Low Level Design (LLD)

SPELLING CORRECTOR

Project By :

Apurv Bhusari

Document Version control

Date issued	Version	Description	Author
21/04/2023	1	Initial LLD	Apurv Bhusari
15/07/2023	2	Updated	Apurv Bhusari
10/10/2023	3	Final	Apurv Bhusari

Contents

ABSTRACT

The goal of this project is to build a NLP based end to end system which is predict wrongly input words from the user end using state of the art model.

1 Introduction

1.1 Why this Low-Level Design Document?

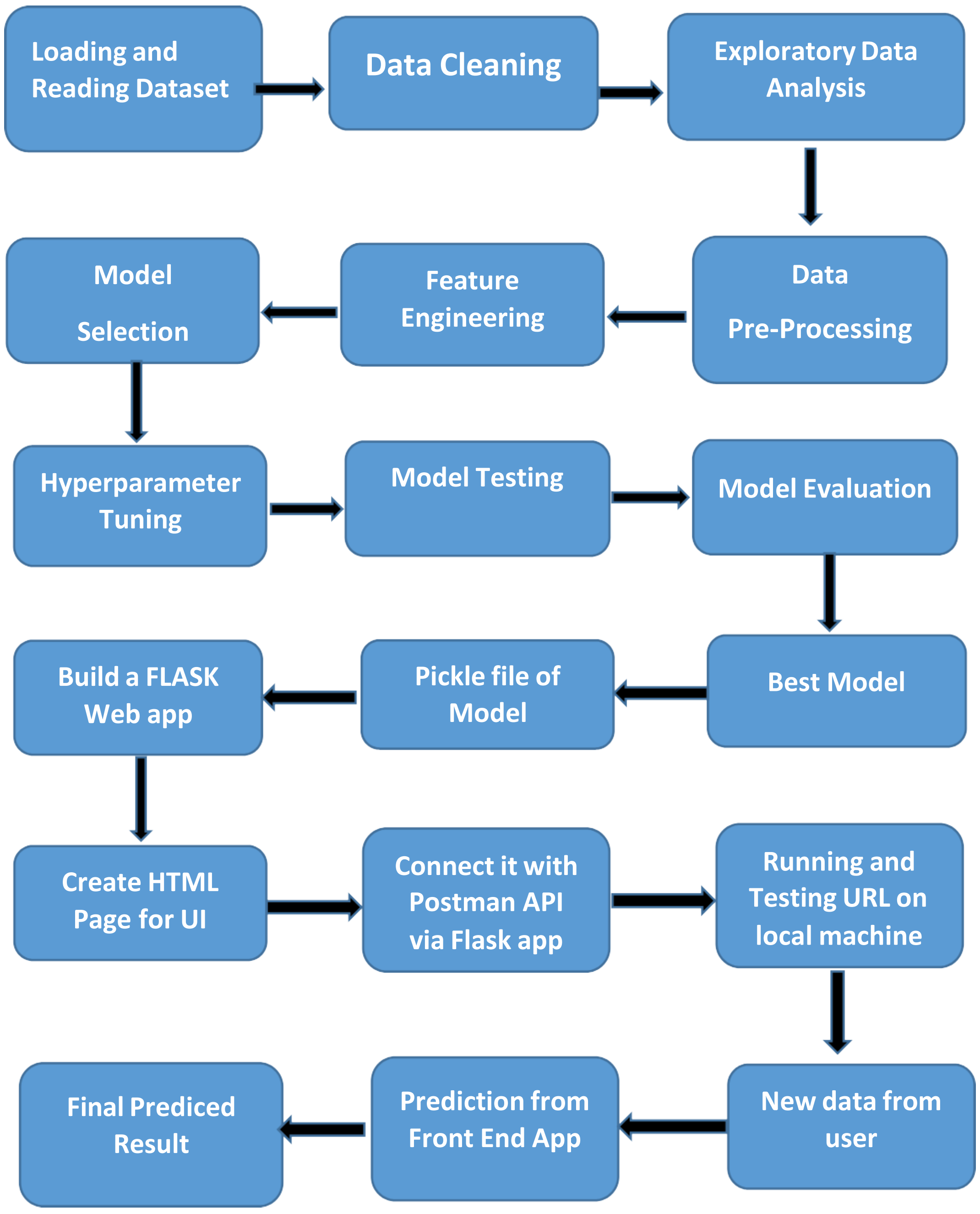
The goal of the Low-level design document (LLD) is to give the internal logic design of the actual program code for the Spelling Corrector.

