Assignment 1

Riccardo Marvasi, Edoardo Saturno, Lucia Gasperini, and **Arianna Albertazzi**
Master's Degree in Artificial Intelligence, University of Bologna
{riccardo.marvasi, edoardo.saturno, lucia.gasperini, arianna.albertazzi3 }@studio.unibo.it

Abstract

Part-of-Speech (PoS) tagging is a fundamental natural language processing task that involves assigning grammatical labels to words in a text, indicating their syntactic roles. Our paper explores and compares some recurrent neural architectures for POS labeling, with the aim of finding the model that guarantees the best effectiveness in terms of F1 score. To this end, we consider the English Penn TreeBank dataset (Marcus et al., 1993) as a benchmark to train and evaluate our proposals and for further robustness we evaluate each model on three seeds. The study includes both qualitative and quantitative error analyses, providing insights into the different performances of our models compared to a baseline.

1 Introduction

The main approaches to PoS tagging can be broadly categorized into three groups: rule-based, feature-based, and neural (Kumawat and Jain, 2015). Rule-based and feature-based methods, while computationally efficient and straightforward, lack predictive capabilities and struggle with capturing complex information, limiting their ability to generalize (Kumawat and Jain, 2015). In contrast, neural models approach this task by learning hierarchical abstract representations of data, for this reason their computation may be really intensive, but they highly effective results. Hence, we opted for the neural approach to address this problem.

In our approach, we encode the corpus using *GloVe embeddings* to provide the models with vectors that encapsulate semantic relationships. Subsequently we implemented a baseline and two recurrent neural models. The selection of recurrent models is motivated by their innate ability to capture sequential dependencies, making them good at discerning intricate linguistic patterns within text.

2 System description

In this section, we present a detailed overview of our pipeline. The initial step involves encoding the corpus into a dataframe. Some elements in the original dataset contained more than one sentence, separated by a blank line. To manage input dimensions effectively, we opt to split these multi-sentence elements, ensuring that each element in our dataframe corresponds to an individual sentence. As a pre-processing step, we lowercased all sentences to align with GloVe embeddings, which are available exclusively for lowercase terms. However, we refrained from employing other common pre-processing mechanisms, such as stemming or punctuation removal, as they can be deleterious for PoS tagging applications. Moving to the model's phase, we initiate by tokenizing all sentences in the dataset. Given their varying lengths, padding becomes necessary. After careful consideration and a thorough study to minimize information loss, we set a maximum sentence length of 50 words, choosing to truncate the few sentences exceeding this limit. Our baseline model is defined as a bidirectional LSTM with a dense layer on top. Building upon this baseline, we introduce two additional models (*Model1* and *Model2*) that differ by the inclusion of an extra LSTM layer (the first one) and an additional dense layer (the second one). *Figure 1* shows a simplified sketch of the three architectures.

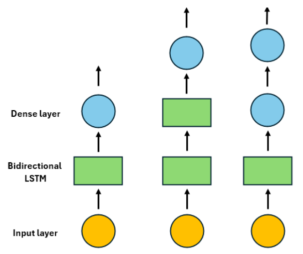

Figure 1: Simplified sketch of the three models

3 Experimental setup and results

The three models collectively share a fundamental architectural framework. Each of them is a recurrent neural network suited for multi-class classification, integrating an embedding layer with pretrained GloVe weights, a masking layer for handling padding, and a bidirectional LSTM layer for capturing sequential dependencies. Additionally, the final layer is consistently a dense layer with softmax activation. All models are compiled with categorical crossentropy loss, and use precision as optimization metric during training. This choice is motivated by an empirical analysis.

In *Model 1* we introduce an additional LSTM layer with 128 neurons, with the goal of enhancing the ability of the model to capture intricate sequential patterns, but also adding complexity. In *Model 2*, we incorporate an extra dense layer after the bidirectional LSTM, utilizing a ReLU activation function. This modification offers a distinct approach to feature extraction compared to the baseline, introducing a non-linear transformation to the data. Across all models, we chose Adam as the optimizer due to its consistently superior results and stability during training. To facilitate a comparative performance analysis, identical batch sizes and numbers of epochs are employed for all three models. These values are carefully selected based on extensive experimentation and the results obtained on the validation set. Table 1 shows the Macro F1 score (for each seed) of the three models and *Table 2* shows the results of the best-performing model.

4 Discussion

We excluded from the computation of the metrics all the punctuation tokens. The quantitative results indicate a comparable performance among the three models, with minimal differences observed in the macro F1 score. Surprisingly, the *baseline model*, despite its simplicity, with one less layer respect to the other two, performs nearly as well as the other models. It is reasonable to think that the advantages gained from introducing an additional LSTM layer in *Model 1* may be offset by the increased complexity, potentially leading to overfitting on the training set. A similar reasoning stands for the *Model 2*. The additional dense layer used to change the way in which the features are extracted does not present substantial benefits compared to the baseline simpler method with just one dense layer.

From a more in-depth analysis observing the confusion matrices, obtained using our models on the test set, it emerges that all models encounter significant problems with tags that are extremely underrepresented (frequency in the training set lower than 0.5%). Also it emerges clearly how tags referred to nouns (that are the most frequent) are the ones that accumulate more false positives, mainly coming from verbs and adjectives. This highlights models' struggle to effectively discern the distinctions between those three categories of parts of speech, while other categories, like determiners are really well discerned.

5 Conclusion

In conclusion, the observed performance of the three presented architectures is notably similar for this specific task. While this outcome might be anticipated given their intrinsic similarities, the competitiveness of the baseline model was somewhat unexpected. It suggests that the baseline architecture is already adept at learning a robust representation of the data for this task. Our solution are limited by the use of a small dataset, in which many tags were extremely under-represented.

Despite these challenges, the promising results observed for the more frequent tags (see Table 3) instill optimism. There's hope that these models, or some variation, hold the potential to achieve noticeable outcomes in PoS tagging applications, particularly with larger and more balanced datasets.

Model	**Seed_2**
Models2_s1	0.850

Table 2: Macro F1 scores of the best model in the test set.

Model	**Seed_1**	**Seed_2**	**Seed_3**
Baseline	0.768	0.771	0.772
Model 1	0.751	0.766	0.755
Model 2	0.759	0.768	0.780

Table 1: Macro F1 scores of the three models for all the seed in the validation set. Batch size = 32, 24 epochs

Most frequent tags	**Baseline**	**Model 1**	**Model 2**
NN	0.88	0.87	0.87
NNP	0.86	0.84	0.84
IN	0.98	0.97	0.98

Table 3: Macro F1 scores for the most frequent tags computed on validation

References

Deepika Kumawat and Vinesh Jain. 2015. Pos tagging approaches: A comparison. *International Journal of Computer Applications*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*.