NLP report

(Arabic POS)

Actions and Configurations

1. **Data Preparation**: The data was downloaded and the maximum length of the segments was determined. The segments (x) and their corresponding POS tags (y) were stored for each in a separate array

2. **Tokenization**:

   Vocabulary and POS Tags: A vocabulary of unique words and a set of unique POS tags were created from the data.

   Index Mapping: Each word and POS tag was assigned a unique index. This mapping was used to convert the words in each document and their corresponding POS tags into indices, and added ‘’UNK’ for unknown words

3. **Model Architecture**: A sequence-to-sequence model with attention was built using TensorFlow and Keras. The model consists of an embedding layer, an LSTM encoder, an LSTM decoder, and an attention mechanism. The final layer is a time-distributed dense layer with softmax activation.

   Model Compilation: The model was compiled with the Adam optimizer and the sparse categorical cross-entropy loss function.

   Model Training: The model was trained for 80 epochs with a batch size of 64. 15% of the data was used for validation during training.

Results and Comparison

- **Tokenizer**: tokenizing the data manually gave better results than keras tokenizer at the training because the size of the data.
-
- **Hyperparameters**: The number of units in the LSTM layers and the dimension of the embeddings were tuned by trial and error I noticed that setting more units for LSTM and more dimensions for embeddings led to overfitting, while setting fewer units and dimensions led to underfitting.
-
- **Accuracy**: The model achieved an accuracy of 90% on the training set and 81% on the testing set, also adding more layers lead to overfitting, because again the size of the data is small,

Interference:

predict_pos_tags(words): This is a function that predicts the POS tags for a list of words.

first, it converts each word in the list to its corresponding index using a dictionary called word_to_index that contains index of each word from the tokenization part If a word is not in the dictionary, it uses the index for 'UNK' (unknown)

the model then makes predictions for each word. The output is a list of probabilities for each POS tag, The function selects the POS tag with the highest probability for each word. It does this by finding the index of the maximum value in the list of probabilities. Finally, it converts the indices back to POS tags using the **index_to_pos** dictionary and returns the list of predicted POS tags.