High Level Design (HLD)

Spelling Corrector System(Corrspell)

Revision Number: 2.0

Last date of revision: 13/09/2023

Document Version Control

Date Issued	Version	Description	Author
20/08/2023	1	Initial HLD - VI .0	Amit Ranjan
13/09/2023	2	Updated - VI .1	Amit Ranjan

Contents

Abstract

The Spelling Corrector System is designed to automatically detect and correct spelling errors in text input. This document outlines the high-level design aspects of the system.

1 Introduction

1.1 Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

- Present all of the design aspects and define them in detail
- Describe the user interface being implemented
- Describe the hardware and software interfaces
- Describe the performance requirements
- Include design features and the architecture of the project
- List and describe the non-functional attributes like:
  - Security
  - Reliability o Maintainability
  - Portability
  - Reusability
  - Application compatibility o Resource utilization
  - Serviceability

1.2 Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

1.3 Definitions

Term	Description
Spelling corrector	A system that automatically detects and corrects spelling errors in text.
IDE	Integrated Development Environment

API	Application programming interface

2 General Description

2.1 Product Perspective

The Spelling Corrector System is a standalone application designed to be integrated into various text-based applications. It acts as a service that takes text input, processes it using a spelling correction algorithm, and returns the corrected text.

2.2 Problem statement

The problem addressed is the prevalence of spelling errors in text content, which can lead to misunderstandings and miscommunication. The Spelling Corrector System aims to enhance the quality of written content by providing accurate and efficient spelling correction.

2.3 PROPOSED SOLUTION

The solution involves creating a spelling correction engine that utilizes a combination of linguistic algorithms, dictionary lookups, and statistical models. This engine will analyze input text, identify potential spelling errors, and suggest corrected replacements.

2.4 Technical Requirements

The Spelling Corrector System should:
- Accept text input from various sources.
- Identify misspelled words.
- Suggest accurate corrections.
- Handle different languages.
- Integrate with applications through APIs.

2.5 Tools and Technologies

Python programming language and frameworks such as NumPy, Pandas, Scikit-learn, TensorFlow and Keras are used to build the whole model.

The system will be developed using:
- Programming language (e.g., Python).
- Natural Language Processing (NLP) libraries.
- Dictionaries and language models.

2.6 Constraints

- The correction process should be fast and efficient.
- The system should handle a large volume of text input.
- Language-specific intricacies must be considered.

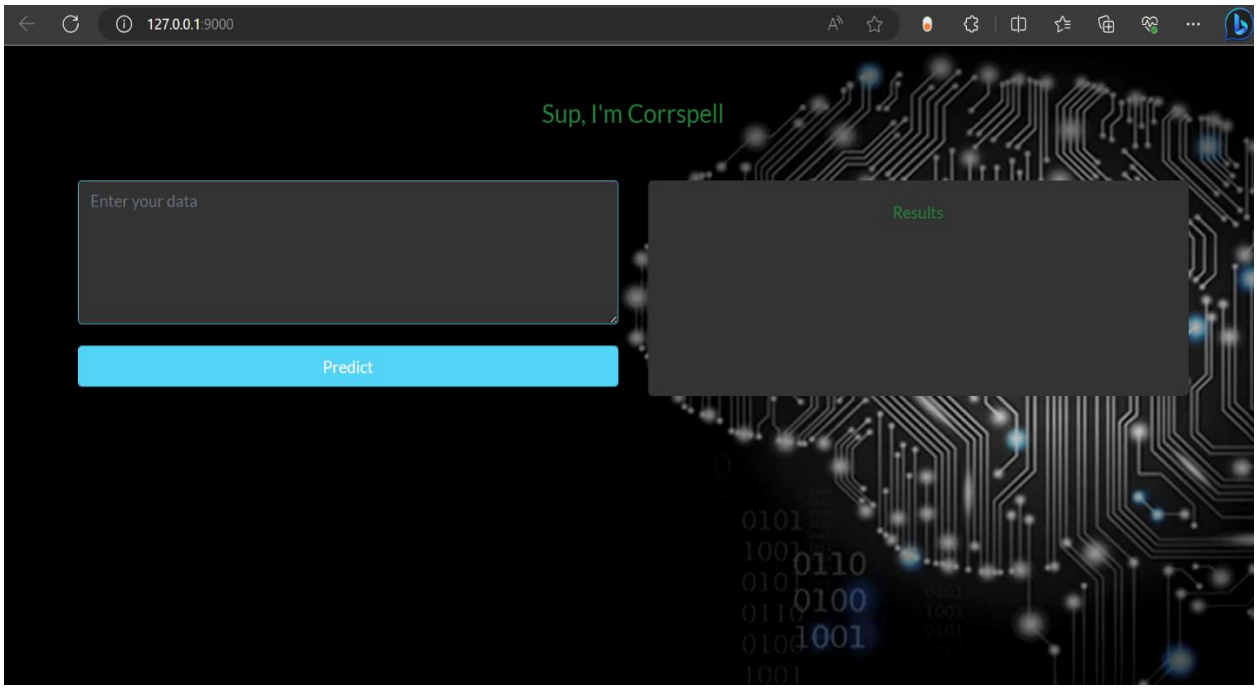
2.7 Assumptions

- The system will have access to a comprehensive dictionary.
- Users expect near-instantaneous correction suggestions.

Design Details

3.1 User Interface

The user interface will vary based on the integration point. For standalone usage, a simple web interface or command-line interface may be provided. For integration with applications, an API will handle input and output.



3.2 Core Spelling Correction Engine

The core engine will consist of:
- Preprocessing: Tokenizing and normalizing input text.
- Error Detection: Identifying potential spelling errors.
- Suggestion Generation: Generating correction suggestions.
- Ranking: Ranking suggestions based on context and frequency.

3.3 Integration with Applications

The system can be integrated into text editors, messaging apps, content management systems, and more. APIs will allow seamless integration, enabling applications to send text for correction and receive corrected text.

3.4 Performance

The system should provide real-time or near-real-time spelling correction for small to moderately sized text inputs. Response times should be within milliseconds to ensure a smooth user experience.

Deployment

The Spelling Corrector System can be deployed as a standalone web application or as a microservice in a cloud environment. APIs will facilitate integration with various applications.

Conclusion

The Spelling Corrector System aims to enhance written communication by automatically identifying and rectifying spelling errors. This document has outlined the high-level design aspects of the system, providing a foundation for its development.