Assignment 1

Matteo Donati, Ildebrando Simeoni and **Diego Biagini**
Master's Degree in Artificial Intelligence, University of Bologna
{matteo.donati10, ildebrando.simeoni, diego.biagini2}@studio.unibo.it

Abstract

In this paper, a well defined and structured pipeline to asses the Part-Of-Speech (POS) Tagging problem is proposed. After a careful analysis of the Treebank dataset, a variety of simple but yet powerful RNN-based models, alongside GloVe embeddings, have been used to tackle the problem at hand. After having tuned the hyperparameters of the proposed architectures using a grid-search approach, the two best models have been selected and evaluated on the test split of the original dataset, obtaining overall good results despite the simplicity of the architectures at hand.

1 Introduction

The POS tagging problem has been tackled in a variety of ways during the decades, and some of the most relevant approaches are: i). rule-based POS tagging, namely one of the oldest method, which is based on the use of hand-written rules to choose a tag for each token; ii). probabilistic POS tagging, which is a more mathematically sound technique based on Hidden Markov Models (HMMs) that tries to solve the optimization problem defined by $\hat{t}_1^n = \text{argmax}_{t_1^n} P(w_1^n|t_1^n)P(t_1^n)$; iii). POS tagging based on neural models. These models, especially RNN-based models, are able to reach outstanding results.

Being the state of the art for the task at hand (Wang et al., 2015; Akbik et al., 2018), RNN-based models have been chosen as the main and only option for this work. In particular, four different architectures have been tuned using the training and validation splits of the Penn Treebank dataset, so to find the best possible configuration for such architectures. Out of the four resulting models, only the two best models were evaluated on the test set, obtaining macro F1-scores of 0.78 and 0.76. These scores refer to the performances of the two models without considering neither punctuation nor symbols tags.

Based on our experimental results, bidirectional LSTMs (BiLSTMs) followed by a single fully-connected layer tend to be the type of architecture that performs best in solving the task at hand.

2 System description

The pipeline we implemented to solve the aforementioned tagging problem is the following.

We started by analysing the Penn Treebank dataset, and by structuring the data so to have a table having two columns: one for sentences and one for lists of POS tags. Then, using GloVe embeddings, we added an additional column to the aforementioned table, so to store lists of embeddings. More specifically, we decided to use $d = 50$ as embedding size, leading to embeddings being vectors of $\mathbb{R}^{50}$. In this context, the embeddings for Out-Of-Vocabulary (OOV) terms have been set equal to $\mathbf{0} \in \mathbb{R}^{50}$. In order to be able to train neural models, we also made sure to pad the lists of POS tags and of embeddings.

After a careful analysis and preprocessing of the data, we implemented the four architectures to be trained on such data. The architectures taken into consideration are shown in table 1. The hyperparameters of these architectures have been tuned so to find the best number of units and neurons among predefined sets of values.

After training, the two best models were selected according to the best validation macro F1-scores so to be evaluated on the test split of the original dataset. Lastly, considering these two models, some error analysis was performed. First, the macro F1, macro precision, macro recall and error rate scores were computed for both models, on both the validation set and the test set, so to compare performances across dataset splits. Then, considering the two models, the confusion matrices and some examples of the most miss-classified sentences were computed and analysed.

Model	Description	Seq-to-seq layers	Head layers
m_0	Baseline model	BiLSTM$_{m_0}$	TD-Dense$_o$
m_1	BiGRU model	BiGRU$_{m_1}$	TD-Dense$_o$
m_2	Additional BiLSTM layer model	BiLSTM$_{m_0}$, BiLSTM$_{m_2}$	TD-Dense$_o$
m_3	Additional dense layer model	BiLSTM$_{m_0}$	TD-Dense$_{m_3}$, TD-Dense$_o$

Table 1: Architectures. All head layers are time-distributed (TD) ones.

