DSA Project Documentation

Autocorrect feature

Objective:

The primary objective is to develop a terminal-based application that suggests correct spellings for input words. It involves the use of trie for efficient word storage and retrieval, and the Damerau- Levenshtein distance algorithm for calculating the closest spelling suggestions. The application will also allow users to manage the underlying dictionary.

Program:

```
class TrieNode:
    def __init__(self):
        self.children={}
        self.is_end_of_word=False
class Trie:
    def __init__(self):
        self.root=TrieNode()
    def insert(self,word):
        node=self.root
        for char in word:
            if char not in node.children:
                node.children[char]=TrieNode()
            node = node.children[char]
        node.is_end_of_word=True
    def search(self,word):
        node=self.root
        for char in word:
            if char not in node.children:
                return False
            node=node.children[char]
        return node.is_end_of_word
def edit_distance(str1, str2):
    m,n=len(str1),len(str2)
    dp=[[0]*(n+1) for _ in range(m+1)]
    for i in range(m+1):
        for j in range(n+1):
            if i==0:
                dp[i][j]=j
            elif j==0:
                dp[i][j]=i
            elif str1[i-1]==str2[j-1]:
                dp[i][j]=dp[i-1][j-1]
            else:
                dp[i][j]=1+min(dp[i-1][j],dp[i][j-1],dp[i-1][j-1])
    return dp[m][n]
def suggest_spellings(trie,word):
    suggestions=[]
    for suggestion in trie_suggestions(trie,word):
        suggestions.append(suggestion)
    return suggestions
def trie_suggestions(trie,prefix):
    node=trie.root
    for char in prefix:
        if char not in node.children:
            return
        node=node.children[char]
    stack=[(node, prefix)]
    while stack:
        current_node,current_prefix=stack.pop()
        if current_node.is_end_of_word:
            yield current_prefix
        for char,child_node in current_node.children.items():
            stack.append((child_node,current_prefix+char))
def main():
    dictionary_file=r"C:\Users\JAISHNI 
ANANTHA\OneDrive\Desktop\DSA(Project3)\dictionary.txt"
    trie=Trie()
    with open(dictionary_file,"r") as file:
        for line in file:
            word=line.strip().lower()
            trie.insert(word)
    user_input=input("Enter a potentially misspelled word: ").lower()
    if trie.search(user_input):
        print(f"{user_input} is a valid word.")
    else:
        suggestions=suggest_spellings(trie,user_input)
        if suggestions:
            print(f"Did you mean: {suggestions}?")
        else:
            print("No suggestions found.")
if __name__ == "__main__":
    main()
```

Output:

```
Enter a potentially misspelled word: swapa

No suggestions found.


Enter a potentially misspelled word: sw

Did you mean: ['swa', 'swap']?


Enter a potentially misspelled word: goa

Did you mean: ['goal', 'goat']?


Enter a potentially misspelled word: go

Did you mean: ['gosht', 'gossip', 'goal', 'goat']?


Enter a potentially misspelled word: dis

Did you mean: ['dispense', 'disturb', 'distract', 'district']?


Enter a potentially misspelled word: dist

Did you mean: ['disturb', 'distract', 'district']?


Enter a potentially misspelled word: distri

Did you mean: ['district']?


Enter a potentially misspelled word: district

district is a valid word.
```