**Explanation Document: Model Development Process**

**1. Introduction**

This document provides an overview of the development process for the POS (Part-of-Speech) tagging and NER (Named Entity Recognition) model. It covers the data preprocessing steps, model architecture, training process, challenges faced, and rationale behind various decisions.

**2. Data Preprocessing**

*a. Loading the Dataset:*

- The dataset was loaded from a TSV file containing three columns: `word`, `pos` (Part-of-Speech tags), and `ner` (Named Entity Recognition tags).

*b. Handling Missing Values:*

- Any rows containing missing values were dropped to ensure the model only received complete data.

*c. Tokenization:*

- Each word in the dataset was tokenized using a `Tokenizer` from Keras, converting words into sequences of numerical indices based on their occurrence in the dataset.

*d. Label Encoding:*

- POS and NER tags were encoded into numerical labels using `LabelEncoder`. This step was necessary for converting categorical data into a format suitable for the model.In our model we have used post padding.

*e. Padding Sequences:*

- Sequences of varying lengths were padded to ensure uniformity in input data. This was done to maintain a consistent input shape for the model.

*f. One-Hot Encoding:*

- The encoded POS and NER tags were one-hot encoded, transforming them into a binary matrix representation to be used as the output labels during model training.

**3. Model Architecture**

*a. Embedding Layer:*

- The model uses an Embedding layer to convert input words into dense vectors of fixed size. This layer helps capture the semantic meaning of words.

*b. Bidirectional LSTM Layer:*

- A Bidirectional LSTM (Long Short-Term Memory) layer was chosen for its ability to capture dependencies from both past and future contexts. This is particularly useful for tasks like POS tagging and NER.

*c. TimeDistributed Dense Layer:*

- Two `TimeDistributed` Dense layers were used, one for POS tagging and one for NER tagging. This allows the model to predict tags for each word in a sequence independently.

*d. Output Layer:*

- The output of each `TimeDistributed` layer is passed through a softmax activation function, generating a probability distribution over possible tags.

**4. Model Training and Hyperparameters**

In our model development process, we split the dataset into training, validation, and test sets using a ratio of 70% for training, and the remaining 30% was evenly divided between validation and test sets, with 15% each. We employed the Adam optimizer for efficient training and utilized categorical cross-entropy as the loss function for both POS tagging and NER tagging tasks. To measure the model's performance, accuracy was chosen as the primary metric. The model was trained for 10 epochs with a batch size of 32, and the training process was monitored with verbose output enabled (verbose=1).

The model architecture included a 128-dimensional embedding layer for input sequences, followed by a bi-directional LSTM layer with 64 units and a recurrent dropout rate of 0.2. The final output layers for both POS and NER tagging used the softmax activation function to handle the multi-class classification problem. These hyperparameters and design choices were made to optimize the model's performance on the given dataset.

**5. Training Process**

*a. Loss Function:*

- The model was compiled with the `categorical\_crossentropy` loss function, which is well-suited for multi-class classification tasks like POS tagging and NER.

*b. Metrics:*

- Accuracy was used as the primary metric for evaluating model performance during training and validation.

*c. Validation:*

- The data was split into training, validation, and test sets. The model was trained on the training set and validated on the validation set to tune hyperparameters.

**6. Evaluation and Challenges**

*a. Model Evaluation:*

- The model was evaluated on the test set, with performance metrics including accuracy, precision, recall, and F1 score calculated for both POS tagging and NER tasks.

*b. Challenges Faced:*

- Data Imbalance: There were some challenges related to data imbalance, particularly with less frequent tags, which affected model performance. To mitigate this, strategies like data augmentation and re-sampling could be considered in future work.

- Sequence Length Variability: Handling varying sequence lengths was another challenge. Padding was used, but longer sequences could potentially benefit from more sophisticated handling, such as truncation or splitting.

**7. Conclusion**

The development of the POS and NER tagging model involved careful preprocessing, the selection of a suitable model architecture, and thoughtful evaluation of the results. Despite some challenges, the model demonstrated strong performance, making it a robust solution for tagging tasks in the Bengali language. Future work could focus on addressing the challenges mentioned above to further enhance model accuracy and generalization.