Assignment 1 report: RNNs for POS tagging

Campardo Giorgia, Chinellato Diego, Fanti Pietro, Longhi Carlo
Alma Mater Studiorum - Master's Degree in AI, Natural Language Processing course

Abstract

In this report we show an application of neural architectures trained on the Penn Treebank dataset (Marcus et al. (1993)) to perform POS tagging. In particular, we show that recurrent models can be employed to achieve a performance of 0.82 in terms of average F1-score on this dataset; moreover, we show that the dataset imbalance hampers performance evaluation.

1. Introduction

In this assignment we performed POS tagging using neural architectures. POS tagging is a disambiguation task that assigns a Part-Of-Speech marker to each word in the input sequence based on both its definition and its context.

Given a corpus of documents, the task of this assignment is to predict the POS tag for each word. The corpus is split in training, validation and test sets; each document of the corpus is split over the sentences in order to have a smaller and more uniform maximal sequence length so that the architectures used could deal with the input size without suffering too much from vanishing gradients.

GloVe embeddings are the only input features of the models, enriched with contextual embeddings for out of vocabulary words. These OOV embeddings are computed independently for each split: starting from the Glove vocabulary, we compute embeddings for OOV terms of the training set based on words appearing in their respective contexts and we add these terms to the vocabulary, and then repeat for the validation and test set, enlarging the vocabulary at each step.

2. Architectures

In every architecture we tested, the input has shape $(batch\_size, max\_token\_len, features)$, while the output has shape $(batch\_size, max\_token\_len, classes)$. The input is first embedded with an Embedding layer, and from the embedding we obtain class scores using one of the following approaches:

- LSTM_1L: a Bidirectional LSTM layer followed by a linear layer
- GRU: a GRU layer followed by a linear layer
- LSTM_2L: two Bidirectional LSTM layers followed by a linear layer
- FC_2L: a Bidirectional LSTM layer followed by two linear layers with ReLU activation in between

We also decided to not include a (log) softmax layer to compute (log) probabilities as the network output, and rather used a dedicated criterion which computes a cross entropy loss from raw, unnormalized class scores in a more numerically stable way.

3. Experiments

Given that the dataset suffers from great class imbalance, we tried to train the models using weights for the cross entropy loss in order to give more importance to underrepresented tags. We also performed hyper-parameters tuning in order to select the best hyper-parameters for each model. This process involved many runs with fine tuned parameters selection. In 1, many hyper-parameters are fixed due to this manual refinement process and only the main parameters that affect the performances the most. Some fixed hyper-parameters we tuned are: the optimizer choice, between Adam and Rmsprop, and we fixed it to Rmsprop because models had better results and converged faster; the batch size, fixed to 32; the size of the hidden state of the models, fixed to 64, tuned also 32 and 128 sizes but didn't yield better results. Then, out of the starting four architectures, two were chosen according to the best f1 score achieved on the validation set, retrained on a bigger train + validation set and finally tested on the test set. To see all the runs and some plots, click here.

Model	Learning Rate	Weighted Loss	Validation F1 Score
LSTM_2L	0.001	True	0.802
GRU	0.00075	True	0.799
GRU	0.001	False	0.798
GRU	0.00025	True	0.796
LSTM_1L	0.00075	True	0.795
LSTM_1L	0.0001	True	0.795
GRU	0.00025	False	0.793
LSTM_2L	0.00075	False	0.787

Table 1: We report only the best runs of the hyper-parameters tuning due to space limitation.

At the end of the hyper-parameters tuning phase, the four models were run with the best hyper-parameters found (all of them are trained with the weighted loss):

- **LSTM_1L** with learning rate 0.00075 performed 0.80 of F1 score in validation;
- **GRU** with learning rate 0.00075 performed 0.77 of F1 score in validation;
- **LSTM_2L** with learning rate 0.001 performed 0.80 of F1 score in validation;
- **FC_2L** with learning rate 0.00025 performed 0.75 of F1 score in validation.

The best two models, **LSTM_1L** and **LSTM_2L**, are then retrained on the training + validation set and evalueted on the test set. The **LSTM_1L** model reaches 0.82 of F1 score on the test set, while the **LSTM_2L** drops its F1 score to 0.77 on the test set.

