EVAL_BERT
This program is responsible for evaluating a BERT-based model's ability to perform text segmentation. It loads a pre-trained BERT model for token classification and uses a custom character-level tokenizer to process input text. The segmentation is evaluated based on precision, recall, and F1-score by comparing the predicted output with ground truth data. The program also calculates metrics at the domain level and saves evaluation results as JSON files and charts. Additionally, it applies Levenshtein distance to handle subword matching, ensuring that minor spelling variations do not significantly affect performance.

EVAL_BERT_2
This program functions similarly to **EVAL_BERT**, but it uses a different model version for segmentation evaluation. It follows the same process of text tokenization, segmentation inference, and evaluation but operates with a model named "bert-segmentation_2." Like its predecessor, it calculates segmentation accuracy using exact and fuzzy matches, generates evaluation charts, and saves performance metrics as JSON files. The main difference lies in the model being tested, allowing comparison between different BERT-based segmentation models.

EVAL_BETWEEN_MODELS
This program performs comparative analysis between different text segmentation models. It gathers evaluation metrics from multiple models, such as **BERT** and **WordNinja**, to compare their segmentation performance across different domains. It computes statistical measures, including the mean, standard deviation, minimum, and maximum values for precision, recall, and F1-score. Additionally, it generates visual charts showing the differences between models and saves the results in structured JSON files. This enables users to determine which model performs best in specific conditions and track deviations from the average performance.

EVAL_WORDNINJA
This program evaluates the segmentation performance of the WordNinja library, a statistical-based text segmentation tool. It processes concatenated strings by splitting them into meaningful words using WordNinja, followed by filtering non-English terms with PyEnchant. The evaluation is performed by comparing predicted segments against ground truth annotations. The program calculates precision, recall, and F1-score while considering exact matches and subword matches using Levenshtein distance. It also generates reports and charts to visualize segmentation accuracy across different text categories.

LEVENSHTEIN_INFERENCE
This is a simple interactive program that checks whether a given word is a subword of another using substring matching and Levenshtein distance. It prompts the user for input and determines if the subword is within a specified distance from the main word, making it useful for applications requiring approximate word matching. The program is lightweight and designed for quick subword validation in real-time user inputs.

WORDNINJA_INFERENCE
This program performs segmentation of concatenated text using the WordNinja library. It integrates fuzzy similarity matching with RapidFuzz to account for approximate word matches. The segmentation accuracy is evaluated using both exact and fuzzy match

techniques, allowing for partial matches when a high similarity score is detected. The program processes sample files and generates evaluation metrics for performance assessment. Additionally, it allows users to set a similarity threshold, making it flexible for different levels of segmentation precision.

BERT_INFERENCE

This program provides an interactive interface for text segmentation using a BERT-based model. It loads a pre-trained BERT model and a custom character-level tokenizer to process input text. The program allows users to input concatenated text and predicts segmentation labels, ultimately splitting the text into individual words. It then prints the original text, predicted labels, and segmented words, making it a useful tool for real-time testing of the model’s segmentation capabilities.