Assignment 1

Alessandro D'Amico, Davide Femia, Sfarzo El Husseini and **Riccardo Paolini**
Master's Degree in Artificial Intelligence, University of Bologna
{ alessandro.damico5, davide.femia, sfarzo.elhusseini, riccardo.paolini5 }@studio.unibo.it

Abstract

In this assignment we address the task of POS tagging on the well known *Penn Treebank Dataset*. The aim is to implement and compare the performances of classic recurrent models which make use of *GloVe* embeddings to later choose the best two among them and analyze their errors. The obtained results are quite satisfactory, even considering the reduced size of the model (in terms of number of parameters).

1 Introduction

POS tagging (*part-of-speech tagging*) is a task that consists in labeling each word in a text corpus on the basis of its grammatical role. The literature on this task includes both traditional models (such as hidden Markov models, visible Markov models, SVMs and maximum entropy classifiers) and state-of-the-art models (such as SALE-BART and ACE). The first ones are mostly non-neural models, that would require a consistent effort to be integrated into novel architectures and usually don't give the best scores, while the second group achieves the best scores, making use of a high number of parameters. Recurrent neural networks such as *LSTMs* and *GRUs*, coupled with the use of pre-trained embeddings (such as *GloVe*) are in the middle with respect to the above defined groups. They're powerful enough to obtain considerable scores and the number of parameters is low.

We've chosen the *GloVe* embeddings of length 100: this is because they provided a representation richer than the ones having length 50, while instead the ones of length 200 and 300 provided a negligible improvement, which also would have required to restructure the model in use, increasing the overall number of parameters (and therefore, loosing its 'lightweight' characteristic).

To handle *OOVs* (*Out-Of-Vocabulary words*) we've used a technique that gives embeddings equal to their corresponding lowercase if they are present in *GloVe* - if no match is obtained, a random embedding is generated (we've done this step because some words such as *Mercedes* are present in the dataset but just the *mercedes* without capital letters is present in *GloVe*). We've also tried to give an embedding based on a mean embedding of similar words based on jaccard similarity between the trigrams of the lowercased OOV words and the trigrams of *GloVe* vocabulary (we used 0.6 as a treshold for the Jaccard similarity) - if a lowercased OOV word doesn't have similar words in *GloVe* a random embedding is generated, but this didn't give better results than random embeddings, probably because the embeddings in *GloVe* are semantic rather than syntactic.

2 System description

We've implemented, trained, validated and tested the 4 following models:

1. BiLSTM (baseline): this model has just a bi-directional layer of LSTM cells, in addition to the dense input layer (followed by the fixed embedding layer) and the dense time-distributed output layer (Figure 1).

2. Double BiLSTM: Similar to the baseline, but with an additional bidirectional LSTM layer (Figure 2).

3. BiLSTM + Dense: Similar to the baseline, but with an additional Dense distributed layer before the output layer (Figure 3).

4. BiGRU: this model is similar to the baseline, but with a bi-directional GRU layer instead of a bi-directional LSTM layer (Figure 4).

3 Experimental setup and results

All the train-validation-test process is reproducible, since we've used a fixed seed. Documents 1-100 are the train set, 101-150 validation set, 151-199

test set and every document is split into sentences. To set the fixed length of the text sequences we've cut sentences down to the 99.5 percentile. To perform training we've used *Adam* with *learning rates* around 10^{-2}, a $batch_{size} = 64$, *categorical crossentropy* as loss, *categorical accuracy* as metric and an early-stopping routine based on the validation set *categorical accuracy*. During validation and testing, the punctuation, the symbols and the span tags used by the model were removed. The obtained results are shown in Tables 1 and 2.

4 Discussion

The results obtained from the four models are similar, therefore we've chosen the best models as the ones minimizing the number of parameters (Table 3): The *BiLSTM (Baseline) and the BiGRU*. The other models didn't provide significant advantages.

