CS 201 - Final Project Report: AutoComplete Editor

Syeda Saleha Raza (L2)
Muhammad Hassan (mh08062)
Muhammad Qasim Khan (mk05539)
Muhammad Saad (ms08063)
Syed Muhammad Ali Imran (si07886)

Introduction

Our project, the AutoComplete Editor, is a personalized text editor that provides auto-completion suggestions to users as they type. This feature enhances typing speed and accuracy by predicting and displaying word suggestions based on the user's input history. The project combines a NextJS frontend with a C++ backend to deliver a seamless user experience. The AutoComplete Editor aims to revolutionize the typing experience by offering personalized suggestions tailored to individual users. Traditional auto-complete editors provide generic suggestions, but our editor learns from the user's typing behavior to offer contextually relevant suggestions. Leveraging data structures like Tries, Bigram, and Trigram Models, our editor builds a comprehensive understanding of the user's writing style and preferences. Tries complete incomplete words while Bigram and Trigram Models predict the next word based on the context.

Data Structures Used

- **Tries (Prefix Tries):** For AutoCompletion of incomplete word. A tree-like data structure used to store a dynamic set of strings where each node represents a common prefix of its children.
  - **Complexity:** Searching for a word or prefix in a trie has a time complexity of $O(m)$, where m is the length of the word or prefix.
- **Bigram Model:** For predicting next word. A statistical language model used for predicting the next word in a sequence based on the previous word.
  - **Complexity:** Generating a bigram model typically involves scanning the text corpus to count occurrences of word pairs. The complexity is $O(n)$, where n is the size of the text corpus.
- **Trigram Model:** For predicting next word. Similar to the Bigram Model, but predicts the next word based on the previous two words.
  - **Complexity:** Creating a trigram model involves counting occurrences of triplets of words in the text corpus. The complexity is $O(n)$, where n is the size of the corpus.
- **Hashmaps:** Used as a FrequencyMap for efficient ranking of suggested words. A collection of key-value pairs used to store and retrieve data efficiently based on keys.
  - **Complexity:** Insertion, deletion, and retrieval in dictionaries typically have an average time complexity of $O(1)$, though it can vary based on the specific implementation.

Application Features

- **Login Screen:** Users are greeted with a login screen upon opening the application, where they can create an account or log in.
- **Text Suggestion:** The application suggests next words in real-time as the user types, enhancing productivity and reducing typing effort. These are personalized to each user based on their typing history.
- **Text Completion:** Auto-completion functionality is integrated into the keyboard, providing quick word suggestions for seamless typing.

Challenges Faced

- **Optimization of Tries:** Optimizing the Trie data structure for efficient storage and retrieval of large datasets posed a significant challenge.
- **Ranking of Suggested Words:** Implementing an effective ranking algorithm for suggested words based on user preferences and relevance was challenging.
- **Research for Next Word Prediction:** Finding and implementing a data structure capable of accurately predicting the next word based on context was a complex task.
- **Integration with Frontend:** Integrating the C++ backend with the NextJS frontend presented technical hurdles and required careful coordination.

Work Division

The project was divided among team members based on their expertise and interests. Each member contributed to various aspects of development, including frontend design, backend implementation, algorithm development, and testing.

- **Muhammad Hassan:** Frontend Development and User Personalisation
- **Muhammad Qasim Khan:** Tries Manipulation
- **Muhammad Saad:** Bi/Trigram and Trie Implementation
- **Syed Muhammad Ali Imran:** User Ranking and Frontend Development

References

1. `https://www.geeksforgeeks.org/python-bigram-formation-from-given-list/`
2. `https://www.geeksforgeeks.org/tf-idf-for-bigrams-trigrams/`
3. `https://www.geeksforgeeks.org/trie-insert-and-search/`