Spell Checker with C++ and Python Testing

Introduction

This document presents a spell checker program written in C++ and a Python testing script to verify its functionality. The code uses a dictionary of common words to check if an input word is correctly spelled or if it has a close suggestion. The program identifies potential spelling errors and suggests corrections based on the Levenshtein distance algorithm.

main.cpp

The is the C++ code that implements the spell checker. The program uses Levenshtein distance to find the closest matching word in the dictionary when the input word is not spelled correctly.

Listing 1: main.cpp

```cpp
#include <iostream>
#include <unordered_set>
#include <vector>
#include <string>
#include <algorithm>
#include <limits>

// SpellChecker Class Definition
class SpellChecker {
public:
    SpellChecker(const std::vector<std::string>& dictionary);
    std::vector<std::string> spellCheck(const std::string& word);

private:
    std::unordered_set<std::string> wordSet;
    int calculateLevenshteinDistance(const std::string& source, const std::string&
        target);
    std::string findClosestMatch(const std::string& word);
};

// SpellChecker Implementation
SpellChecker::SpellChecker(const std::vector<std::string>& dictionary) {
    for (const auto& word : dictionary) {
        wordSet.insert(word);
    }
}

std::vector<std::string> SpellChecker::spellCheck(const std::string& word) {
    if (wordSet.find(word) != wordSet.end()) {
        return {word};  // Word is correctly spelled
    }

    std::string closestMatch = findClosestMatch(word);
    return closestMatch.empty() ? std::vector<std::string>{} : std::vector<std::
        string>{closestMatch};
}

int SpellChecker::calculateLevenshteinDistance(const std::string& source, const std
    ::string& target) {
    std::vector<std::vector<int>> dp(source.size() + 1, std::vector<int>(target.size
        () + 1));

    for (size_t i = 0; i <= source.size(); ++i) dp[i][0] = i;
    for (size_t j = 0; j <= target.size(); ++j) dp[0][j] = j;

```

```
        for (size_t i = 1; i <= source.size(); ++i) {
            for (size_t j = 1; j <= target.size(); ++j) {
                int cost = (source[i - 1] == target[j - 1]) ? 0 : 1;
                dp[i][j] = std::min({dp[i - 1][j] + 1, dp[i][j - 1] + 1, dp[i - 1][j - 1] + cost});
            }
        }
        return dp[source.size()][target.size()];
    }

    std::string SpellChecker::findClosestMatch(const std::string& word) {
        int minDistance = std::numeric_limits<int>::max();
        std::string closestMatch;

        for (const auto& dictWord : wordSet) {
            int distance = calculateLevenshteinDistance(word, dictWord);
            if (distance < minDistance) {
                minDistance = distance;
                closestMatch = dictWord;
            }
        }

        // Return suggestion only if it's reasonably close
        return (minDistance <= word.size() / 2 + 1) ? closestMatch : "";
    }

    // Main Application
    int main() {
        std::vector<std::string> dictionary = {"apple", "banana", "orange", "grape", "peach", "pear", "mango", "melon"};
        SpellChecker checker(dictionary);

        std::string input;
        while (true) {
            std::cout << "Enter a word to spell-check (or 'exit' to quit): ";
            std::cin >> input;
            if (input == "exit") break;

            auto suggestions = checker.spellCheck(input);
            if (suggestions.empty()) {
                std::cout << "No suitable suggestion found.\n";
            } else if (suggestions[0] == input) {
                std::cout << "The word '" << input << "' is spelled correctly.\n";
            } else {
                std::cout << "Did you mean: " << suggestions[0] << "?\n";
            }
        }

        return 0;
    }
```

Explanation

The `SpellChecker` class in `main.cpp` checks if a word is in the dictionary. If the word is wrong (misspelled), the program calculates the Levenshtein distance between the input and each dictionary word to find the closest match. The script `test.py` verifies the program's accuracy by running multiple test cases and comparing results.

test.py

The Python script used to test the functionality of the spell checker program. It uses subprocesses to execute the compiled C++ program and compares the actual output with expected results.

Listing 2: test.py

```
import subprocess

def run_spellchecker_test(word, expected_output):
```

```
    process = subprocess.Popen(["./spellchecker.exe"], stdin=subprocess.PIPE, stdout=
        subprocess.PIPE, text=True)
    out, _ = process.communicate(input=f"{word}\nexit\n")
    actual_output = out.splitlines()[0]  # Grabbing the result line
    result = "PASS" if actual_output.strip() == expected_output.strip() else "FAIL"
    print(f"Test: {word}\nExpected: {expected_output}\nActual: {actual_output}\nResult:
        {result}\n")

# Define test cases
tests = [
    ("apple", "Enter a word to spell-check (or 'exit' to quit): The word 'apple' is
        spelled correctly."),
    ("applle", "Enter a word to spell-check (or 'exit' to quit): Did you mean: apple?"),
    ("bananna", "Enter a word to spell-check (or 'exit' to quit): Did you mean: banana?"
        ),
    ("orrange", "Enter a word to spell-check (or 'exit' to quit): Did you mean: orange?"
        ),
    ("grap", "Enter a word to spell-check (or 'exit' to quit): Did you mean: grape?"),
    ("pech", "Enter a word to spell-check (or 'exit' to quit): Did you mean: peach?"),
    ("a", "Enter a word to spell-check (or 'exit' to quit): No suitable suggestion found
        ."),
    ("ra", "Enter a word to spell-check (or 'exit' to quit): No suitable suggestion
        found."),
    ("ora", "Enter a word to spell-check (or 'exit' to quit): Did you mean: orange?"),
]

# Run all tests
for word, expected in tests:
    run_spellchecker_test(word, expected)
```

Test Results

The following are the test results obtained by running `test.py`:

```
Test: apple
Expected: The word 'apple' is spelled correctly.
Actual: The word 'apple' is spelled correctly.
Result: PASS

Test: applle
Expected: Did you mean: apple?
Actual: Did you mean: apple?
Result: PASS

Test: bananna
Expected: Did you mean: banana?
Actual: Did you mean: banana?
Result: PASS

Test: orrange
Expected: Did you mean: orange?
Actual: Did you mean: orange?
Result: PASS

Test: grap
Expected: Did you mean: grape?
Actual: Did you mean: grape?
Result: PASS

Test: pech
Expected: Did you mean: peach?
Actual: Did you mean: peach?
Result: PASS
```

```
Test: a
Expected: No suitable suggestion found.
Actual: No suitable suggestion found.
Result: PASS

Test: ra
Expected: No suitable suggestion found.
Actual: No suitable suggestion found.
Result: PASS

Test: ora
Expected: Did you mean: orange?
Actual: No suitable suggestion found.
Result: FAIL
```

Conclusion

The spell checker program works as expected for most test cases, suggesting the correct words or indicating correct spelling. However, it fails for the word "ora," which could be due to limitations in the Levenshtein distance threshold or the dictionary's contents. Further tuning or dictionary expansion may improve accuracy for edge cases.