Natural Language Processing
Report for Assignment 1

Davide Sangiorgi, Valerio Tonelli, Falco Riccardo,
Yellam Naidu Kottavalasa

Academic Year 2021-2022

Abstract

We have implemented Part-Of-Speech (POS) tagging with recurrent neural networks using sentence padding and masking, GloVe embeddings and trying out different combinations of hyper-parameters such as number of nodes and network structure, obtaining a final Macro-F1 score of 81.5% on an unbalanced and small-sized dataset for our best model, and of 80.4% for the second best one.

1 Introduction

POS tagging is an NLP process which marks each word in a sentence with a tag describing the grammar/meaning of that word. It plays a role in tasks such as word disambiguation, as well as in creating an explainable language model.

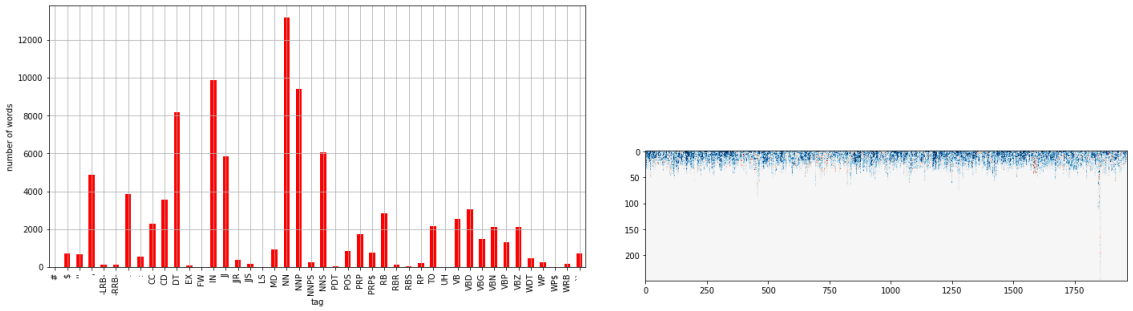
In the following, we demonstrate how Penn Treebank tags can be assigned to a dependency treebank provided by NLTK via a Recurrent Neural Network and discuss implementation details, results, solved pitfalls and potential improvements.

2 Preprocessing

The dataset is a small and unbalanced corpus (see Figure 1) composed of 200 documents, each of which contains one or more sentences. Each word in each sentence is associated with one among 45 different tags. Train/validation/test splits are defined based on the document number.

Preprocessing has been carried out as follows: (*i*) Removal of the `SYM` tag, since it appears only once in the train set, replaced with an "and" of tag `CC` (`UH` and `FW` tags are also infrequent, but since they are meaningful part-of-speech we have left them); (*ii*) Replacement of words with tags `-LRB-` and `-RRB-` with parentheses symbols, since we are using fixed embeddings (GloVe) which recognize the latter but not the former; (*iii*) Lower-casing, because of the smaller dimensionality of GloVe embeddings with respect to the case sensitive ones, which we considered excessive for our relatively small dataset.

Vocabulary creation has been done directly from words to their embeddings with a dimensionality of 50; Out-Of-Vocabulary terms have been handled by assigning to them a random vector with small magnitude.



(a) *Histogram of tags counts.*

(b) *Channel 1 of word embeddings, where each column is a sentence. Notice the prevalence of white space, which are padded words.*

Figure 1: *Characteristics of the input dataset.*

3 Models

We have trained and tested different neural networks, according to the assignment specification. All models take inputs of shape `(batch_size, max_seq_length, n_dims)` where `n_dims` is the dimension of a word embedding and `max_seq_length` is the longest sentence in the training set. Any sentence in the validation or test set which is longer has been truncated, and smaller sentences have been padded with zeros at their end. The network produces outputs of shape `(batch_size, max_seq_length, n_tags)`, which are probability distributions over tags in the last dimension.

All models have the following structure: (*i*) A non-trainable Masking layer, so that zero embeddings are effectively ignored; (*ii*) A Bidirectional Recurrent layer, such as LSTM or GRU; (*iii*) (Optionally) an additional LSTM or dense layer; (*iv*) A final dense layer with softmax activation.

Further parametrization of the model include the choice of learning rate, number of nodes for the hidden layers, and whether to use L2 regularization for hidden layers. In total, 192 models have been trained and evaluated. Early stopping has been used in all models to avoid over-fitting.

