Assignment 1

Antonio Politano, Enrico Pittini, Riccardo Spolaor and **Samuele Bortolato**
Master's Degree in Artificial Intelligence, University of Bologna
{ antonio.politano2, enrico.pittini, riccardo.spolaor, samuele.bortolato }@studio.unibo.it

Abstract

Our approach consists in addressing the POS tagging task by means of Recurrent models, using the words *GloVe* embeddings. In particular, the OOV words are handled by training their embeddings using the same *GloVe* procedure. The achieved performances are overall quite good: indeed, the obtained scores are high and most of the errors are understandable and minor, among similar tags.

1 Introduction

Part-Of-Speech tagging is the process of marking up words of a text as corresponding to particular parts of speech. The POS tagging task can be addressed both with non-neural approaches (e.g. *Hidden Markov Models*, *Naive Bayes*, *Conditional Random Field*) and with neural approaches (e.g. *Recurrent models*, *Transformer models*).

In this work, we explore different Recurrent models, using the 100 dimensions words *GloVe* embeddings. Our approach for handling the OOV words consists in performing the same training procedure described in the *GloVe* paper (Pennington et al., 2014).

The dataset taken into account is the dependency treebank, divided by sentences, and split into $1,959$ training samples, $1,277$ validation samples and 638 test samples. By analysing the dataset it can be observed that the tag distribution is equivalent between training and validation sets. Among the most frequent tags names, prepositions and adjectives can be found.

As preprocessing, a simple lowercase transformation and a substitution of the numeric tokens with the special token `[num]` are applied (the reason of that second transformation is that most of the numeric tokens have the same tag). Different models with different hyperparameters have been tested. These Recurrent models perform quite well, with good scores, even with not many epochs of training. Most of the errors are minor and understandable, among similar tags.

2 System description

The handling of the OOV words consists in expanding the standard *GloVe* embedding model with new embedding vectors. This is achieved by training the OOV words embeddings using the same procedure described in the *GloVe* paper.

In particular, the same procedure is applied three times. At each call, the embedding vectors of the already known words are kept frozen, while only the embeddings of the new OOV tokens are trained.

1. The standard *GloVe* embedding model is expanded with the training set OOV words embeddings. For doing so, the co-occurence matrix between tokens is built.

2. The current embedding model is further expanded with the validation set OOV words embeddings, building the co-occurence matrix considering both the training and validation sets.

3. The current embedding model is expanded one last time with the test set OOV words embeddings, building the co-occurence matrix considering the training, validation and test sets.

Regarding the models, the following four models have been considered.

- Baseline model: Bidirectional LSTM plus final Dense layer.
- First model: Bidirectional GRU plus final Dense layer.
- Second model: Two Bidirectional LSTM plus final Dense layer.
- Third model: Bidirectional LSTM plus two final Dense layers.

Model name	Val accuracy	Val f1-score
Baseline model	0.92	0.77
First model	0.92	0.82
Second model	0.93	0.79
Third model	0.92	0.78

Table 1: Models results

3 Experimental setup and results

Different hyperparameters have been tuned, in particular: the dropout rate p, inside the Bidirectional layer (best value is 0.2); the dimension of the hidden states returned by the Bidirectional layer (best value is 128); the merge mode for concatenating the hidden vectors in the Bidirectional layer (best mode is channels concatenation).

For selecting the best combination of hyperparameters, the f1-score macro on the validation set has been used as metric of reference. It is important to point out that in performing such evaluation the punctuation classes have not been considered.

Regarding the training procedure, the Adam optimizer has been chosen, with learning rate 0.001. The cross-entropy loss has been used. Finally, early stopping and reduce LR on plateau have been used.

Table 1 shows the results of the four models with the best combination of hyperparameters. As it can be noticed, all the models achieve quite good results, especially considering that the most basic model which always predicts the most frequent tag has a validation f1-score macro of 0.07.

4 Discussion

The two best models on the validation set, i.e. the first and second models, are evaluated also on the test set. The f1-score macro of the both models on the test set is 0.86. Since the first model is slightly better on the validation set, it has been selected as best model. Now, an error analysis is carried out on this best model over the test set.

