Lab 3

**Spelling Checker Application using Minimum Edit Distance Algorithm**

1. Write a python method which compute Levenshtein minimum edit distance algorithm, the method should take two words and return its minimum distance

def levenshtein\_distance(word1, word2):  
 m, n = len(word1), len(word2)  
  
 *# Create a 2D matrix to store the distances* dp = [[0] \* (n + 1) for \_ in range(m + 1)]  
  
 *# Initialize the matrix with the base cases* for i in range(m + 1):  
 dp[i][0] = i  
 for j in range(n + 1):  
 dp[0][j] = j  
  
 *# Populate the matrix using dynamic programming* for i in range(1, m + 1):  
 for j in range(1, n + 1):  
 cost = 0 if word1[i - 1] == word2[j - 1] else 2  
  
 dp[i][j] = min(dp[i - 1][j] + 1, *# Deletion* dp[i][j - 1] + 1, *# Insertion* dp[i - 1][j - 1] + cost) *# Substitution  
  
 # The bottom-right cell contains the minimum edit distance* return dp[m][n]  
  
*# Example usage:*word1 = "intention"  
word2 = "execution"  
result = levenshtein\_distance(word1, word2)  
print(f"The Levenshtein distance between '{word1}' and '{word2}' is: {result}")

output:

The Levenshtein distance between 'intention' and 'execution' is: 8

1. Write a python function to create a dictionary of English vocabulary using any corpus from Project Gutenberg (a **library of over 60,000 free eBooks**). This dictionary will be used to correct mis-spelled words

import string  
from collections import Counter  
from nltk.tokenize import word\_tokenize  
from nltk.corpus import gutenberg  
from nltk.corpus import stopwords  
  
def create\_vocabulary\_dict(corpus\_name):  
 *# Download the NLTK resources if not already downloaded* import nltk  
 nltk.download('punkt')  
 nltk.download('stopwords')  
  
 *# Get the list of words from the specified Gutenberg corpus* raw\_text = gutenberg.raw(corpus\_name)  
  
 *# Tokenize the text into words* words = word\_tokenize(raw\_text)  
  
 *# Remove punctuation and convert to lowercase* words = [word.lower() for word in words if word.isalpha()]  
  
 *# Remove stop words* stop\_words = set(stopwords.words('english'))  
 words = [word for word in words if word not in stop\_words]  
  
 *# Count the frequency of each word* word\_counts = Counter(words)  
  
 *# Create a dictionary with words as keys and their frequencies as values* vocabulary\_dict = dict(word\_counts)  
  
 return vocabulary\_dict  
  
*# Example usage:*corpus\_name = 'bryant-stories.txt' *# Change this to the desired corpus*vocabulary\_dict = create\_vocabulary\_dict(corpus\_name)  
  
*# Print a sample of the vocabulary dictionary*print("Sample Vocabulary Dictionary:")  
for word, frequency in list(vocabulary\_dict.items())[:10]:  
 print(f"{word}: {frequency}")

output:

Sample Vocabulary Dictionary:

stories: 14

tell: 28

children: 39

sara: 1

cone: 1

bryant: 1

two: 63

little: 596

riddles: 1

rhyme: 1

1. Write a spelling checker method which take a mis-spelled word and dictionary, and return a list of five-most likely corrected words.

def spelling\_checker(misspelled\_word, vocabulary\_dict, distance\_function):  
 # Calculate Levenshtein distance between the misspelled word and all words in the dictionary  
 distances = [(word, distance\_function(misspelled\_word, word)) for word in vocabulary\_dict]  
  
 # Filter out words that are not valid English words  
 valid\_words = [word for word, \_ in distances if word in vocabulary\_dict]  
  
 # Sort the valid words based on their Levenshtein distance (lower is better)  
 sorted\_distances = sorted(distances, key=lambda x: x[1])  
  
 # Extract the five most likely corrected words among the valid words  
 top\_five\_corrections = [word for word, \_ in sorted\_distances if word in valid\_words][:5]  
  
 return top\_five\_corrections  
  
  
# Example usage:  
corpus\_name = 'shakespeare-caesar.txt'  
vocabulary\_dict = create\_vocabulary\_dict(corpus\_name)  
  
misspelled\_word = 'entr'  
corrections = spelling\_checker(misspelled\_word, vocabulary\_dict, levenshtein\_distance)  
print(f"Top five corrections for '{misspelled\_word}': {corrections}")

output:

Top five corrections for 'entr': ['enter', 'went', 'entire', 'bent', 'et']

1. Write a simple test drive python code to test the above methods

print(f"Vocabulary Dictionary created from '{corpus\_name}':")  
print("Sample entries:", list(vocabulary\_dict.items())[:10])  
  
print("\nSpelling Checker:")  
print(f"Top five corrections for '{misspelled\_word}': {corrections}")

**output:**

**Vocabulary Dictionary created from 'shakespeare-caesar.txt':**

**Sample entries: [('tragedie', 2), ('julius', 1), ('caesar', 190), ('william', 1), ('shakespeare', 1), ('actus', 5), ('primus', 1), ('scoena', 1), ('prima', 1), ('enter', 64)]**

**Spelling Checker:**

**Top five corrections for 'entr': ['enter', 'went', 'entire', 'bent', 'et']**