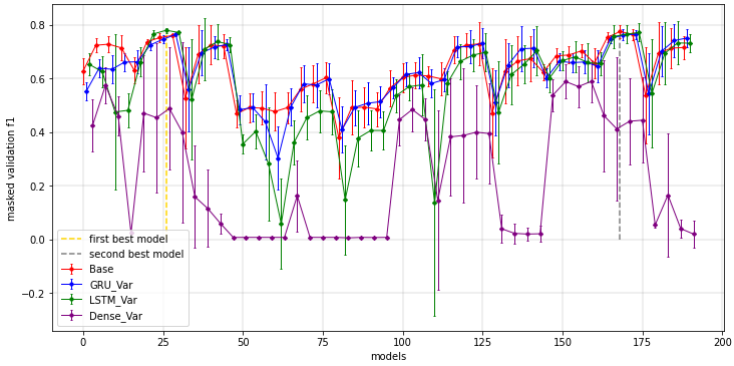


Figure 2: *Performance of the four model variants with respect to different combinations of hyper-parameters.*

4 Evaluation

Our best-performing models (in terms of Macro-F1) achieve scores of 81.5% and 80.4% respectively, using the following hyper-parameters: (1) 64 nodes, learning rate of 0.01, no regularization and an extra LSTM layer; (2) Default LSTM layer with no additional layers, 64 nodes, learning rate of 0.01 and L2 regularization with a weight of 10^{-5}. These results are significantly better than the classifier which assigns to each word its most frequent tag, which achieves a 73% score. Scores are also coherent between test and validation sets, as they are within 5% of each other for our best models, and are actually better on the test set. This is likely due to the unbalanced nature of the dataset: indeed, accuracy is about 10% higher than F1-score, further supporting this claim.

Over all tested models, generally the best number of nodes is 64 and the best learning rate is 0.01; L2 regularization generally helps when using a weight of 10^{-5}. With reference to Figure 2, among neural network types the variant with the added dense layer tends to perform worst, with the other three being quite similar to each other.

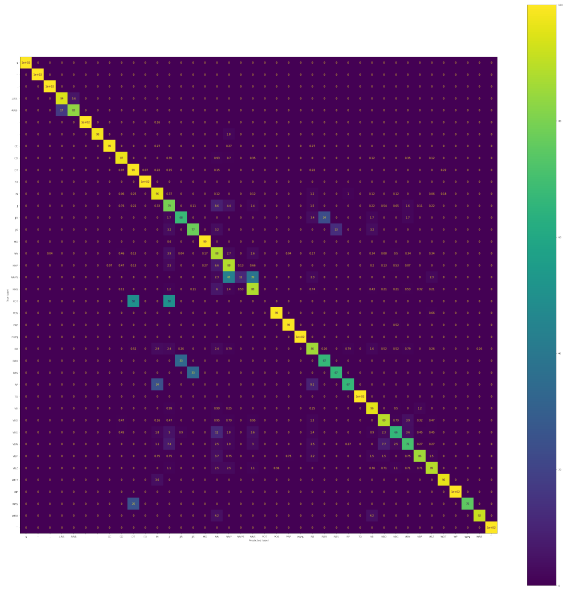


Figure 3: *Confusion matrix over tags for test set.*

5 Error Analysis

A couple of mistakes of our original design include how padding was handled, which was initially to use a tag of its own. This would drastically mask true performances due to padded words constituting more than 90% of the input, as it can be seen from Figure 1. A second error that we fixed was to apply L2 regularization to all layers, including the last one: since the output is a probability distribution, regularization was incorrectly bringing all values towards zero.

It is also interesting to look at performances with respect to different classes, as it can be seen from Figure 3: `FW` and `UH` are never predicted correctly in the validation set (they don't appear at all in the test set), again due to the small number of instances in the train set. Infrequent classes such as `RB`, `RP`, `WP$` under-perform and `NNPS` also does, most likely due to not using case sensitivity. As a positive note, we do see clustering of guesses among similar tags, such as different noun or verb types.

6 Conclusion

Our models manage to obtain good results, although far from being State-of-the-Art, by using a simple neural network structure that improves upon a non-contextual algorithm such as the most frequent classifier. Potential improvements to our model include using case sensitivity, a loss that devalues errors among similar tags, and studying whether reducing or increasing the number of embedding dimensions, for instance by using PCA, would have made a difference.