Full Documentation

Introduction

This project implements a **Sinhala Spell Checker** using **Artificial Intelligence** (AI). The model leverages a **Long Short-Term Memory (LSTM)** neural network to identify and auto-correct misspelled Sinhala words. The dataset contains over 100,000 Sinhala words with labels indicating correctness.

Model Architecture

- **Input**: A sequence of characters representing a word.
- **Embedding Layer**: Transforms each character into a dense vector representation.
- **LSTM Layer**: Captures sequential patterns in characters to classify words.
- **Dense Layers**: Fine-tune the model for binary classification.
- **Output**: Probabilities for the word being correct or incorrect.

Importance of AI in Spell Checking

1. **Context-Aware Suggestions**:
    - Unlike traditional methods, AI models consider the context of characters in a word, improving accuracy.
2. **Automation**:
    - The system automatically corrects words, reducing manual intervention in tasks like document processing or text editing.
3. **Multilingual Support**:
    - AI models like this can be adapted for other languages, benefiting users in non-English languages like Sinhala.

Code Explanation

1. Loading and Preprocessing the Data

- **Loading the dataset**: The data is loaded from an Excel file (`data-spell-checker.xlsx`) using the `pandas` library. It has two columns:
    - `word`: The Sinhala word.
    - `label`: 1 for correct words and 0 for incorrect words.
- **Data Cleaning**:
    - Any missing values (`NaN`) are dropped using `dropna()`.
- **Tokenization**:
    - The `Tokenizer` processes each word at the **character level** to convert Sinhala text into sequences of integers. This approach helps the model analyze patterns in character combinations.
- **Padding Sequences**:
    - Since words have varying lengths, shorter sequences are padded with zeros to ensure all sequences have the same length (`max_len`).
- **Labels**:
    - The `label` column (correct/incorrect) is converted into one-hot encoded vectors (`to_categorical`), making it suitable for training a classification model.

- **Data Splitting**:
  - The dataset is split into training and validation sets using an 80/20 split for model evaluation.

---

2. Model Definition

- **Embedding Layer**:
  - Converts character-level integer sequences into dense vector representations of size 50. This helps the model understand semantic relationships between characters.
- **LSTM Layer**:
  - A Long Short-Term Memory (LSTM) layer captures dependencies between characters, enabling the model to understand word patterns better.
- **Dropout Layer**:
  - Prevents overfitting by randomly deactivating 30% of the neurons during training.
- **Dense Layers**:
  - The first dense layer with 64 units learns higher-level patterns.
  - The final dense layer with 2 output neurons uses the `softmax` activation for binary classification (0 = Incorrect, 1 = Correct).

---

3. Training the Model

- **Compilation**:
  - The model is compiled with the **Adam optimizer**, **categorical cross-entropy loss**, and **accuracy metric** for performance evaluation.
- **Training**:
  - The model is trained for 10 epochs with a batch size of 32.
  - Final results:
    - **Accuracy**: 95.46%
    - **Loss**: 0.1199
  - This high accuracy suggests that the model correctly predicts whether a word is correct or incorrect most of the time.

---

4. Word Prediction and Correction

- **Predicting Correctness**:
  - Each word is converted into a padded sequence and passed through the trained model to predict its class (correct/incorrect).
- **Auto-Correction**:
  - If the word is incorrect, the **`difflib.get_close_matches()`** function suggests a correction based on similar words in the list of correct words (`correct_words`).
- **Sentence Processing**:
  - Each word in a given sentence is checked for correctness and corrected if necessary.
  - The output includes:
    - Original Sentence
    - Misspelled Words
    - Corrected Sentence

Applications

1. **Education**:
    - Helps students and teachers in typing error-free Sinhala documents.
2. **Text Processing**:
    - Useful for government offices, publishing houses, and online tools for text validation.
3. **Language Learning**:
    - Assists non-native speakers in learning Sinhala by identifying and correcting spelling errors.

How High Accuracy Helps

- The model's **95.46% accuracy** means it identifies nearly all misspelled words, minimizing errors in suggestions.
- This ensures reliable performance in practical applications like generating error-free documents.

Conclusion

This AI-powered Sinhala spell checker combines the power of LSTM and tokenization to identify and correct errors effectively. With its high accuracy, it is a valuable tool for improving the quality of Sinhala text in various domains.