By looking at the plots and examples shown in the appendix section, the following considerations can be drawn.

First of all, the classes with highest support are predicted quite well: indeed, all of them have a f1-score above 0.8.

Furthermore, there are only four classes with a f1-score below 0.8, namely the classes `PDT`, `NNPS`, `RBR` and `RP`, which are classes with very low support.

Most of the model errors are about similar tags. For example, the model struggles in discriminating between the singular and plural forms of the same tag, between the proper and common nouns, or between the comparative adjectives and comparative adverbs.

However, while the just mentioned errors are understandable, there are also some kinds of errors which are unexpected. For instance, the model often misclassifies `-RRB-` tokens, assigning `NN`.

Regarding the worst classified tokens, most of the errors are quite understandable. In particular, the following considerations can be made.

First of all, the token `yen` is the most frequently misclassified one. The model has difficulties in discriminating between the singular and plural form of this token: if it follows a number, the model assigns the plural form, while the true one is the singular.

Several frequently misclassified tokens are tokens which can have different POS tags depending on their usage in the sentence. For instance, the token `that` has an almost even distribution of tags between `WDT`, `DT` and `IN`.

Finally, several frequently misclassified tokens are uncommon proper nouns (e.g. `waertsilae`). Since the model has never seen that token, or very rarely, it struggles to properly classify it.

A very similar analysis has been carried out also over the validation set. No significant differences have been noticed.

5 Conclusion

Overall, the Recurrent models perform quite good. Indeed, the scores are quite high and most of the errors are understandable and minor, among similar tags.

Nonetheless, there are some kinds of errors which are quite peculiar, e.g. the errors on the `-RRB-` tag. For fixing this kind of behaviour and improving the models on these particular cases, a further analysis and preprocessing of the data could be carried out.

6 Links to external resources

Link to GitHub repository.

References

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Appendix

In this appendix, some plots regarding the error analysis on the best model over the test set are shown.

Figure 1 shows the analysis of the worst tags. As it can be seen, there are only few tags with f1-score below 0.8, which are tags with very low support.

