**Braille Auto-Correct and Suggestion System**

**By Pranay Malhotra**  
Email: pranaymalhotra2003@gmail.com  
LinkedIn: <https://www.linkedin.com/in/pranay-malhotra-190818343/>

**Project Overview**

This project implements a basic auto-correct and suggestion system for Braille input typed via a standard QWERTY keyboard. It is designed to assist users who input Braille characters using six specific keys (D, W, Q, K, O, P), each representing one of the six Braille dots. The system accepts sequences of these Braille dot combinations and returns the closest matching word(s) from a predefined dictionary.

Due to the inherent challenges of Braille input on QWERTY keyboards—such as accidental extra or missing dots—this system focuses on correcting common input errors to improve user experience and minimize frustration.

**Approach**

* **Input Parsing:** The system accepts multiple keys pressed simultaneously to represent a single Braille character, parsing the input into sets of dots corresponding to Braille patterns.
* **Dictionary Mapping:** A dictionary of words is converted into their Braille key sequence equivalents to facilitate pattern matching.
* **Similarity Scoring:** Using a custom dot-overlap scoring algorithm, the system compares the input sequence against dictionary entries to find the closest matches.
* **Suggestions:** The top three most similar words based on this score are suggested to the user.
* **Modular Design:** The architecture is modular, making it easy to extend with more advanced algorithms in the future.

**Current Features**

* Supports Braille input via QWERTY keys representing Braille dots.
* Parses and matches multi-character Braille sequences.
* Provides up to three suggestions based on similarity scores.
* Does not rely on external libraries, ensuring simplicity and ease of use.

**Future Enhancements**

* **Length-Tolerant Matching:** Improve handling of missing or extra dot inputs.
* **Advanced Algorithms:** Incorporate Levenshtein distance and Trie data structures to boost accuracy and speed.
* **Real-Time Suggestions:** Enable live suggestions as the user types.
* **Learning Mechanism:** Adapt suggestions based on user corrections and usage patterns.
* **User Interface:** Develop a GUI or web-based front-end to enhance accessibility and usability.
* **Support for Braille Contractions and Multiple Languages:** Extend dictionary support to include contractions and other languages.

**Closing Note**

This is an initial draft of the Braille Auto-Correct and Suggestion System, developed to demonstrate core functionality and provide a foundation for future improvements. I am eager to refine and expand the system with advanced features and optimizations, ideally through an internship opportunity. Constructive feedback and suggestions are welcome.

Thank you for reviewing my submission.

— Pranay Malhotra