Project Documentation

1. Introduction

This project focuses on implementing Autocomplete, SpellCheck, and Correction suggestions for misspelled words using the Trie data structure.
We assume that our system is used by one user, and is used multiple times (e.g a keyboard that the user uses on a daily basis).
For each word, we should keep track of the frequency or the number of times the user has typed it.

2. Implementation

Distance definition:

In the context of this project, the distance between two words is a simple comparison of the characters at corresponding positions (indexes) in two strings. This distance metric is commonly referred to as the character-by-character distance or position-based distance.

2.1 Inserting Words into the Trie

Before implementing the functionalities, build the Trie using the words.txt file provided. Read each word from the dictionary and Insert it into the Trie.

2.2 Functionality Implementation

1. Autocomplete

The autocomplete functionality generates word suggestions for a given input.
In implementing you should:

1. Find all words that can be a correct autocompleted version of the given word
2. Prioritize the suggestions based on their frequency. (words with higher frequency have higher priorities)
3. Display the top 5 results to the user. If multiple words have the same frequency, display any of them.

2. Spell Check

This can easily be achieved by checking whether the word exists in the trie.

3. Correction Suggestions for Misspelled Words

Providing correction suggestions for misspelled words can be achieved using various approaches. Here we will implement a simple approach to accomplish this functionality.

The implemented approach follows these steps:

1. Identify the longest correct prefix for the misspelled word.
2. Retrieve all words in the Trie that start with the longest correct prefix.
3. Reverse the misspelled word and build a Trie from the reversed words.
4. Identify the longest correct suffix for the reversed misspelled word. You can implement this by keeping a trie of reversed words
5. Retrieve all reversed words in the Trie that start with the longest correct suffix.
6. Calculate the distance between the misspelled word and each suggestion.
7. Output the top 5 suggestions based on the lowest distance to the user.

- if the longest correct prefix/suffix did not generate enough suggestions, consider using the second longest prefix/suffix as well. Implementing this feature is not required but, has bonus points.

3. Notes

a. You should develop a interface for the user to interact with. This could be a simple TUI. Implementing a GUI will have bounus points.
b. You can complete this project in groups of at most two
c. You should develop your program in Python, C++, or Java.
d. This project accounts for 1 point in your final grade.
    i. Autocomplete => 30%
    ii. Spell Check => 10%
    iii. Correction Suggestion => 60%
e. Feel free to ask questions

Good Luck 🌱