Arabic POS Tagging

1. Introduction

Part-of-Speech (POS) tagging, the process of assigning grammatical categories to words, is a fundamental task in Natural Language Processing (NLP). For morphologically rich languages like Arabic, this task is particularly challenging due to high levels of ambiguity from complex word structures and unwritten short vowels. This project tackles Arabic POS tagging by comparing a traditional rule-based method against several deep learning architectures, including RNNs, LSTMs, and Transformers. The goal is to evaluate the performance gains from context-aware models and visualize the results using NetworkX.

2. Data Description

The project utilized **998 sentences** from the Arabic Parallel Universal Dependencies (PUD) test set (`ar_pud-ud-test.conllu`). The data was preprocessed by removing diacritics and tokenizing sentences. The vocabulary was built to handle 16 unique POS tags. All sequences were padded to a maximum length of 257 tokens. The dataset was split into a 70% training set (698 sentences) and a 30% testing set (300 sentences).

Dataset: Arabic-PUD from Universal Dependencies.

3. Baseline Experiment: Rule-Based Analysis

3.1. Goal & Methodology

The goal was to establish a performance baseline using a non-machine learning approach. We used the **CAMeL Tools** library with its Modern Standard Arabic morphology database. For each word, the analyzer returns possible morphological analyses; we simply chose the POS tag from the first result. This lexicon-based method does not use sentence context for disambiguation.

3.2. Results & Conclusion

The baseline tagger was fast but highly limited. Its primary weakness is its inability to use context, leading to frequent errors with ambiguous words. It also performs poorly on dialectal or out-of-vocabulary (OOV) words. These limitations highlight the need for context-aware deep learning models.

Token	Predicted POS Tag
عايز	noun
أروح	verb
Google	foreign
بالتاكسي	noun
،	punc
بس	verb

Table 1: Sample output from the CAMeL Tools baseline POS tagger.

4. Deep Learning Experiments

To overcome the baseline's limitations, several deep learning models were developed. All models used pre-trained **AraVec** word embeddings (300-dimensions) to represent words as dense vectors.

4.1. Data Preprocessing

To prepare the data for the deep learning models, a comprehensive pipeline was executed. This process was organized into two main parts: preprocessing the input text and preprocessing the target labels, followed by splitting the data for training and testing.

Input Text Preprocessing

The raw Arabic sentences were converted into a numerical format through the following steps:

- **Tokenization:** Sentences were tokenized into individual words. Before this, all diacritics (short vowels and markings) were removed to simplify the input.
- **Sequence Conversion:** A vocabulary was built from all unique words, and each word was mapped to a unique integer index, converting text sequences into numerical ones.
- **Padding:** To ensure uniform input size for the neural networks, all tokenized sentences were padded with a special value to a maximum length of 257 tokens.

Target Preprocessing and Data Splitting

- **Label Encoding**: The string-based POS tags (e.g., "NOUN", "VERB") were converted into unique integer representations.
- **One-Hot Encoding:** These integer labels were then transformed into one-hot vectors. Each tag was represented as a binary vector with a '1' at the index corresponding to its integer value and '0's elsewhere, a format required for categorical classification.

Data Splitting

After all preprocessing, the dataset was split into a 70% training set (698 sentences) and a 30% testing set (300 sentences) for model development and evaluation.

4.2.Experiment 1: Simple RNN

A basic sequence model was built using a stack of four SimpleRNN layers. After 10 epochs, it achieved a test accuracy of **93.82%**, demonstrating a massive improvement over the baseline by learning to use word context

4.3. Experiment 2: BiLSTM

To better capture long-range dependencies and use both past and future context, a model with four stacked BiLSTM layers was implemented. It slightly outperformed the RNN, achieving a test accuracy of **94.16%**, confirming the benefit of bidirectional processing.

