RNN + FastAPI Creation ASSIGNMENT

Problem Statement:
This assignment aims to utilize Recurrent Neural Network (RNN) algorithms to perform sentiment analysis on a given dataset. Sentiment analysis involves determining the sentiment (positive, negative, or neutral) expressed in textual data. By implementing RNN models, students are expected to learn how to process sequential data and capture contextual information for sentiment classification.

Guidelines:
1. Foundational Knowledge:
   - Understand the principles of Recurrent Neural Networks (RNNs) and their suitability for sequential data processing tasks.
   - Familiarize yourself with different RNN architectures such as vanilla RNNs, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU).
   - Recognize the advantages of RNNs in capturing temporal dependencies in sequential data.

2. Data Exploration:
   - Analyze the dataset's structure and characteristics, paying particular attention to the textual data and sentiment labels.
   - Explore preprocessing techniques for text data, such as tokenization, padding, and vectorization.

3. Preprocessing and Feature Engineering:
   - Preprocess the textual data by converting it into a suitable format for input to the RNN model.
   - Encode sentiment labels into numerical values if necessary.
   - Split the dataset into training and testing sets, ensuring a balanced distribution of sentiment classes.

4. RNN Model Construction:
   - Choose an appropriate RNN architecture based on the nature of the sentiment analysis task and dataset size.
   - Define the architecture of the RNN model, including the number of layers, hidden units, and activation functions.
   - Compile the RNN model with an appropriate loss function and optimizer.

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

6. Model Evaluation:
   - Evaluate the trained RNN model on the testing data using evaluation metrics such as accuracy, precision, recall, and F1-score.
   - Analyze the model's performance across different sentiment classes and identify any biases or shortcomings.

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

8. API Creation Using FastAPI:
   - Implement an API using FastAPI to expose the sentiment analysis model as a service.
   - Define endpoints for prediction and health checks.
   - Ensure the API handles input validation and returns appropriate responses.

Step-by-Step Approach to RNN Modeling:

1. Setup and Data Preparation:
   - Import necessary libraries: TensorFlow/Keras, numpy, pandas, FastAPI.
   - Load the dataset for sentiment analysis.
   - Preprocess the textual data and encode sentiment labels.

2. RNN Model Architecture:
   - Choose an appropriate RNN architecture based on the dataset characteristics.
   - Define the architecture of the RNN model, including the number of layers and hidden units.

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

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

5. Model Evaluation:
   - Evaluate the trained RNN model on the testing data using appropriate evaluation metrics.
   - Analyze the model's performance and identify areas for improvement.

6. Fine-tuning and Optimization:
   - Fine-tune the RNN 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:
   - Set up a FastAPI application.
   - Create an endpoint for receiving text input and returning the sentiment prediction.
   - Create an endpoint for health checks to ensure the API is running smoothly.
   - Test the API locally to ensure it correctly processes input and returns predictions.

Link to Dataset for the Assignment:
- Twitter Sentiment Analysis Dataset
[https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis/data]