Status Prediction System

Introduction

The Status Prediction System is a machine learning-based application designed to predict internal status labels based on external status descriptions. This documentation provides a comprehensive overview of the project, including data preprocessing, model development, API implementation, and testing results.

Preprocessing

Cleaning and Formatting Dataset

- The dataset was loaded from the provided JSON file.

- No specific cleaning was required as the data was already in a structured format.

- External status descriptions were tokenized and padded for model input.

- Internal status labels were encoded into numerical format using LabelEncoder.

Model Architecture

Chosen Model Architecture

- The chosen model architecture is based on a Long Short-Term Memory (LSTM) neural network.

- The model consists of an embedding layer followed by an LSTM layer and dense layers for classification.

- An embedding layer was used to represent words as dense vectors.

- The LSTM layer captured the sequential nature of the input data.

Training Procedure

Model Training Process

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

- Hyperparameters such as batch size, number of epochs, and embedding dimension were tuned through experimentation.

- Training was conducted on the preprocessed dataset, with validation performed on a separate test set.

- Model training was monitored for metrics such as accuracy and loss to assess performance.

API Implementation

API Usage Documentation

- The API provides a single endpoint `/predict` for making predictions.

- Input to the API endpoint should be a JSON object containing the external status description.

- The API returns a JSON response containing the predicted internal status label.

Testing Results

Testing and Validation Summary

- The API functionality was thoroughly tested using sample inputs and edge cases.

- Predictions were validated against a validation dataset to measure model generalization.

- Performance metrics such as accuracy, precision, recall, and F1-score were calculated.

- The model demonstrated robust performance with high accuracy on both training and validation datasets.

Deployment Instructions

Deployment Instructions

- Deploying the API requires installing the necessary dependencies and configuring the environment.

- Ensure the trained model (`LSTM_model.pkl`) is available in the deployment environment.

- Run the FastAPI application using a suitable web server (e.g., uvicorn).

- Access the API endpoint at the specified URL for making predictions.

* * *

Name : Goutham K G

E-mail : gouthamktm707@gmail.com