NEXT WORD PREDICTION USING LSTM

# Abstract:

The project implements a next-word prediction model using LSTM in Keras. It reads and preprocesses a user-specified dataset, trains an LSTM model, and generates sequences based on user input. The script includes functions for dynamic sequence generation and visualizing training history. Recommendations focus on refining preprocessing, optimizing model parameters, and enhancing user input handling.

# Problem Statement:

Develop an efficient next-word prediction model using LSTM (Long Short-Term Memory) networks in Keras. The goal is to accurately predict the succeeding word in a given text sequence. The project involves reading and preprocessing a user-provided dataset, training an LSTM model with appropriate hyperparameters, and implementing a user-friendly

interface for dynamic sequence generation. The challenge lies in optimizing model

performance, handling diverse input scenarios, and ensuring effective training visualization. Additionally, explore methods to enhance model interpretability and scalability for potential real-world applications.

# Methodology:

## Dataset Acquisition and Exploration:

* Obtain a diverse text dataset for training the next-word prediction model.
* Explore the dataset to understand its characteristics, ensuring it represents the language diversity and context relevant to the application.

## Data Preprocessing:

* Clean the dataset by removing unnecessary characters, handling punctuation, and converting text to lowercase.
* Tokenize the text into words or subwords to create a vocabulary.
* Split the dataset into training and validation sets for model evaluation.

## Model Design and Configuration:

* Construct a sequential model using Keras with an Embedding layer for word

representation, an LSTM layer for sequence learning, and a Dense layer for prediction.

* Experiment with hyperparameters such as embedding dimensions, LSTM units, and vocabulary size to optimize model performance.
* Implement dropout layers to mitigate overfitting.

## Model Training:

* Train the model using the preprocessed dataset, monitoring training and validation loss.
* Utilize early stopping to prevent overfitting and save the best-performing model.

## Sequence Generation:

* Develop a function for generating sequences based on user input.
* Allow users to specify the number of words to generate dynamically.
* Ensure the model handles various input scenarios gracefully.

## Visualization:

* Implement functions to visualize the training history, including loss and accuracy trends over epochs.
* Utilize Matplotlib for creating clear and informative plots.

## Hyperparameter Tuning and Optimization:

* Experiment with different configurations, including embedding dimensions, LSTM units, and learning rates, to fine-tune model performance.
* Implement grid search or random search for hyperparameter optimization.

## Documentation and Code Refinement:

* Add comprehensive comments and documentation to enhance code readability and understanding.
* Refine the code based on feedback and insights from experimentation.

## Evaluation and Testing:

* Evaluate the model on the validation set to assess its predictive performance.
* Test the model with user input to ensure effective and dynamic sequence generation.

## Deployment and Future Work:

* Explore opportunities for deploying the model in real-world applications.
* Consider future improvements, such as incorporating attention mechanisms or transformer architectures for enhanced performance and scalability.

**Outputs:**