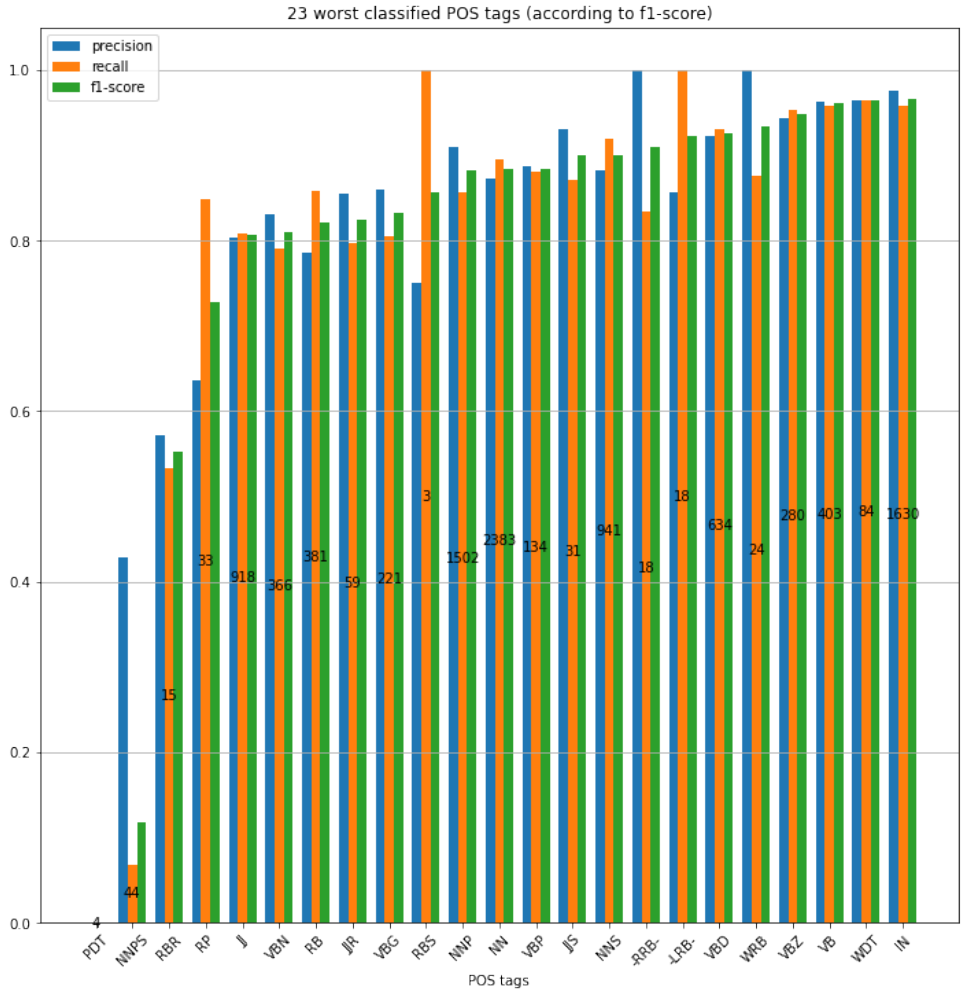

Figure 1: Analysis of the worst classified tags, according to the f1-score macro. The number inside each bar represents the support of that class.

Figure 2 shows the confusion matrix on the worst classified tags. As it can be seen, most of the errors are between similar tags. For instance, most of the misclassified `NNPS` tokens are predicted either as `NNP` or `NNS`. Or, for making another example, the model often confuses `RBR` with `JJR`. However, there are also kinds of errors which are quite unexpected, for instance the errors on the `-RRB-` class: several of these tokens are predicted as `NN`.

The following figures 3, 4 and 5 show some high relative error rate sentences containing some of the errors just mentioned.

Figure 6 shows the worst classified tokens analysis. The tokens described in section 4 can be noticed: the `yen` token; the tokens with an almost even tags distribution (e.g. `up`, `that`, `plans`); the uncommon proper nouns (e.g. `waertsilae`,or `soviet`).

The following figures 7, 8 and 9 show some high relative error rate sentences containing some of the errors just mentioned.

