Assignment 1

De Angelis Paolo, Piergentili Giacomo, Pigliapoco Francesco and **Scaioli Lorenzo**
Master's Degree in Artificial Intelligence, University of Bologna
{ paolo.deangelis7, giacomo.piergentili2, francesco.pigliapoco, lorenzo.scaioli }@studio.unibo.it

Abstract

In this experiment we try to solve the POS tagging problem by training three different RNNs with bidirectional LSTM layers. We explain the theory behind this type of structure and the details of our setup. We then evaluate the models, analyze the errors made in detail and categorize them. In this way we confirm the great capacity of LSTM models in sentence analysis. Finally, we propose promising solutions to tackle the task and address the dataset imbalance and the similarity in some POS tagsets.

1 Introduction

There are several approaches for the POS tagger's problem and they can be grouped into four different categories: rule-based, artificial neural network, stochastic and hybrid approaches. One of the most famous approach is the "Hidden Markow Model" which is a stochastic technique. Our algorithm exploits the "Long Short-Term Memory" (LSTM), a recurrent neural network (RNN) designed to handle problems inherent in long-term information flow within data sequences. It introduces mechanisms such as memory cells and control gates to preserve and select relevant information along sequences, making them particularly effective in dealing with long-term sequential data. The advantages of the LSTM tagger over other taggers grow proportionally with the tagsets size (Horsmann and Zesch, 2017). For our experiment we define three models to solve the POS-tagging problem on the Penn TreeBank corpus.

2 System description

This section describes the main steps that best summarize our work on the dataset. Firstly, a text pre-processing pipeline is defined which includes only the text lowercase transformation, because, in GloVe embedding, a very good tokenizer is alredy defined, therefore stemming or lemmatization were not necessary. After that, an embedding matrix is created for words in the dataset using the GloVe model, in particular glove-wiki-gigaword-50. Any out-of-vocabulary (OOV) words in the train set are handled generating random embeddings, while `['UNK']` token is defined for OOV words in the evaluation (Stack Overflow). Lastly, each word in every sentence in the dataset is tokenized and every sentence is padded to have a uniform length.

3 Experimental setup and results

Taking advantage of the LSTM architecture, we defined three models: Baseline, Model 1 and Model 2. All models are sequential neural networks defined in the keras library and they all share the following layers:

- **Embedding layer**: a layer that converts the input vocabulary into the pre-trained GloVe vectors of size 50. The topic weights were configured with a pre-loaded embedding matrix. We chose to keep the weights of the embedding layer fixed to keep the GloVe vectors immutable while preserving the existing semantic information. After the embedding layer we insert a masking layer to ignore padding values in the input.
- **LSTM layer**: a bidirectional LSTM layer of 128 units.
- **Dense layer**: a TimeDistributed Dense layer of 45 units that produces the output of our neural network, useful in temporal sequence processing applications. The `ReLU` activation function is replaced in the last layer by a `softmax`, typical of multiple classification.

Model 1 extends the Baseline by adding another bidirectional LSTM layer before the output layer while Model 2 introduces a dense layer, both having 128 units. For the training we chose

`AdaDelta` as optimizer with a starting learning rate of 1, and `categorical_crossentropy` as loss function. We train each model four times setting different random seeds in order to check robustness.

Validation set	f1	precision	recall
Baseline	0.6323	0.6593	0.6171
Model 1	0.6540	0.6878	0.6456
Model 2	0.6789	0.7120	0.6772

Table 1: evaluation of the macro metrics of the models on validation set with the second seed.

4 Discussion

The results in Table 1 show that both Model 1 and Model 2 have better values on all three of the metrics analyzed compared to the Baseline model. It is interesting to see that the three metrics grow together and are strongly related to each other. From the evaluation of the models across four random seeds we saw that Model 2 always perform better than the other on all the metrics. Therefore we conduct the error analysis on Model 2. As expected, the value of the metrics on the test set are lower than on the validation set, but the difference is so little that it can not be related to overfitting (Appendices Table 2).
During error analysis we found two main groups of misclassified POS tagset:

- Noun singular or mass (NN), noun plural (NNP), proper noun singular (NNS), proper noun plural (NNPS) and adjective (JJ).
- Verb past tense (VBD), verb gerund/present participle (VBG), verb past participle (VBN) and adverb (RB).