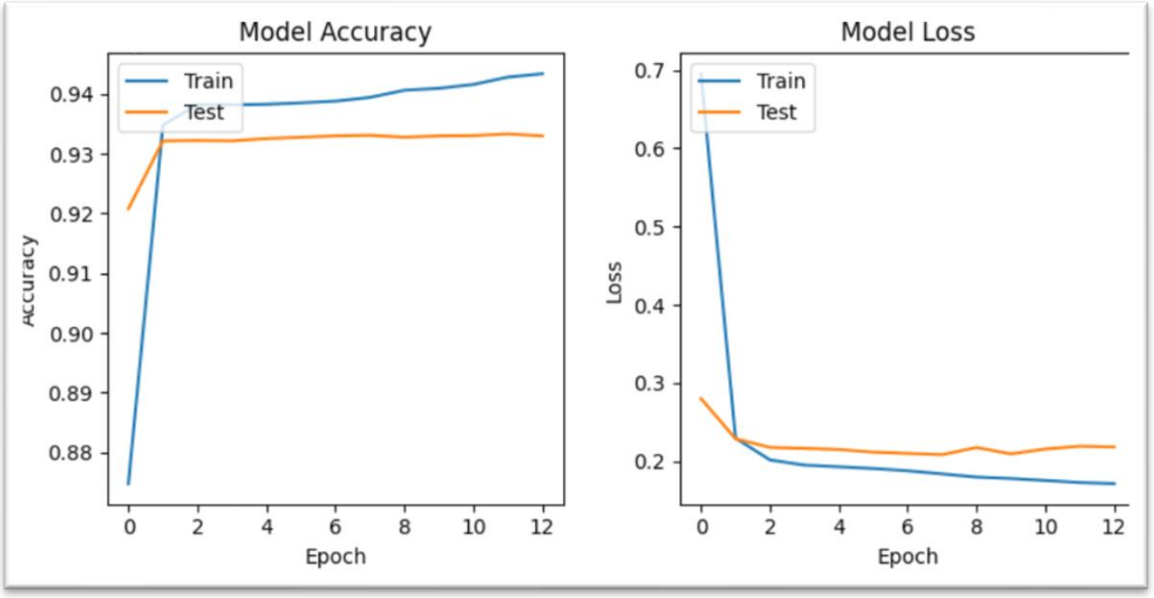

Figure 2: BiLSTM Model Performance.

4.4.Experiment 3: Hybrid CNN + BiLSTM Model

This hybrid model used a 1D CNN to extract local n-gram features from embeddings, which were then fed into a BiLSTM layer for sequential modeling. This architecture yielded the best performance, with a test accuracy of **94.44%**, suggesting a powerful synergy between local feature extraction and sequential analysis.

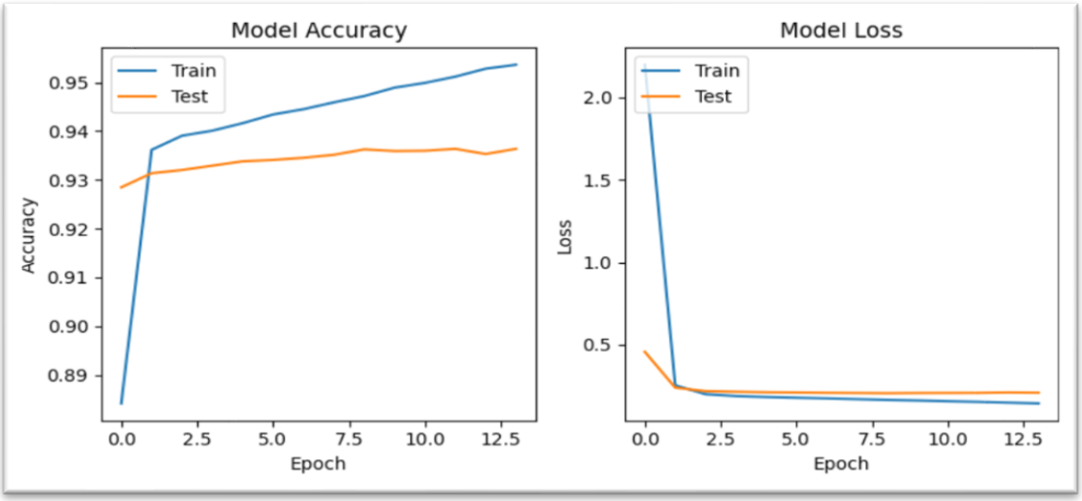

Figure 2: Hybrid CNN+BiLSTM Model Performance

4.5. Experiment 4: BiGRU

A BiGRU model, a simpler variant of the BiLSTM, was also tested. It achieved a test accuracy of **94.05%**, performing nearly as well as the BiLSTM. This indicates that the simpler GRU architecture is sufficient for this task, offering comparable performance with potentially greater efficiency.

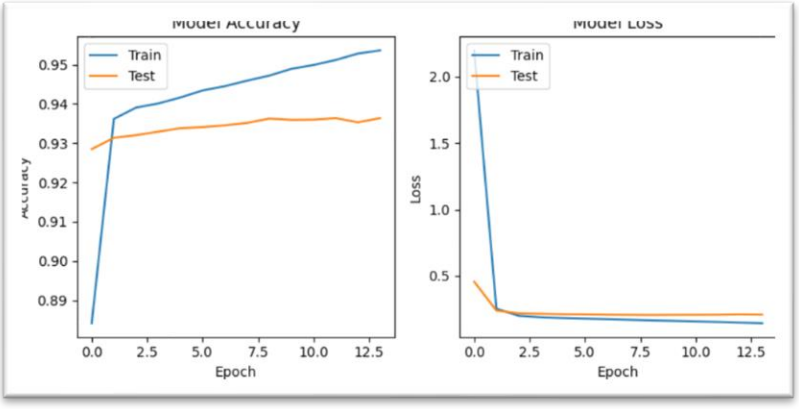

Figure 3: BiGRU Model Performance.

4.6. Experiment 5: Transformer Model (CAMeLBERT)

A state-of-the-art, pre-trained Transformer model
(CAMeL-Lab/bert-base-arabic-camelbert-mix-pos-msa)
was applied using the Hugging Face pipeline. This approach required no training and demonstrated excellent qualitative performance, correctly handling complex word segmentation and nuanced POS tags. For many applications, this would be the most practical and effective solution.