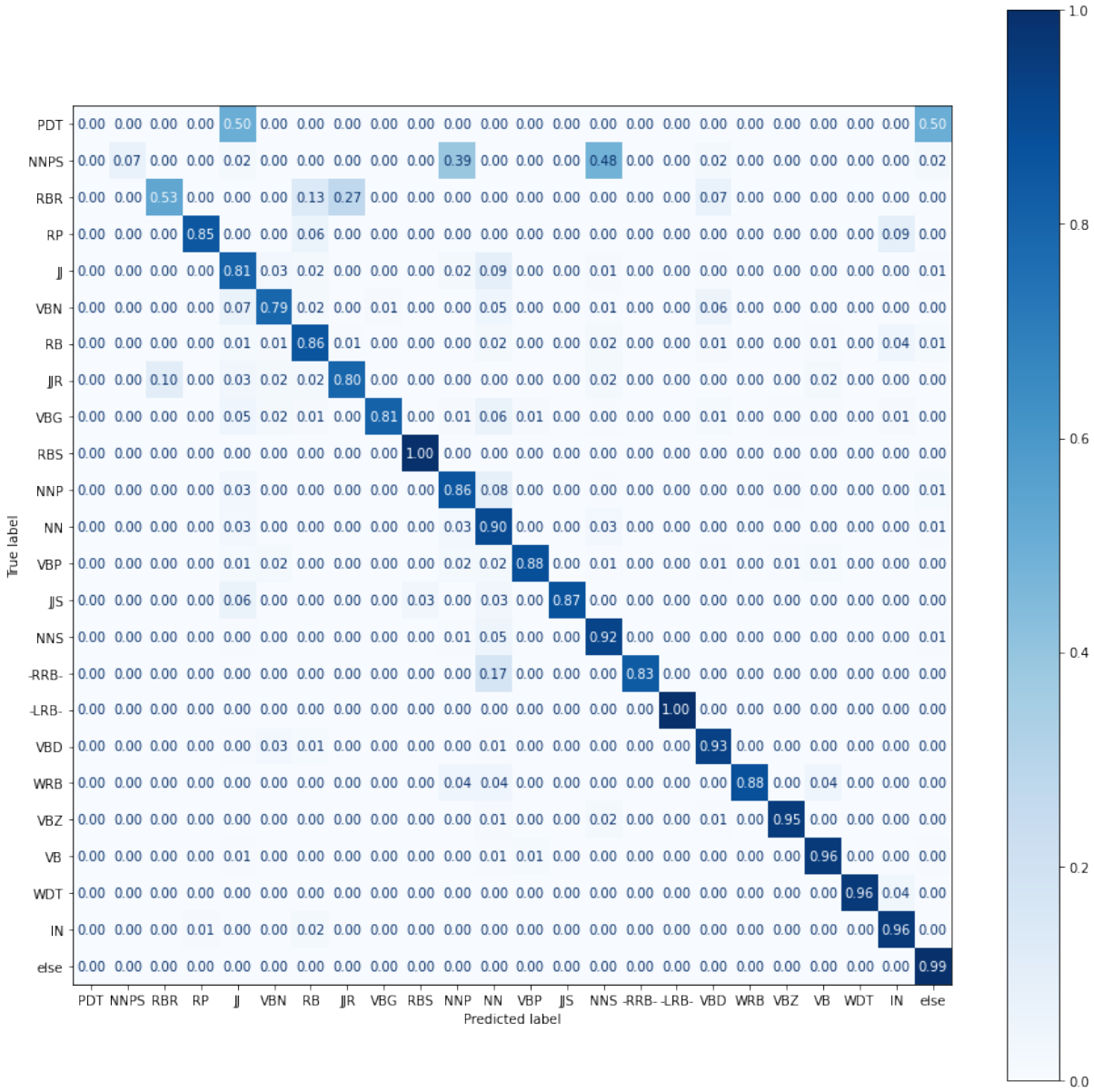

Figure 2: Confusion matrix of the worst classified tags

```
LEGEND
        word: correctly classified word
        word[TRUE_TAG/WRONG_TAG]: misclassified word
        word[TRUE_TAG/WRONG_TAG]: misclassified tag of interest
        word: non-evaluated word (i.e. punctuation)

1) Sentence index 149
Relative error: 0.28
`` this market 's[VBZ/POS] still going[VBG/VBP] through its pains[NNS/NN] , '' said philip puccio , head of equity trading at prudential-bache[NNP/JJ]
securities[NNPS/NNS] .

2) Sentence index 48
Relative error: 0.28
one[CD/DT] analyst , arthur stevenson , of prudential-bache[NNP/JJ] securities[NNPS/NNP] , new york , estimated that[IN/DT] [num] % or more of brazil
's newly[RB/VBN] made[VBN/VBD] automobiles run[VBP/VBN] on alcohol and ca n't use gasoline[NN/NNS] .

3) Sentence index 569
Relative error: 0.27
precious[NNP/JJ] metals[NNPS/NNS] : futures[NNP/NNS] prices eased as[RB/IN] increased stability and strength came into the securities markets .
```

Figure 3: Examples of some high relative error rate sentences containing errors on the NNPS tag

```
1) Sentence index 357
Relative error: 0.22
columbia stock recently[RB/VBD] hit[VBD/VBN] [num] [num] , after reaching [num] [num] earlier[RBR/JJR] this year on rumors[NNS/VBG] that mr. spiegel
would take the thrift private[JJ/NN] .

2) Sentence index 605
Relative error: 0.15
still , usx fared[VBD/RB] better[RBR/RB] than other major steelmakers , earning more per[IN/JJ] ton of steel shipped than either[DT/IN] bethlehem steel
corp. , which posted a [num] % drop in net income , or inland steel industries[NNPS/NNP] inc. , whose profit[NN/NNS] plummeted [num] % .

3) Sentence index 608
Relative error: 0.11
charles bradford , an analyst with merrill lynch capital markets[NNPS/NNS] , said usx may have received orders lost[VBN/NN] by competitors who were
involved in labor contracts earlier[RBR/JJR] this year .
```