The model performs quite well both in terms of *accuracy* and $macro - F_1$ score. The results obtained on the validation set are lower than the ones on test set: this is because some minority classes [1] are never predicted as they are, this results in a penalization of the overall $macro - F_1$. The imbalanced nature of the training set makes the model perform better on the classes having higher support, while the minority classes are frequently mismatched: this is because the limited number of minority instances in the train set makes it difficult to learn their features and, at the same time, the model gets partially biased towards similar (but more numerous) classes. Another source of errors is the ambiguity of the role of some tags in the sentences: if the tags would have been multiple and guessing one class would have been enough, this problem would be surmountable, but instead the annotators decided to give a single class to each word: this is a simplification that forces the use of some conventions (some kind of "right-hand rules") and sometimes leads to the increase of the ambiguity inside the model - which possibly could have learned them using more samples containing those conflicting TAGs[2].

[1]The classes *FW* and *UH* are present in the training set and in the validation set (even if there are just few samples) and are absent in the test set

[2]The use of these conventions is discussed in the error analysis and refers to the dataset official documentation (Santorini, 1990)

5 Conclusion

The overall *accuracy* and $macro - F_1$ As expectable, the models performed well on the most numerous classes and bad on the least numerous. This is a characteristic shared by most of the neural networks and to which the use of *GloVe* pretrained embeddings cannot compensate. In conclusion, we can highlight some possible ways to improve the models:

- using a trained model (even another LSTM) to generate better contextual embeddings for OOV words
- using a bigger model for the whole process (this may have a higher cost both at train and at inference time)
- using bigger embeddings, 200-300 (in this case also the contextual embedding of the LSTM/GRU and other parameters numbera have to be increased)
- using regularization techniques (such as Dropout, Recurrent Dropout, L2), which we didn't apply to avoid misinterpretations of the given assignment
- using libraries to augment the data of the smaller class tags
- allowing the training of the *GloVe* embeddings (which in this assignment is prohibited), which could be refined on the basis of the actual context in use. In this case the way we've built the dictionary could influence the whole process and embeddings "not requested" should be anyway blocked - this is not an easy modification and it is not guarantee to improve the objective performance.
- lemmatizing some words'[3] - e.g. *v.* and *vs.* in *versus*

6 Links to external resources

The repository of this work:
https://github.com/DavideFemia/POS-tagging-with-RNNs

[3]We've avoided preprocessing text because in comparison to Twitter datasets, which contain lots of emoji, symbols and slangs, this corpora was quite "clean" itself.

References

Beatrice Santorini. 1990. Upenn treebank tag guide. https://www.cis.upenn.edu/~bies/manuals/tagguide.pdf. Accessed: 2022-01-11.

	Accuracy	Macro-F1
BiLSTM (baseline)	0.912	0.799
Double BiLSTM	0.917	0.801
BiLSTM +Dense	0.913	0.763
BiGRU	0.914	0.798

Table 1: Resulting scores on the validation set

	Accuracy	Macro-F1
BiLSTM (baseline)	0.917	0.850
Double BiLSTM	0.923	0.856
BiLSTM +Dense	0.922	0.850
BiGRU	0.922	0.864

Table 2: Resulting scores on the test set

	trainable params	total params
BiLSTM (baseline)	90.4K	1.3M
Double BiLSTM	189.2K	1.4M
BiLSTM +Dense	106.9K	1.3M
BiGRU	69.7k	1.2M

Table 3: Number of parameters in the implemented models

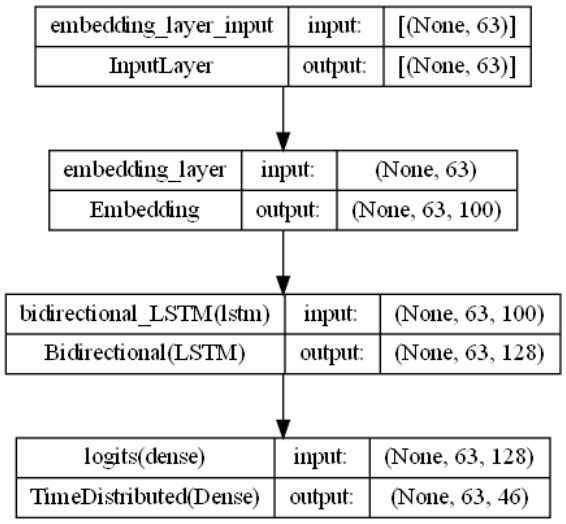

Figure 1: BiLSTM (baseline)

