NLP Assigment 1: Part-of Speech (POS) tagging as Sequence Labelling using Recurrent Neural Architectures

Domenico Dell'Olio, Giovanni Pio Delvecchio and **Raffaele Disabato**
Master's Degree in Artificial Intelligence, University of Bologna
{ domenico.dellolio, giovanni.delvecchio2, raffaele.disabato}@studio.unibo.it

Abstract

In this report, an approach to POS Tagging involving GloVe embeddings, a small Recurrent Neural Network and some variations on its architecture is presented. Ultimately, this approach achieve good results, with best architectures reaching 78% macro F1-scores on validation and test sets, but still showing some flaws probably due to the small and unbalanced dataset and low model complexity.

1 Introduction

Part-Of-Speech (POS) tagging is a fundamental NLP task, consisting of assigning a syntactic label to natural language tokens. Among the main approaches to this problem there are Rule-based methods which are explainable, but often require an impractical high number of rules. Then there are Stochastic methods, mainly relying on Hidden Markov Models (HMM), n-gram models or Conditional Random Fields (CRF). They achieve very good results, but can be heavily biased by the corpus (Kanakaraddi and Nandyal, 2018). The same characteristics are shared with Deep Learning methods, mainly exploiting Recurrent architectures and pre-trained word embeddings. The presented approach belongs to this class, proposing to apply a simple Bi-LSTM + Dense network with GloVe embeddings (Pennington et al., 2014) to solve the task on a portion of the Penn Treebank corpus (Marcus et al., 1993). Further experiments are made to test some variations (Figure 1) on this architecture. All the models are then compared on a validation split and then only the best two are compared on a test split. Both evaluations are performed considering the F1-macro metric, discarding punctuation and symbol tokens. The results show that the baseline and the 2LSTM are the best models (77.6% and 78% macro F1 respectively on test set) while the other two variations (2Dense and GRU) have some limitations due to their architectures.

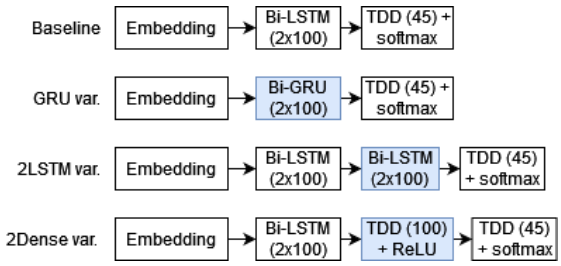

Figure 1: Tested model architectures. Variations from the baseline are highlighted in blue. TDD stands for "Time Distributed Dense".

2 System description

The dataset is composed of already tokenized and labeled text paragraphs, with each end of period marked by an empty line. Every token is tagged with one of 45 classes, with an uneven distribution across splits. The pre-processing pipeline on the dataset consists of lower-casing, sentence splitting (on new lines or ";" tokens), sentence and label padding. Every example is brought to the length of the longest sentence in the training set. The padding is masked by the embedding layer of the network and it does not contribute to the loss. About word representation, GloVe 100d embeddings are employed. Out-Of-Vocabulary words are dealt with uniformely random embeddings. Another kind of "contextual" embedding was tested but discarded, as it produced slightly lower scores. The architectures tested on this dataset are presented in Figure 1.

3 Experimental setup and results

The dataset is split in the following way: documents 1-100 are the training set, 101-150 are used as validation and 151-199 are used as test set. All the models are trained for a maximum of 100 epochs with the Adam optimizer and an Early Stopping callback monitoring the validation loss, with a patience of 10 epochs and a minimum delta of

10^{-4}. Hyperparameters are tuned by hand observing the training loss curves and the performances after each run. In particular:

- Embedding dimension: 100 chosen from {50, 100, 200, 300}. This proposes the best trade-off between good scores and model complexity.
- Batch size: 64 chosen from {32, 64, 128}.
- Learning rate: 1e-3 for the GRU variation, 1e-2 for the rest, chosen from {1e-2, 5e-3, 1e-3}
- The units for each layer are chosen among {50, 100, 150, 200}, which empirically give good results.

Different learning rate schedulers were tested, as the task resembled one of fine-tuning, but they did not bring remarkable improvements. Table 1 contains the results of the models on validation and test sets. The aggregation does not include F1-score of symbol and punctuation classes and considers the score of missing classes as 0, establishing a lower limit to the actual models performances.

Model	Macro val. precision	Macro val. recall	Val. macro F1
Baseline	**0.814**	**0.791**	**0.786**
GRU var.	0.785	0.782	0.777
2LSTM var.	0.793	0.780	0.779
2Dense var.	0.755	0.752	0.746
	Macro test precision	Macro test recall	Test macro F1
Baseline	0.781	0.779	0.776
2LSTM var.	**0.790**	**0.782**	**0.780**

Table 1: results of the runs of the models using seed 21 for reproducibility.

4 Discussion

The results show that the baseline is a very good starting model. In fact, the only variation that slightly outperforms it is the 2LSTM, having more capacity. The other networks are marginally worse as one employs a GRU layer that is simpler than the LSTM used in the baseline and the 2Dense reinforces the part of the network that doesn't focus on contextual information. We also observe that the best models (baseline and 2LSTM variations) scores on the test set are similar to the ones on validation and also the distribution of F1-scores across classes is coherent. From these, we can see that the set of difficult classes includes:

- Some classes with almost negligible support as Foreign Words (FW) and Interjections (UH). The network can't learn their role also because they are represented by very different tokens. In fact, there are also tiny classes as Existential "There" (EX), which are correctly classified thanks to the context and the restricted pool of words representing them.
- Inherently ambiguous classes. An example is the Proper Noun in Plural Form (NNPS), which is confused with Singular Proper Nouns (NNP) or Plural Nouns (NNS). Even if the 2LSTM variation improves a bit on these kind of classes, they are still associated to a good part of wrongly predicted examples.
- Predeterminers (PTD). They usually preceed Determiners (DT) and should be easily detected, but they often are predicted as DT or Adjectives (JJ), probably due to PTD being a smaller class.

These problems could be addressed by considering more data to solve class unbalance, by using a bigger model, by adding some rule-based correction after the prediction is done to disambiguate some classes as PDT, or using a sub-word encoding (as the Byte-Pair) to better predict those tokens which role is strictly linked to their wordform, as FW.

5 Conclusion

In this work we have presented a possible way to approach the POS tagging problem with a RNN architecture and GloVe word embeddings, trying different combinations of layers and hyper-parameters. In the end, we achieved good results with the baseline model and managed to improve it while keeping the architecture relatively simple (2LSTM has $\approx$1.4M parameters, $\approx$400k trainable). In particular, only the 2LSTM variation outperformed the baseline, with very close F1-scores both on validation and test set. This easily follows from the fact that this model has more capacity, while GRU and 2Dense variations were too simple w.r.t. the baseline. However, all the models still suffer from class unbalance and inherent ambiguous terms, which could be addressed by using more data, more complex models, different kind of word representations and/or exploiting some rule-based corrections after the prediction process.

6 Links to external resources

- Weights of the networks that produced the results in Table 1 are available here.

References

Suvarna G Kanakaraddi and Suvarna S Nandyal. 2018. Survey on parts of speech tagger techniques. In *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*, pages 1–6.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.