University of Windsor

Assignment 3

Aleena Ali Azeem

110190830

COMP 8547 Advanced Computing Concepts – Winter 2025

Assignment 3 Report

Task 2: Word Completion Using Tries

Objective: Implement a word completion feature using a Trie.

Implementation:

▪ Load the vocabulary into a Trie.

▪ Find all words in the Trie that start with the given prefix.

Output: A list of words that start with the given prefix.

Introduction:

In this assignment, I implemented a **word completion feature** using a **Trie (Prefix Tree)**. The goal was to load a vocabulary into a Trie and find all words that start with a given prefix. This feature is similar to predictive text suggestions used in search engines and text editors.

For this implementation, I used **Java** and read the vocabulary from a provided dataset (`dropbox_words.txt`), which contains words extracted from a CSV file.

Methodology:

Step 1: Extract Vocabulary from Dropbox Dataset

- The given dataset (`dropbox_data.csv`) contains various headings related to Dropbox.
- Extracted words from these headings.
- Filtered unique words and stored them in `dropbox_words.txt`.

Step 2: Implement Trie Data Structure

- Created a **TrieNode class** to represent each letter node.
- Implemented the **Trie class** with methods to:
  - Insert words.
  - Search for words that start with a prefix.
  - Retrieve and display matching words.

Step 3: Read Words from File and Populate Trie

- Read words from `dropbox_words.txt` and added them to the Trie.

Step 4: Accept User Input and Perform Word Completion

- The program takes a **prefix input** from the user.
- Retrieves and displays all words that start with that prefix.

Code:

```
package Assignment;
import java.io.*;
import java.util.*;
class Node {
    Map<Character, Node> childNodes; // Store child nodes (characters)
    boolean isWordComplete; // Flag to indicate end of a word

    public Node() {
        childNodes = new HashMap<>();
        isWordComplete = false;
    }
}

class PrefixTree {
    private Node rootNode;

    public PrefixTree() {
        rootNode = new Node();
    }

    // Adds a word to the Trie
    public void addWord(String word) {
        Node current = rootNode;
        for (char letter : word.toCharArray()) {
            // If letter is not present, create a new node
            current.childNodes.putIfAbsent(letter, new Node());
            // Move to the next node
            current = current.childNodes.get(letter);
        }
        // Mark the end of the word
        current.isWordComplete = true;
    }

    // Helper function to collect words with a given starting sequence
    private void gatherWords(Node node, String prefix, List<String> suggestions) {
        if (node.isWordComplete) {
            suggestions.add(prefix);
        }
        for (char letter : node.childNodes.keySet()) {
            gatherWords(node.childNodes.get(letter), prefix + letter, suggestions);
        }
    }

    // Finds words that start with the provided prefix
    public List<String> getSuggestions(String prefix) {
        Node current = rootNode;
        for (char letter : prefix.toCharArray()) {
            if (!current.childNodes.containsKey(letter)) {
                return new ArrayList<>(); // No matching words found
            }
            current = current.childNodes.get(letter);
```

```
        }
        List<String> matchingWords = new ArrayList<>();
        gatherWords(current, prefix, matchingWords);
        return matchingWords;
    }
}

public class TrieWord {
    public static void main(String[] args) {
        PrefixTree autoCompleteSystem = new PrefixTree();

        // Reading words from file (make sure dropbox_words.txt is in the same
directory)
        String fileName = "dropbox_words.txt";
        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
            String word;
            while ((word = reader.readLine()) != null) {
                autoCompleteSystem.addWord(word.trim());
            }
        } catch (IOException e) {
            System.err.println("Oops! Something went wrong while reading the file: "
+ e.getMessage());
            return;
        }

        // Taking user input for word prediction
        Scanner inputScanner = new Scanner(System.in);
        System.out.print("Start typing a word: ");
        String userPrefix = inputScanner.nextLine();

        // Fetching words that start with the given prefix
        List<String> wordSuggestions = autoCompleteSystem.getSuggestions(userPrefix);

        // Display the results
        if (wordSuggestions.isEmpty()) {
            System.out.println("Sorry, no words found starting with: " + userPrefix);
        } else {
            System.out.println("Words that start with '" + userPrefix + "': " +
wordSuggestions);
        }

        inputScanner.close();
    }
}
```

Output:

```
Start typing a word: cl
Words that start with 'cl': [cloud]
```

Figure 1: Shows how when we input cl it gives us cloud.

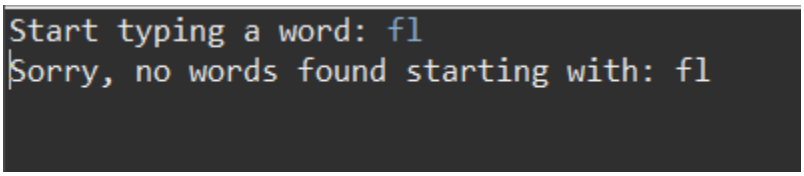

Figure 2: Shows how when we input ‘fl’ it shows that there are no words starting with that.

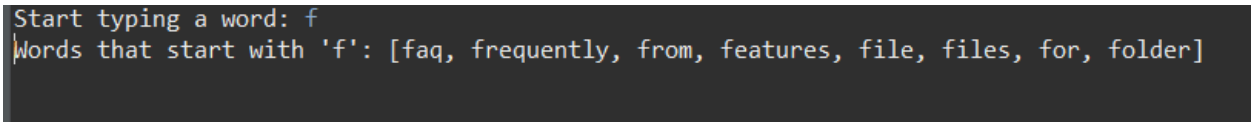

Figure 3: Shows how when we input f we are given a range of words starting with ‘f’.