Assignment 1

Mauro Dore, Giacomo Gaiani, Gian Mario Marongiu and **Riccardo Murgia**
Master's Degree in Artificial Intelligence, University of Bologna
{ mauro.dore, giacomo.gaiani, gianmario.marongiu, riccardo.murgia2 }@studio.unibo.it

Abstract

In this report we analyze the behavior of three different architectures based on Long Short Term Memory networks (LSTMs) trained to accomplish the Part-of-speech (POS) tagging. The study included a performance comparison of the networks, an analysis of how the presence of punctuation contributed to the accuracy of the models, how they perform using different weight strategies and what are the most common prediction errors on the validation and test set of the best performing model. It was observed that the F1 score difference between the models is minimal, and the prediction errors on both validation and test set showed similar trends.

1 Introduction

Over the years, there have been many proposals on how to tackle the POS tagging task. The most familiar approches (Chiche and Yitagesu, 2022) are rule-based, stochastic, artificial neural networks and hybrid approaches. The disadvantage of rule-based approaches is that they require a deep knowledge of the language and do not work well when the new input text is not known (Kumawat and Jain, 2015). The stochastic approaches exploits the probability of a given word belonging to a specific class, the disadvantage of this system is that some sequences of tags can come up for sentences that are not correct according to the grammar rules of a certain language (Kumawat and Jain, 2015). Between 2019 and 2021 has been observed (Chiche and Yitagesu, 2022) that 68% of the proposed approaches are based on deep learning methods. In order to reach the best results they may require a large amount of training data, furthermore, they may be computationally expensive to train and prone to overfitting if the training data is not representative of the test data (Sunita et al., 2023). The approach described in this report belongs to this last category and it is divided in the following main steps:

1. Pre-processing and data visualization: the dataset is merged into a single dataframe. The sentences are split, using the period as separator.

2. Embedding: The embedding has a length of 300, and it is automatically performed using GloVe. Every word in the training set receives a distinct embedding, except for the terms that are in the training set but not in GloVe, which take a random Out-of-GloVe (OOG) embedding that is different for every term. Each word in the validation set but not in the training set takes a fixed Out-of-Vocabulary (OOV) embedding that is the same for every word.

3. Creation of the BiLSTMs and evaluation: iteratively, the three models are created, fine tuned and validated on the validation set with three different seeds to guarantee reproducibility and robust estimation of the F1 macro score computed over all tokens. Then, given the best performing model, a comparison is carried out between the prediction errors on the validation and test set.

2 System description

The models are coded using the Pytorch frameworks, they are defined as follows:

- Baseline: a Bidirectional LSTM with a Dense layer on top.
- Model 1: two Bidirectional LSTMs with a Dense layer on top.
- Model 2: a Bidirectional LSTM with two dense layers on top. Between the dense layers a ReLu activation function is placed.

3 Experimental setup and results

The table (1) shows how each model performed on the validation and test sets. These mod-

Model	F1_val	F1_test	std_deviation
Baseline	0.770	0.830	0.010
Model1	0.766	0.835	0.010
Model2	0.775	0.839	0.008

Table 1: F1 macro score on evaluation and test set. The standard deviation is computed on the validation set averaging over 3 seeds.

els were trained using the *torch.Adam* optimizer with the implementation of a custom early stopping. A grid search is performed to find the best model hyperparameters: batch size $(32, 64)$; hidden size of the LSTM $(100, 200, 300, 400)$; learning rate $(0.01, 0.001, 0.0001)$; weights policy (W_0, W_1, W_2). In particular, the weight policy determine the importance of each class in the computation of the loss function.

1. W_0: every class (punctuation included) has weight 1.
2. W_1: weights of the punctuation classes are set to 0, every other class has a weight inversely proportional to its occurrence.
3. W_2: every class (punctuation included) has a weight inversely proportional to its occurrence.

4 Discussion

As expected, we observed that the punctuation was an important parameter to include during the training phase, which is why the models that used W_1 never achieved the higher F1 score, all else being equal. We also noticed that with W_2 the model tended to reduce the prediction errors on the rarest classes but the overall F1 did not change in a significant way. Finally, W_0 resulted in the more accurate strategy when the goal was the overall F1 score without regard to rare classes. All the models led to similar results. The best model on the test set seems to be Model2, both for its F1 score and for its standard deviation, while the Model1 works well on the validation set. The Model2 prediction errors on the validation and test set follow a similar trend. These errors can be divided into three macro-categories:

1. The classes ignored in the computation of the metric (punctuation and padding class) are often miss-classified, as expected.
2. The rarest classes because of their scarce occurrence are sometimes miss-classified.
3. The most common classes are rarely miss-classified, but when it happens it is often because of their similarity (e.g. NNP with NNPS or RP with IN).

5 Conclusion

The three models showed similar performances and none of them outperformed the others in a significant way, even though Model2 showed a slightly higher precision and stability (lower standard deviation). The main limitation of the implemented solutions were caused by the characteristics of the dataset. The number of word occurrences for each class with respect to the total number of classes was not satisfactory, moreover, there is an high unbalance in the distribution of terms for each class. Better performances might be obtained by extending the vocabulary with all the words contained in GloVe, even those who are not present in the training set. Moreover, it might be useful to enhance the datasets with more balanced data. With that being said, with a bigger dataset it might be useful to extend the preprocessing phase with some normalization techniques in order to reduce the size of the final vocabulary. Another promising technique might consists in developing an hybrid approach by defining a set of rules to identify the possible POS-tags of each word, as described in (Tamburini, 1970).

6 Links to external resources

The full code can be found at: https://github.com/GianM0027/NLP_homework_1

References

Alebachew Chiche and Betselot Yitagesu. 2022. Part of speech tagging: a systematic review of deep learning and machine learning approaches. *Journal of Big Data*, 9(1):10.

Deepika Kumawat and Vinesh Jain. 2015. Pos tagging approaches: A comparison. *International Journal of Computer Applications*, 118(6).

Sunita Sunita, Ajit Kumar, and Neetika Neetika. 2023. A Comprehensive Survey of Techniques Used for Part-of-Speech Tagging of Code-Mixed Social Media Text.

Fabio Tamburini. 1970. (better than) state-of-the-art pos-tagging for italian texts.