It is clear that the POS tags in each group are very similar to one another and therefore it is quite obvious that the model struggles to find differences between the categories. Two possible solutions to face this type of error are either improving the embedding layer or dividing the classification process. The embedding layer can be improved in many ways: we could use a more detailed wiki-glove of dimension 300 that better represents each word or we could directly use different embedding representations such as Common Crawl and Twitter. Another solution is grouping similar classes into macro-categories and dividing the classification process into two parts: the first part, where a first model classifies the word into a macro-category such as adjectives, nouns, verbs, etc. and the second part, where a second model classifies the word into a more specific category between those related to the previously defined macro-category, allowing the model to learn the most peculiar differences between similar classes.
Another type of misclassification is related to less frequent POS tags. In fact, by plotting the mean of each metric over the 8 most and least frequent POS tags, we found out that the most frequent tags reach about 80% precision during the first 20 epochs, while the least frequent POS tags struggle to reach 35% precision and start to be correctly classified only after 15 epochs (Appendices Figure 1). This type of error is quite common on unbalanced datasets because the model trains more on the most frequent POS tags and almost ignores the less frequent ones. This is due to the fact that the loss function of the training is the categorical cross entropy on each tag processed, regardless of its true label. As a result, the model learns to correctly classify the most frequent tags in the first epochs of the training while it learns the least frequent tags later in the training, having already achieved good accuracy on the most frequent tags. Possible solutions for this type of error could be to train the model for more epochs or to change the loss function according to the frequency of the classes. A last solution could be data augmentation, a technique that modifies the dataset by adding copies of sentences, where rare tags occur, with oversampling approaches (Sáez et al., 2016).

5 Conclusion

In this experiment we trained three different models to solve the POS tagging problem and analysed the results of the best model. It is well known in natural language processing that RNNs are very useful tools to deal with sentence analysis, and it was interesting to see that even the Baseline model was able to achieve good accuracy. We were surprised by the small differences between the bigger models and the Baseline: less than 5% on macro f1. However, we attribute this small increase to the coarse embedding layer chosen for the experiment. The error analysis revealed two main sources of error: the similarity in the meaning of many POS tags, and the highly unbalanced data set. However, we are confident that the solutions proposed in section 4 will overcome these problems.

6 Links to external resources

Here's the link to our GitHub repository with the code and all the trained models: https://github.com/LorenzoScaioli/NLP_POS- tagging.

References

Tobias Horsmann and Torsten Zesch. 2017. Do LSTMs really work so well for PoS tagging? – a replication study. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 727–736, Copenhagen, Denmark. Association for Computational Linguistics.

José A. Sáez, Bartosz Krawczyk, and Michał Woźniak. 2016. Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets. *Pattern Recognition*, 57:164–178.

Appendices

Model 2	f1	precision	recall
Validation set	0.7135	0.7522	0.7090
Test set	0.7017	0.7250	0.6932

Table 2: evaluation of the macro metrics of the best model (Model 2 with the third seed) on validation and test set.

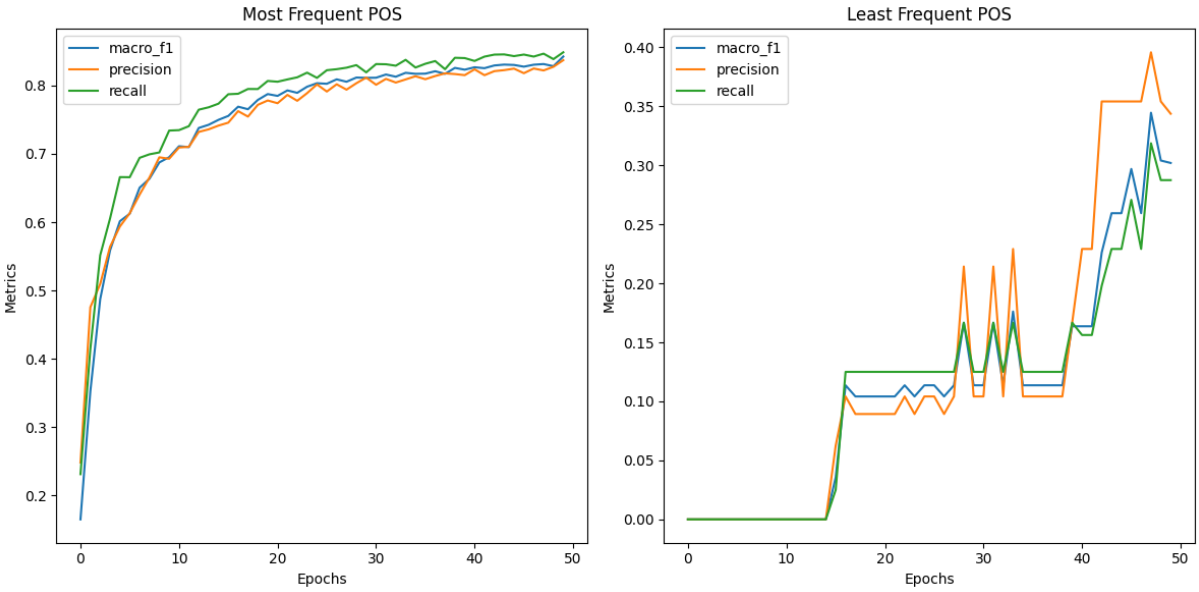

Figure 1: This graph represents the development across the training epochs of macro-f1, macro-precision and macro-recall of the 8 most and least frequent POS.