4. Results and Error Analysis

Given the plots of the F1 score per class on the validation set of the best two models, it is clear that all the models aren't able to classify correctly token with the tags **FW** and **UH** due to the fact that there aren't many examples of these two classes in the dataset (**UH** only 3 examples in train + validation set and **FW** only 4). Test performances on these two classes cannot be evaluated as there are no examples on the test set. Other tags that have poor performances in the validation are **LS**, **WP$**, **PDT**, **NNPS**, even though they're variable among different models. **LS**, **WP$** and **PDT** are three underrepresented tags, the first is not even present in the test set, the second has only 4 examples in the test set out of 14, the third has only 4 examples in the test set out of 23 total examples. One motivation behind the poor performances on the **NNPS**(plural proper nouns) tag is that it doesn't have as many examples as its singular counterpart **NNP**(singular proper nouns). Moreover, plenty of the **NNPS** tagged words are probably OOVs and contextual embeddings may not be the best embedding for such words. One possibly better way to address this issue could be to take into consideration the embedding of their singular counterpart in **NNP**, if present in the vocabulary, in computing its contextual embedding. Results on the test set partially confirm performances and findings discussed above: the **NNPS** tag has a good improvement on the test performances, but still has a F1 score below 0.5; the baseline model (the best, **LSTM_1L**) is able to improve performances from validation to test on several classes like **PDT**, **NNPS** and **RBS**. The second best model (**LSTM_2L**) seems more flawed: it is able to improve some of the poor performances on tags **WP$**, **NNPS**, **RBS** at the expense of the simple **TO** tag that had a F1 score of 0.99 in validation which drastically drops to 0.03 in the test. To see the plots click here.

Bibliography

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

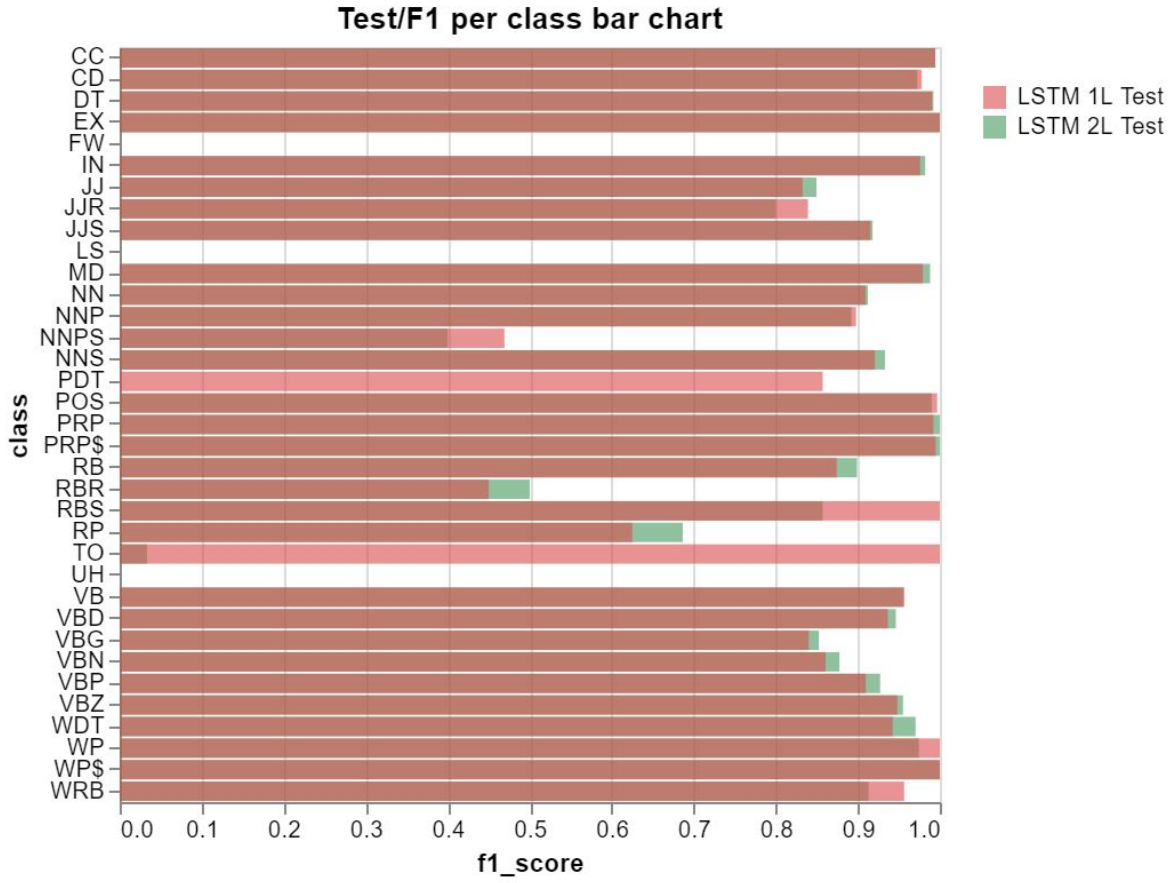

Figure 1: Per-class F1 scores of the best two models on the testing set