Assignment 1

Alessandro Lombardini, Giacomo Melacini, Matteo Rossi Reich and **Lorenzo Tribuiani**
Master's Degree in Artificial Intelligence, University of Bologna
{ alessandr.lombardin3, giacomo.melacini, matteo.rossireich, lorenzo.tribuiani }@studio.unibo.it

Abstract

Part-of-speech (POS) tagging is a popular NLP process that consists in marking up every word in a corpus as corresponding to a particular part of speech. This assignment aimed at experimenting with the problem from the preprocessing to the model architectures and error analysis. This work enabled the use of different RNN models to effectively perform labeling. The experimented techniques lead to good results and allowed a deep understanding of a simple NLP workflow.

1 Introduction

To address this sequence labeling task the corpus was firsts analyzed, split into sentences and then embedded using GloVe. OOV words were dealt with in a static way. Pytorch has been used as machine learning framework to build different models and train them. Moreover, the most promising models have been tuned to improve their performance. Finally, error analysis was performed to better understand the behaviour and criticalities of the models.

2 System description

In order to solve the POS tagging problem, different architectures have been used. The baseline model uses a simple two layers architecture: a Bidirectional LSTM layer and a Fully-Connected layer on top of it. In addition to this, two more models were proposed: one adding a second LSTM layer and another adding a second dense layer. Moreover a fourth one using a GRU architecture was used.

Sentences have then been embedded using Glove with 840B tokens and 300 dimensions. Its Pytorch implementation allows the use of a custom function to deal with OOV which was used to try and reduce their number. To do so the OOV were analyzed and split into the following categories:

1. proper nouns like *Colonsville*;
2. numbers with many decimals or big integers divided by a comma or point like *82.38945* or *10,195,163*;
3. words divided by a dash like *fetal-tissue*;
4. spelling errors like *pathlogy*;
5. words divided by forward or backward slashes like *summer\\/ winter*.

As regards the first category, it is right for proper nouns to be out of vocabulary, so they were kept with the standard behaviour. In the case of the numeric category, embeddings of the integer cast were used. The embedding of the words divided by dashes was obtained by summing the embedding of the splits. For instance: *embedding(fetal-tissue) = embedding(fetal) + embedding(tissue)*. Spelling errors were corrected with a spell-checker. On the other hand, words divided by "\\/" couldn't be easily handled because the semantic meaning of the division depended on the example. For instance, in *summer \\/ winter* it is adversative while in *7 \\/ 8* it is not. In order to farther reduce the padding, a custom batch sampler that put together examples of similar length in the same batch was used.

3 Data

The data fed to the network were preprocessed in order to ease the learning. The distribution of tags between the train, validation and test split seems to be fairly even, however in the splits there is clearly class imbalance. This is to be expected as some parts of speech naturally appear more frequently, but it will pose a problem as in the case of NNP which greatly outnumber NNPS and will lead to classification errors. By analyzing the distribution of lengths of the corpus it becomes evident that it should be divided by sentences rather than documents to reduce the difference in lengths between examples making the padding in the batches much smaller.

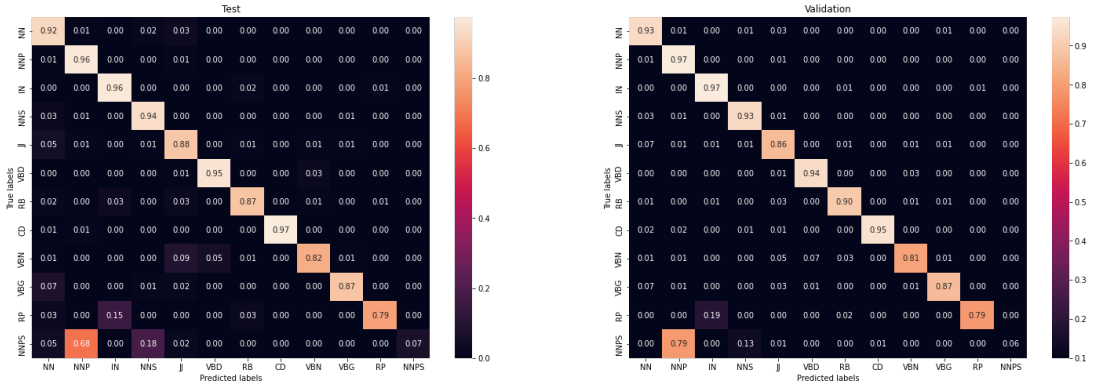

Figure 1: Confusion matrices for test and validation normalized w.r.t. the true labels.

4 Experimental setup and results

All the architectures were first trained with a $batch\_size = 32$ and $hidden\_size = 64$ until they overfitted. By looking at the best loss values of the validation sets, it was decided to tune the baseline model and the GRU model. They reached the best values and are also very simple models and simpler, when keeping performance, is better. The batch size and hidden size were tuned for each model using a grid search. The best set of parameters that came out of the tuning can be seen in table 1.

	batch size	hidden size
GRU	64	128
LSTM	32	128

Table 1: Tuned parameters for GRU and LSTM models

Finally, the tuned LSTM and GRU models have been trained and their performance has been analyzed. As shown in table 2, both models reach a similar F1 macro score in the test set.

	val	test
GRU	0.79	0.86
LSTM	0.74	0.82

Table 2: F1 macro score for tuned GRU and LSTM models

5 Discussion

From table 2, it is possible to see that the GRU model is performing better than the LSTM model. The tuning of the hyperparameters led to a slight improvement of the performance of the classifiers. As it can be seen from image 1, the errors in the train and validation set are similarly distributed. Since the confusion matrix is sorted based on the frequencies of tags in the train set, it is also noticeable that the majority of mistakes happen with rare tags. Looking more accurately at the kind of mistakes that are done, it is possible to observe that they are explainable and are kind of mistakes that also humans could make. For instance, both models have problem at distinguishing particles and prepositions/subordinating conjunction. The other very common mistake is that of classifying plural proper nouns (NNPS) as singular proper nouns (NNP). For instance, *securities* is predicted as NNP while the true label is NNPS. A reason for which the model makes mistakes in tagging NNPS is that there is some mislabeling. For instance in a sentence the word *savings* is labeled as NNP while it should be labeled, being plural, as NNPS.

6 Conclusion

The work done showed how simple recurrent models succeed in POS tagging. Data preprocessing has shown again to be fundamental to reach good results. Even if we found the GRU model to have the best scores, all the models have achieved fair results. It was noticed that the majority of mistakes come from the less represented tags in the train set. For this reason, a possible way to improve the performance of the model could be to apply weights to the examples. In this way less represented classes could gain importance during the training phase. Another way to improve the score would be to correct the labeling mistakes regarding plural proper nouns. The models would then probably be able to recognize them better. On the other hand, it seems there is no need to further increase the architecture complexity of the models because all of them were able to reach a very low bias with some of them reaching 100% accuracy in the training set.