5. Visualization with NetworkX

To intuitively display the results, tagged sentences were visualized as directed graphs using `NetworkX` . Each word and its predicted tag form a node, with edges connecting the words in sequence. This representation is useful for inspecting model predictions and identifying error patterns.

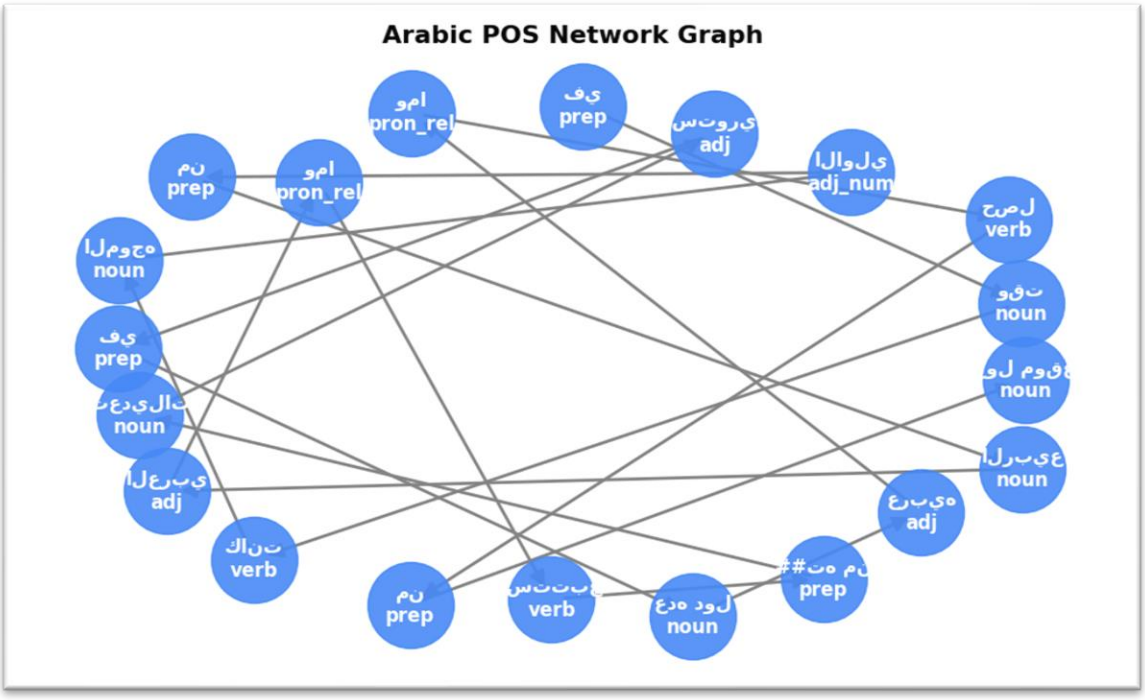

Figure 4: A NetworkX visualization of a sentence tagged

6.Results Comparison and Discussion

All deep learning models dramatically outperformed the non-contextual baseline. The hybrid **CNN + BiLSTM model achieved the highest accuracy of 94.44%**.

Model Architecture	Test Accuracy	Test Loss
Simple RNN	93.82%	0.2274
BiLSTM	94.16%	0.1770
CNN + BiLSTM (Hybrid)	**94.44%**	**0.1744**
BiGRU	94.05%	0.1852

Table 2: Comparison of test performance across different deep learning models.

7. Overall Conclusion

This project confirmed that context-aware deep learning models are vastly superior to rule-based methods for Arabic POS tagging. Bidirectional and hybrid architecture (CNN+BiLSTM) proved most effective among the trained models. The success of These models relied heavily on transfer learning, both through pre-trained words embeddings (AraVec) and the use of a fine-tuned Transformer model (CAMeLBERT), which offered state-of-the-art performance out-of-the-box

8.Project Reflection

Biggest Challenge

1. The biggest challenge was processing data for deep learning methods. This involved the complexities of converting Arabic text and their corresponding POS tags into the proper numerical format required by neural networks.

2. The second major challenge was improving model accuracy. This involved experimenting with various optimization techniques, tuning hyperparameters, and exploring different model architectures to push the performance boundaries and achieve the highest possible accuracy on the test set.

What I Learned

This project provided end-to-end experience in executing an NLP project. Key learnings include: the architectural trade-offs between RNN, LSTM, and GRU models; the critical importance of transfer learning via pre-trained embeddings and large language models like CAMeLBERT; and a deeper appreciation for the specific challenges of a morphologically rich language like Arabic.

9. Tools and Resources Used

Programming Language: Python 3

Core Libraries: TensorFlow/Keras, CAMeL Tools, Hugging Face Transformers, Gensim, Scikit-learn, NetworkX.

External Resources:

Dataset: Arabic-PUD from Universal Dependencies.

Embeddings: AraVec from the AraVec Project.

Transformer Model: CAMeLBERT from the Hugging Face Model Hub.