Model	F1 val.	F1 test	Pr. val.	Pr. test	Re. val.	Re. test	e val.	e test
m_0	0.779	0.783	0.795	0.786	0.778	0.787	10.47%	9.58%
m_2	0.774	0.756	0.780	0.760	0.778	0.758	10.60%	10.11%

Table 2: Macro scores and error rates.

3 Experimental setup and results

The hyperparameters of the four architectures presented in table 1 were tuned using a grid-search approach. In particular, such approach has been used to select the best number of units and neurons of layers BiLSTM$_{m_0}$, BiGRU$_{m_1}$, BiLSTM$_{m_2}$ and TD-Dense$_{m_3}$ respectively. This was made possible by first defining a set C of possible candidates for each of the mentioned layers to be tuned. In particular, $C_{\text{BiLSTM}_{m_0}} = C_{\text{BiGRU}_{m_1}} = C_{\text{BiLSTM}_{m_2}} = C_{\text{TD-Dense}_{m_3}} = \{32, 64, 128, 256\}$. Then, the best configuration for each architecture was obtained by training and validating on the respective dataset splits. From the computed results, the best number of units/neurons for BiLSTM$_{m_0}$, BiGRU$_{m_1}$, BiLSTM$_{m_2}$ and TD-Dense$_{m_3}$ were 256 units, 256 units, 256 units and 128 neurons respectively.

All the seq-to-seq layers, namely BiLSTM$_{m_0}$, BiGRU$_{m_1}$, BiLSTM$_{m_2}$, use a recurrent regularizer factor equal to 1e$-$2. Moreover, TD-Dense$_{m_3}$ uses a ReLU activation, while TD-Dense$_o$ use a softmax activation so to produce a probability distribution over the possible tags.

All the models were trained using categorical cross-entropy as loss function and Adam as optimizer. In particular, the initial learning rate has been set equal to 1e$-$2, but a specific learning rate schedule has been used so to divide such value by a constant factor of 10 in case the validation error stopped improving. Lastly, early-stopping was used so to prevent overfitting and restoring the best possible weights for the model at hand.

Table 2 presents and compares the results of the two best models on both the validation and test sets. In particular, all the scores were computed not considering neither punctuation nor symbols tags, and considering as 0 the scores associated with zero-support classes.

4 Discussion

Based on the scores presented in table 2, we noticed how the baseline model, m_0, is the best performing model. Indeed, all the other models, which were supposed to extend and improve the baseline model did not performed as well as m_0. Moreover, for both the best models, the error rate on the validation set, e val., is smaller than the error rate on the test set, e test, mainly due to the presence of more zero-support classes in the test set than in the validation set.

Considering some of the most miss-classified examples, we noticed three main problems: i). both m_0 and m_2 are not able to effectively distinguish between noun classes (i.e. `NN`, `NNS`, `NNP`, `NNPS`, `NNPS`); ii). proper nouns can be mistaken with non-nouns tags; iii). some of the samples of the original dataset contain wrong labels. All of these problems were also exhaustively covered by Manning (2011).

5 Conclusion

The here-presented models are simple models that are capable of reaching average results, but still unable to reach the current state of the art. Indeed, some of the previously discussed errors are thought to be due to the simplicity of the particular models at hand.

In order to improve the overall results, one possibility would be to increase m_0 capacity (e.g. by increasing the number of units of the BiLSTM layer), or to use more sophisticated architectures, such as transformers, alongside correcting the gold standard mistakes.

