Token-Level Part-of-Speech Tagging Using Recurrent Neural Networks on the POS 1 Dataset

Group 7: Shadman Salif Swanan(22101573),Showrin Rahman(24141255), and Arpita Ghosh(24341213)

Keywords: LSTM, RNN, GRU ,POS 1

Abstract: For our project, we utilized the POS 1 dataset to carry out Part-of-Speech (POS) tagging through the implementation of four different neural network architectures: Simple Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Bidirectional LSTM (BiLSTM). We began by exploring and preprocessing the dataset, then proceeded to construct the models using TensorFlow and Keras. Each model was fine-tuned and evaluated based on metrics such as accuracy, macro F1-score, confusion matrices, and classification reports. Among all the models, BiLSTM yielded the highest performance, reaching an accuracy of 97% and a macro F1-score of 0.849. This was closely followed by GRU, which attained 96% accuracy and a macro F1-score of 0.836. RNN came next, while LSTM had the lowest macro F1-score at 0.803.

1 Introduction

In natural language processing, POS tagging is one of the vital tasks. It labels words in a sentence using parts-of-speech tags. In order to compare four recurrent neural network models Simple RNN, LSTM, GRU, and BiLSTM—we were given the POS 1 dataset for this study. **The Simple RNN** serves as the foundational recurrent model, capturing basic sequential dependencies [1]. **The LSTM (Long Short-Term Memory)** network addresses the vanishing gradient problem and is designed to capture long-term dependencies more effectively [2]. **The GRU (Gated Recurrent Unit)** is a simplified variant of LSTM, offering similar performance with fewer parameters [3]. **The BiLSTM (Bidirectional LSTM)** enhances the context understanding by processing the sequence in both forward and backward directions [4]. By evaluating and comparing each model's performance on the dataset, we wanted to identify the optimal model for POS tagging.

2 Methodology

2.1 Dataset Description

The POS 1 dataset is split into two parts: training and testing sets.

- There were 19,183 sentences in the training set and 4,796 in the testing set.
- The test set was kept for the final evaluation, while the training set was used for data analysis and model training.

The dataset contains three columns:

- **Sentence #** – A sentence's serial number appears in the first column.
- **Sentence** – The sentence itself appears in the second column.
- **POS tags** – The third column contains the specific POS tags for each sentence.

The dataset had no missing information. However, further investigation revealed that the number of tokens and the number of POS tags for three sentences did not match. There were 41 unique POS tags, with a total of 418,968 tags in the training set. Tags such as `NN`, `NNP`, and `IN` were frequent.

We created bar plots to show the sentence length distribution, top 10 most common words, and top 10 most common POS tags which helped us understand the data.

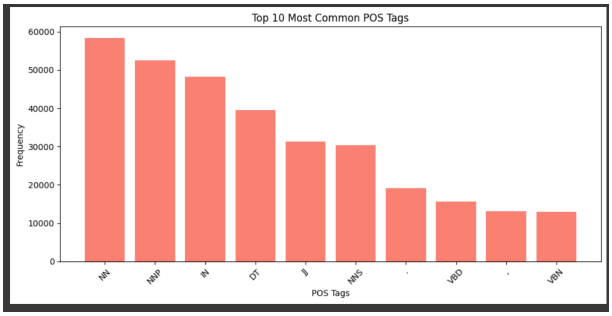


Figure 1: Sentence length distribution.

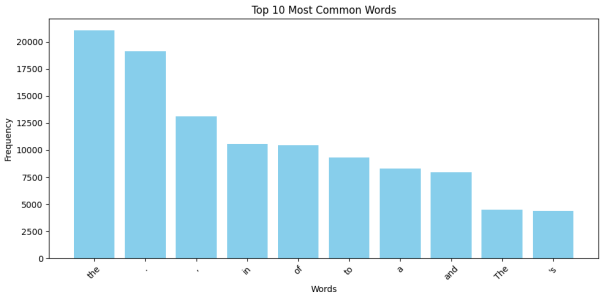


Figure 2: top 10 most common words

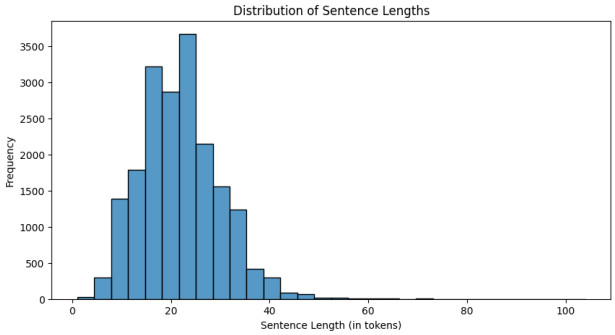


Figure 3: top 10 most common POS tags

2.2 Data Preprocessing

The following preprocessing steps were applied to the dataset:

- Dropped training data with mismatching number of tokens and POS tags.
- Converted POS tag strings to lists.

- Used Keras `Tokenizer` to convert words to numbers [5].
- Encoded POS tags as numbers using `LabelEncoder` [6].
- Padded all sentences to 100 words for consistent input [**?**].
- The model used an `Embedding` layer to convert words into 300-dimensional vectors, which are learned during training [7].

The training data was split into 80% for training and 20% for validation to tune the models.

3 Model Architectures

We implemented four deep learning models using TensorFlow [8] and Keras [9].

3.1 Input Layers

Each model begins with an `Embedding` layer to convert integer word sequences into dense vector representations [7].

- The embedding dimension was set to 300.
- The embedding layer was trainable, allowing the model to fine-tune embeddings during training .