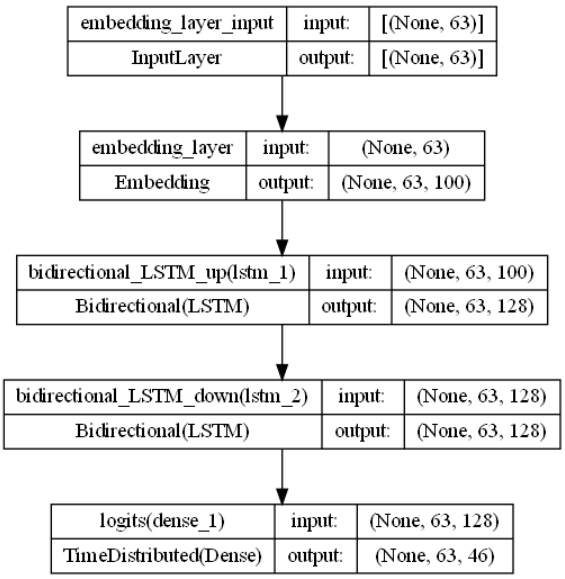

Figure 2: Double BiLSTM

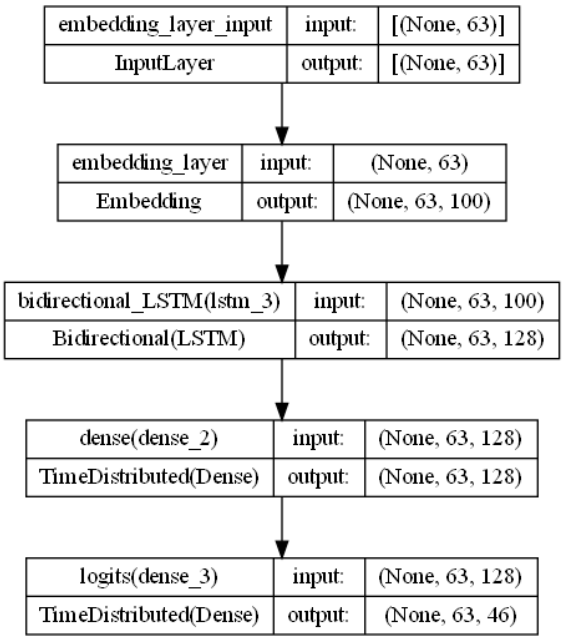

Figure 3: LSTM+dense

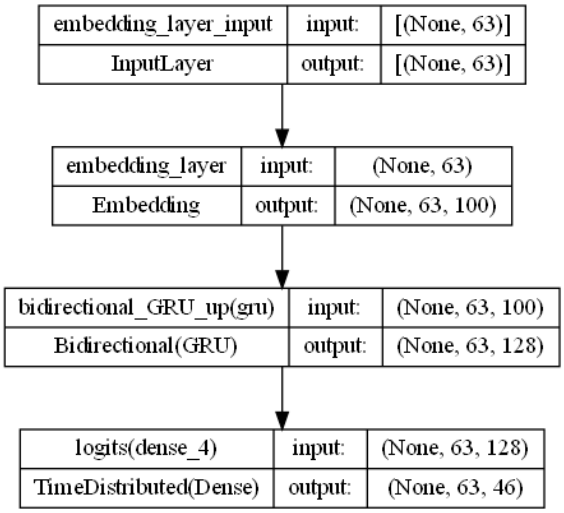

Figure 4: Double BiGRU