Figure 4: Examples of some high relative error rate sentences containing errors on the RBR tag

```
1) Sentence index 474
Relative error: 0.24
absorbed in doling out `` feeding[NNP/VBG] frenzy[NNP/NN] 's '' tidbits , the authors gloss over the root causes[NNS/NN] of wedtech[NNP/NN] ,
namely[RB/VBG] the section [num] -lrb- a[NN/DT] -rrb-[-RRB-/NN] federal program under whose auspices[NNS/VBN] the scandal took place .

2) Sentence index 476
Relative error: 0.21
'' programs like section [num] -lrb- a[NN/DT] -rrb-[-RRB-/NN] are a little[RB/JJ] like leaving gold in the street and then expressing[VBG/VBD]
surprise when thieves walk by[RP/IN] to scoop it up[IN/RP] .
```

Figure 5: Examples of some high relative error rate sentences containing errors on the −RRB− tag

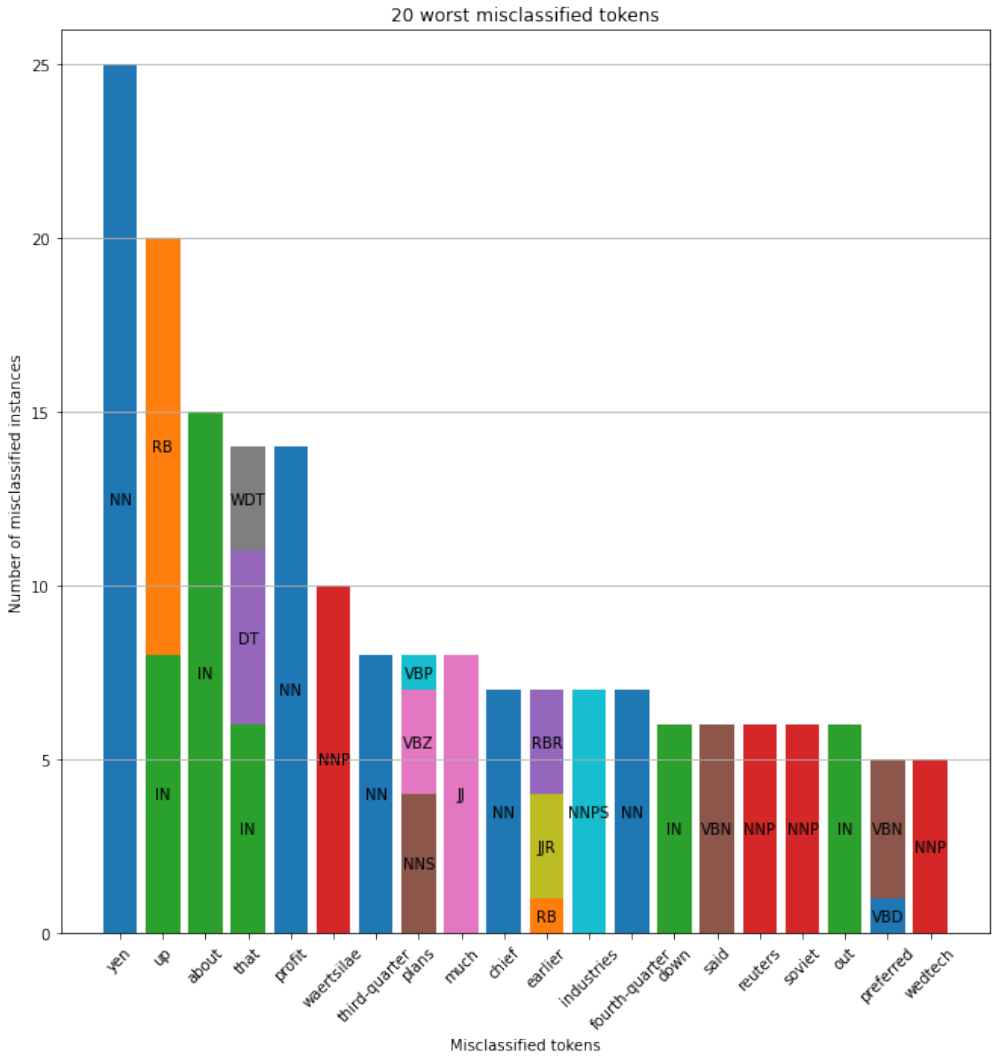

Figure 6: Analysis of the worst classified tokens. The colored splitting inside each bar represents the number of different true POS tags for each token.

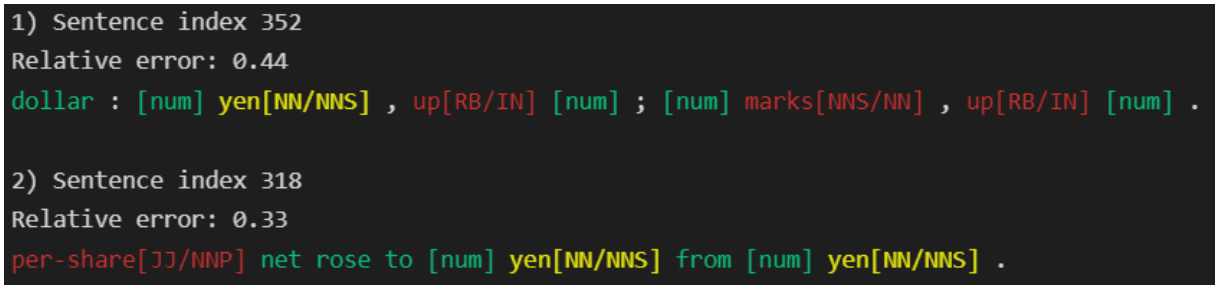

Figure 7: Examples of some high relative error rate sentences containing errors on the `yen` token

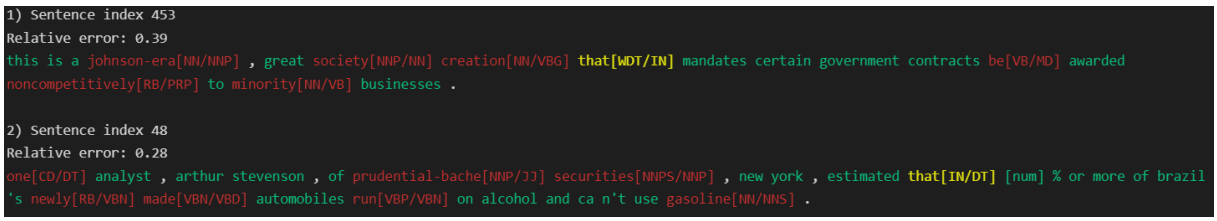

Figure 8: Examples of some high relative error rate sentences containing errors on the `that` token

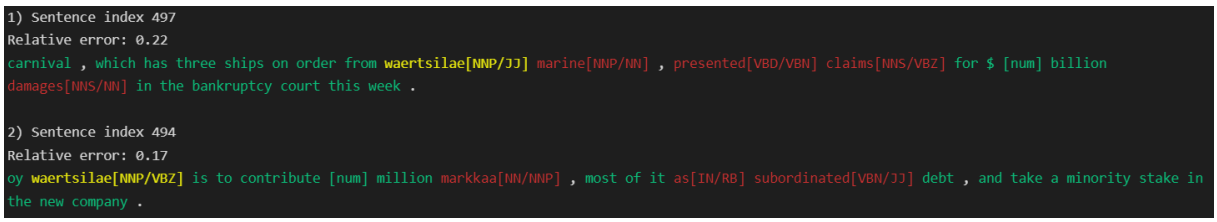

Figure 9: Examples of some high relative error rate sentences containing errors on the `waertsilae` token