1.2. Scope

Low-level design (LLD) is a component-level design process that follows a step-by

step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work.

2. Architecture



3. Architecture Description

3.1 Loading and Reading Dataset

The primary source of data for this project from Kaggle.The dataset is comprised of 10,324 records.

3.2 Data Cleaning

Data cleaning is the process of detecting and correcting errors, inconsistencies, or missing values in a dataset. It is an essential step in data pre-processing and is often the most time-consuming part of the data analysis process.

There are many different techniques for cleaning data, depending on the nature of the dataset and the types of errors it contains. Some common techniques include:

- ✓ Identifying and removing duplicates
- ✓ Handling missing values (e.g., replacing them with a default value or dropping them)
- ✓ Correcting data formatting issues (e.g., inconsistent date formats)
- ✓ It is important to carefully inspect the data and carefully consider the appropriate cleaning techniques to apply, as these can significantly affect the quality and usefulness of the data.

3.3 Exploratory Data Analysis (EDA)

Exploratory Data Analysis refers to the critical process of performing initial investigations on data to discover patterns, spot anomalies, test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

3.4 Data/Text Pre-Processing

Before building any model, it is crucial to perform pre-processing tofeed

the correct data to the model to learn and predict. Model performance

depends on the quality of data fed to the model to train.

This Process includes:

- ✓ Data transformation: Modifying the data to fit the requirements of the analysis or model being used.
- ✓ Text Cleaning: Before giving data to anyone Data reduction: Reducing the number of features or samples in the dataset toimprove the efficiency of the analysis or model.
- ✓ Outlier Treatment: Outliers Detection and Removal

3.5 Feature Engineering

Feature engineering is the process of designing and creating new features or variables from existing data to improve the performance of a machine learning model. It is a key step in the data preprocessing process, as the quality and relevance of the features can significantly affect the model's ability to learn and generalize.

Some common techniques for feature engineering include:

- ✓ Feature selection: Selecting a subset of the most relevant features from the data.
- ✓ Feature extraction: Creating new features from existing data by applying transformations or combining existing features.
- ✓ Feature creation: Generating new features based on domain knowledge or by applying algorithms to the data.

3.6 Model Selection

In This step we apply state of the art models to our processed data and select the model with best results for further fine-tuning to make the best possible model that can be made for accurate and correct prediction.

Model building is an iterative process, and the specific steps and techniques used will depend on the nature of the data and the goals of the analysis.

3.7 Model Evaluation

We evaluate our

3.8 Deployment Process

To deploy a model, we used the following steps:

1. Save the trained model as a h5file using Python's pickle library.
2. Create a Flask app in Python, which will act as the server for your model.
3. Define the routes for the Flask app, which will determine the behavior of the server when it receives different HTTP requests.
4. In the routes, you can load the pickle file and use it to make predictions based on the input received in the request.
5. We created HTML templates to display the results of the predictions on a website.
6. Test the Flask app using Postman or a similar API testing tool to ensure it is working correctly.

4 Unit Test Cases

Test Case Description	Pre-Requisite	Expected Result
Verify whether the Application URL is accessible to the user	1. Application URL should be defined.	1. Application URL should be accessible to the user.
Verify whether the application loads completely for the user when the URL is accessed	1. Application URL is accessible 2. Application URL is deployed.	Application URL should load completely for the user when URL is accessed.
Verify whether user can see input field after opening URL	1. Application is accessible	User should be able to see input fields after opening URL
Verify whether user can edit all the input fields	1. Application is accessible	User should be able to edit all the input fields
Verify whether user has options to filter the inputs fields	1. Application is accessible	User should filter the options of input fields
Verify whether user gets submit button to submit the inputs	1. Application is accessible	User should get submit button to submit the inputs
Verify whether user can see the output after submitting the inputs	1. Application is accessible	User should get outputs after submitting the inputs