3.2 Recurrent Layers

- **Simple RNN:** A single `SimpleRNN` layer with 32 units was used with `return_sequences=True` to generate a sequence of outputs for time-distributed tagging [10].
- **LSTM (Long Short-Term Memory):** This model used one `LSTM` layer with 32 units and `return_sequences=True` [11].
- **GRU (Gated Recurrent Unit):** A single `GRU` layer was used, configured similarly to the LSTM with 32 units and `return_sequences=True` [12].
- **BiLSTM (Bidirectional LSTM):** One `Bidirectional` wrapper was applied to a single `LSTM` layer with 32 units to capture both forward and backward contextual information in sequences [13].

3.3 Output Layer

Each model ends with a `TimeDistributed(Dense(...))` layer using the softmax activation function to output probabilities for each POS tag at each time step [14].

3.4 Hyperparameters and Training

The hyperparameters and training configuration for the models were as follows:

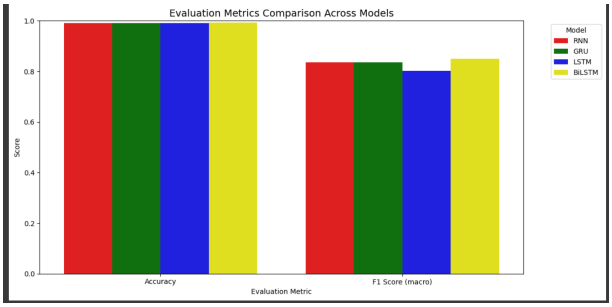
- **Embedding Dimension:** 300
- **Units in Recurrent Layers:** 32
- **Batch Size:** 256
- **Epochs:** Up to 15
- **Optimizer:** Adam with a learning rate of 0.001 [15].
- **Early Stopping:** Enabled with a patience of 2 epochs, restoring the best model [16].

Model Evaluation Results

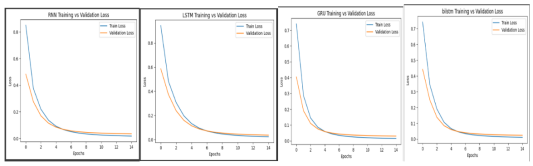
We evaluated the models on the test set using accuracy, macro F1-score, confusion matrices, and classification reports. The results are shown below:

Model	Accuracy (%)	Macro F1-Score	Weighted F1-Score
Simple RNN	0.96	0.835	0.96
LSTM	0.96	0.836	0.96
GRU	0.96	0.803	0.96
BiLSTM	0.97	0.849	0.97

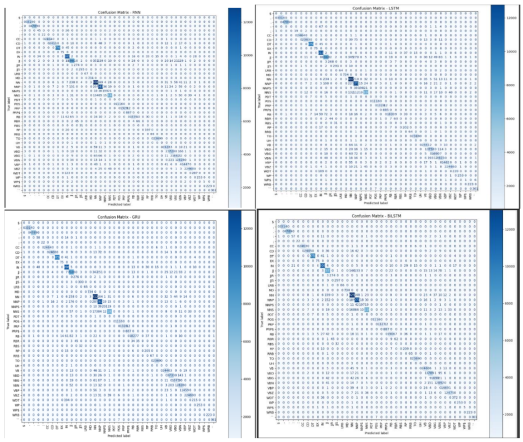
Table 1: Comparison of model performance using accuracy, macro, and weighted F1-scores



Training and Validation Loss Curve:



Confusion Matrix:



Conclusion

In conclusion, because it includes context from both sentence directions, the BiLSTM model performed the best for POS tagging. Long sequences were difficult for the LSTM model to handle. GRU and simple RNN both did reasonably well. This experiment demonstrates that advanced RNN models, such as BiLSTM, are quite good at POS tagging.

Despite BiLSTM's great efficacy, all models have drawbacks and have trouble differentiating between tags that are quite similar. Additionally, some tags in the dataset have extremely low

frequencies, which limits the effectiveness of less common tags. Additionally, capturing the sequence improved the performance of GRU and BiLSTM. It has drawbacks, too; redesigned models such as BERT may perform better with better context capture.

Both feature engineering and hyperparameter optimisation can constantly be improved. We can add data for less common tags, give each model a different sensitivity, merge several models, and so on to further enhance the outcomes.

References

[1] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[4] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," in *IEEE Transactions on Signal Processing*, vol. 45, no. 11. IEEE, 1997, pp. 2673–2681.

[5] F. Chollet, "Keras: The python deep learning library," *https://keras.io/*, 2015.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Duchesnay, "Scikit-learn: Machine learning in python," p. 2825–2830, 2011. [Online]. Available: https://scikit-learn.org

[7] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, *Distributed Representations of Words and Phrases and their Compositionality*, 2013. [Online]. Available: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[8] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," https://www.tensorflow.org, 2015.

[9] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, p. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[11] ——, "Long short-term memory," *Neural Computation*, vol. 9, p. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[12] K. Cho, Y. Bengio, O. Vinyals, M. Schuster, N. Shazeer, Y. Wu, Z. Zhang, Q. V. Le, and W. Zaremba, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014. [Online]. Available: https://arxiv.org/abs/1406.1078

[13] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, p. 602–610, 2005. [Online]. Available: https://doi.org/10.1016/j.neunet.2005.06.042

[14] G. E. Hinton, "A practical guide to training restricted boltzmann machines," *Neural Networks: Tricks of the Trade*, p. 599–619, 2012. [Online]. Available: https://www.springer.com/gp/book/9783642231107

[15] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations (ICLR)*, 2015. [Online]. Available: https://arxiv.org/abs/1412.6980

[16] L. Prechelt, "Early stopping - but when?" *Neural Networks: Tricks of the Trade*, pp. 55–69, 2012. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-23594-8_4