References

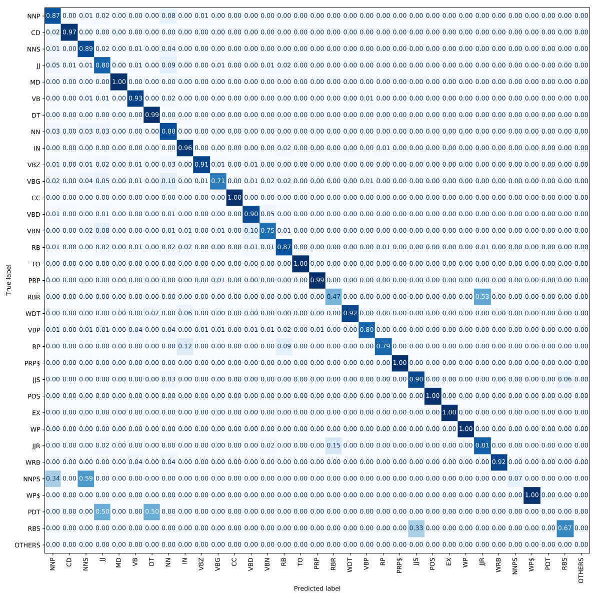
Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Christopher D. Manning. 2011. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In *Computational Linguistics and Intelligent Text Processing*, pages 171–189, Berlin, Heidelberg. Springer Berlin Heidelberg.

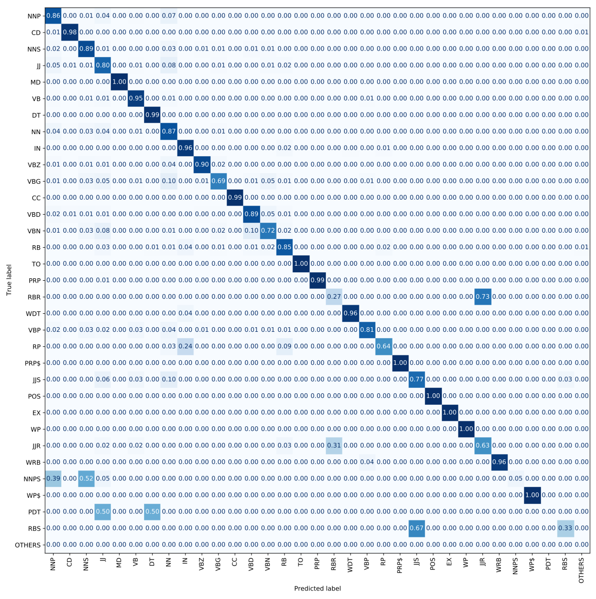
Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Zhao Hai. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *ArXiv*, abs/1510.06168.

A Appendix

A.1 Confusion matrices



(a) m_0



(b) m_2

Figure 1: Confusion matrices

Figure 1 shows the two confusion matrices computed for both m_0 and m_2 on the test set. In particular, such matrices use a special class, `OTHERS`, which represents all punctuation, symbols and zero-support classes. The sets of tags that are the most miss-classified tags are listed below:

- {`NN`, `NNS`, `NNP`, `NNPS`}, that is the set of noun tags.
- {`RBR`, `JJR`}, that is the set of comparative adverb and comparative adjective tags.
- {`RBS`, `JJS`}, that is the set of superlative adverb and superlative adjective tags.
- {`DT`, `PDT`}, that is the set of predeterminer and determiner tags.

One can notice how all the tags in each of these sets have similar part-of-speech roles.

A.2 Text examples

The here-presented text examples show some of the most miss-classified sentences of m_0 and m_2 without considering neither punctuation nor symbols tags.

- Considering the sentence `['Business', ':', 'Savings', 'and', 'loan']` with true tags `['NN', 'NNS', 'CC', 'NN']`, the model m_0 predicts `['NNP', 'NNS', 'CC', 'NNP']`, showing an error rate of 50%. This example highlights the confusion between noun tags.
- Considering the sentence `['GRAINS', 'AND', 'SOYBEANS', ':']` with true tags `['NNPS', 'NNP', 'NNPS']`, the model m_2 predicts `['NNS', 'CC', 'NNP']`, showing an error rate of 100%. This example highlights both the confusion between noun tags, as well as errors in the given dataset. Indeed, the conjunction `AND` should be tagged as `CC` (i.e. coordinating conjunction) instead of `NNP`.