LSTM + FastAPI Creation ASSIGNMENT

Problem Statement:
This assignment aims to utilize Long Short-Term Memory (LSTM) algorithms to perform next word prediction on a given dataset. Next word prediction involves predicting the next word in a sequence of text based on the preceding words. By implementing LSTM (Long Short-Term Memory) networks, students are expected to learn how to process sequential data and capture long-term dependencies for predicting the next word in a sentence.

Guidelines:
1. Foundational Knowledge:
   - Understand the principles of Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks.
   - Recognize the advantages of LSTMs in capturing long-term dependencies in sequential data.

2. Data Exploration:
   - Analyze the dataset's structure and characteristics, paying particular attention to the sequential nature of the text data.
   - Explore preprocessing techniques for text data, such as tokenization and sequence generation.

3. Preprocessing and Feature Engineering:
   - Preprocess the textual data by converting it into sequences of fixed length suitable for input to the LSTM model.
   - Tokenize the text data and generate sequences with corresponding labels (next words).

4. LSTM Model Construction:
   - Define the architecture of the LSTM model, including the number of LSTM layers, hidden units, and embedding dimensions.
   - Compile the LSTM model with an appropriate loss function and optimizer.

5. Model Training:
   - Train the LSTM model on the training data, monitoring performance metrics such as loss.
   - Implement techniques to prevent overfitting, such as early stopping and dropout regularization.

6. Model Evaluation:
   - Evaluate the trained LSTM model on the testing data using appropriate evaluation metrics.
   - Analyze the model's performance in predicting the next word and identify any areas for improvement.

7. Fine-tuning and Optimization:
   - Fine-tune the LSTM model by adjusting hyperparameters such as learning rate and batch size.
   - Explore techniques for optimizing LSTM performance, such as gradient clipping and learning rate scheduling.

8. API Creation Using FastAPI:
   - Implement an API using FastAPI to serve the trained LSTM model.
   - Create endpoints for predicting the next word based on input text sequences.
   - Ensure the API is capable of handling requests efficiently and returning predictions promptly.

Step-by-Step Approach to LSTM Modeling:
1. Setup and Data Preparation:
   - Import necessary libraries: TensorFlow/Keras, numpy.
   - Load the dataset for next word prediction.
   - Preprocess the textual data by tokenizing and generating sequences.

2. LSTM Model Architecture:
   - Define the architecture of the LSTM model, including the number of LSTM layers and hidden units.

3. Building the LSTM:
   - Build the LSTM model using TensorFlow/Keras layers, including embedding layers.
   - Compile the model with an appropriate loss function and optimizer.

4. Model Training:
   - Train the LSTM model on the training data, specifying the number of epochs and batch size.
   - Monitor training progress and performance.

5. Model Evaluation:
   - Evaluate the trained LSTM model on the testing data using appropriate evaluation metrics.
   - Analyze the model's performance in predicting the next word.

6. Fine-tuning and Optimization:
   - Fine-tune the LSTM model by adjusting hyperparameters and exploring optimization techniques.
   - Validate the optimized model's performance and compare it with baseline results.

7. API Creation Using FastAPI:
   - Install FastAPI and Uvicorn
   - Create a FastAPI app
   - Run the FastAPI app

Link to Dataset for the Assignment:

- LSTM Next Word Prediction Dataset
  [https://www.kaggle.com/datasets/hakim11/lstm-next-word-prediction-data/data]