Assigment 1

Giorgia Castelli, Alice Fratini and **Madalina Ionela Mone**
Master's Degree in Artificial Intelligence, University of Bologna
{giorgia.castelli2, alice.fratini2, madalina.mone}@studio.unibo.it

Abstract

This report explores the task of POS tagging using neural networks, specifically focusing on variations of BiLSTM models. Three different models were developed: a baseline BiLSTM model with a Dense layer, a variant with an additional LSTM layer, and another variation of the baseline with an additional Dense layer. The models were trained using GloVe embeddings to initialize the word vectors, with special handling for OOV tokens.

Our experiments revealed that adding layers to the baseline model improved performance. However, these improvements came with a trade-off in terms of model complexity and the risk of overfitting.

1 Introduction

Part-of-Speech (POS) tagging is a crucial task in natural language processing (NLP) that involves labeling words in a text with their corresponding grammatical categories, such as nouns, verbs, and adjectives. This process is foundational for various NLP applications, including syntactic parsing, machine translation, and sentiment analysis, as it helps in understanding the structure and meaning of sentences.

For addressing this problem we used BiLSTM RRN which is commonly used in the literature. The aim of this project is to achieve a better F1 score, specifically for the TAGs with higher frequency that have a low score.

2 System description

2.1 Model Architecture

In this work, three neural network models were implemented to tackle the task of POS tagging.

Baseline Model (BiLSTM with Dense Layer): It consists of a Bidirectional LSTM layer. The output from the BiLSTM layer is then passed through a Dense layer with a softmax activation function. This Dense layer acts as the classifier, mapping the BiLSTM outputs to the corresponding POS tags.

Model 1 (BiLSTM with an Additional LSTM Layer): This model add an LSTM layer on top of the initial BiLSTM layer. The additional LSTM layer improves the model's ability to generalize across different linguistic structures.

Model 2 (BiLSTM with an Additional Dense Layer): The second variant of the baseline model retains the original BiLSTM layer but introduces an additional Dense layer before the final classification layer.

All models use GloVe embeddings for initial word representations, providing semantically meaningful input to the BiLSTM layers. They are trained with categorical cross-entropy loss and optimized using the Adam optimizer, ideal for managing varying gradients in neural networks.

2.2 Tokenization and Vocabulary Creation

We performed word-level tokenization, breaking sentences into individual words and mapping each to 50-dimensional GloVe embeddings for effective POS tagging. Out-of-vocabulary tokens were assigned a special <UNK> token with randomly initialized vectors, enabling the model to handle unseen words during training and inference.

3 Experimental setup and results

3.1 Setup

The experimental setup was designed to systematically evaluate the performance of different neural network architectures for POS tagging. Below is a detailed description of the setup, including the architectures, hyperparameters, and evaluation methods used:

- The learning rate was initially set to 0.001 for all models. This value was chosen based on common practices for training neural networks and adjusted slightly based on preliminary experiments.

- Batch Size: A batch size of 64 was used across all experiments, balancing the need for computational efficiency and model stability.
- Epochs: Each model was trained for 20 epochs, with early stopping implemented based on the validation loss to prevent overfitting.
- Sequence Length: Input sequences were padded or truncated to a fixed length of 100 tokens to standardize the input across all models.

3.2 Results

Model 1 slightly outperforms the base model, achieving the highest average performance (0.849), suggesting that the additional complexity improved generalization. However, Model 2's performance is comparable to the base model, indicating that its modifications may not provide significant benefits.

Model	F1 Score
Baseline model	0.844
Model 1	**0.850**
Model 2	0.843

Table 1: Performance comparison of models.

4 Discussion

Our experiments demonstrated that increasing model complexity generally led to improved performance, particularly in terms of F1 scores, which indicates a better balance between precision and recall. The model with an additional LSTM layer (Model 1) showed enhanced ability to capture sequential dependencies, leading to higher accuracy and F1 scores on both the training and validation datasets. Similarly, the model with an additional Dense layer (Model 2) improved the model's capacity to learn complex representations, further refining its POS tagging performance.

However, the trade-off between model complexity and generalization was evident. While the more complex models performed better on the training data, they also required careful tuning to avoid overfitting, as indicated by the slightly lower performance on the validation set compared to the training set.

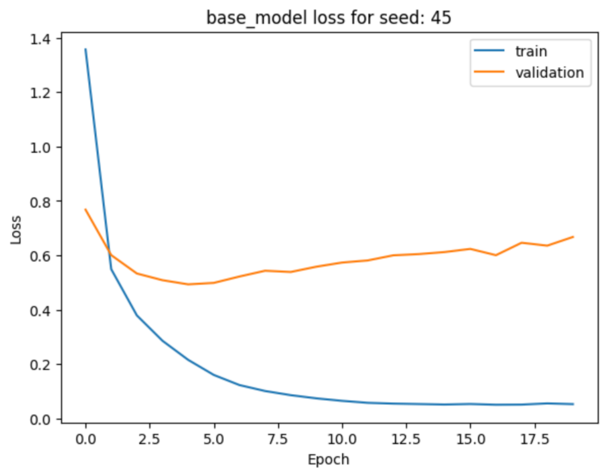

Figure 1: Baseline model loss for seed 45 .

5 Conclusion

In this study, we explored different neural network architectures for the task of POS tagging, specifically focusing on variations of the LSTM model. We implemented a baseline model featuring a simple BiLSTM layer followed by a Dense layer, and extended this architecture by adding either an additional LSTM layer or an additional Dense layer.

In summary, this work highlights the importance of balancing model complexity with the need for generalization in NLP tasks like POS tagging. The results suggest that while more sophisticated models can offer better performance, particularly in capturing complex linguistic patterns, they must be managed carefully to ensure that they generalize well to unseen data. Future work could focus on optimizing these models further, possibly through more advanced pre-processing techniques, for istance OOV terms can be handled using the neighbourhood strategy to calculate the embedding-matrix or handling numerical tags with special tokens.