-score
2Bi-LSTM + Dense	95.33%	0.8324
LSTM + Dense	95.56%	0.8574

4 Conclusions

Despite the small variations to the baseline model (LSTM + Dense), all the models tested present similar performances and f1-scores. The main reason is that the majority of the work is carried out by the first LSTM (or similar architectures like GRU).
There are two main limitation of our model:

- due to the Embedding layer, there is a fixed maximum length for the input sentences determined at training time; this cause our network to works only with sentences of length smaller or equal than this size.
- when the input sentence is composed mostly of OOV words; the choice of assigning random vector embedding to them could lead the network to produce random POS tags. We could limit this problem by choosing a more appropriate embedding for OOV words or using a bigger GloVe dataset.

We obtain overall good results, we observe that most of the tags are predicted correctly and the majority of the ones that are misclassified are grammatically similar as we can see in the example below from the test set:

```
['DT', 'JJ', 'NNP', 'NNP', 'NN', ',', 'RB', ',', 'VBD', 'DT', 'NN', ...]
['DT', 'JJ', 'NNP', 'NNP', 'NNP', ',', 'RB', ',', 'VBD', 'DT', 'NN', ...]
```

Listing 1: Comparison between true and predicted label of a test sentence

References

[1] Tom O'Malley et al. *KerasTuner*. `https://github.com/keras-team/keras-tuner`. 2019.