GENIE

TEXT ANALYSER AND CORRECTOR

B-14

BY-
LAHAARIKA RAJUU - PES2UG25AM138
MANASI TRIVEDI - PES2UG25AM152
M SAI SRIVANI - PES2UG25EC070
NIBEDITA BANERJEE - PES2UG25CS330

TABLE OF CONTENTS

PROBLEM STATEMENT

With the rise of digitalisation, typed content has become the primary mode of communication. However, this shift has also made grammar and spelling errors more common.

Traditional grammar checkers rely heavily on rule-based correction methods.

This problem can be addressed by leveraging a hybrid approach that combines rule-based grammar correction—such as spell checking using Levenshtein distance—with context-aware corrections powered by machine learning models like BERT, T5, and LLaMA. This integration ensures both accuracy in detecting surface-level errors and relevance in understanding the deeper context of the text.

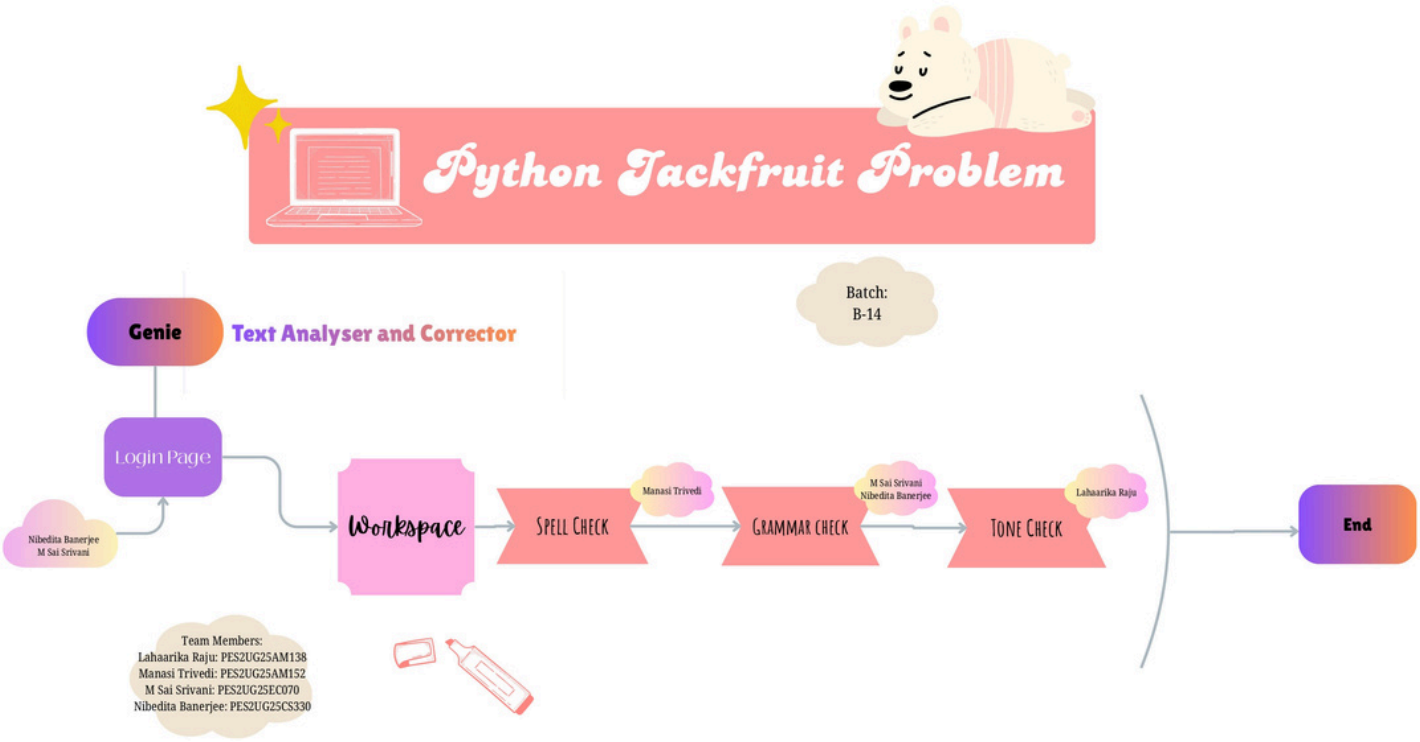
Objectives:

- Detect misspelled words and incorrect grammar in user-entered text.
- Uses contextual embeddings and trained models to provide the most context specific corrections.
- Suggests the top corrections for the user to choose from.
- Automatically reconstructs the corrected sentence based on the user's choice.
- Evaluates its performance using metrics such as ROUGE .

The final deliverables should include:

- A python implementation of the text analyser and corrector.
- A user friendly interface (using Streamlit).
- Display the corrected text split into grammar corrections, spelling corrections and the detected tone.

BLOCK DIAGRAM



APPROACH USED

Grammar Checker:

1. Editing the dataset into a suitable format of incorrect sentences, correct sentences and error type.
2. Tokenizing this text and training a sequence to sequence model on the modified dataset.
3. The output of the model is then checked against a surely correct version which is provided by LLaMA via an API.

Tone checker:

1. The entered text is tokenized and stored.
2. The dataset has sentences, and their respective tones. This construct is then tokenized and assigned their own ids.
3. This text is now mapped and reverse mapped to the ids of their respective tones.
4. Once the model is trained on the mapped and reverse mapped texts, predictions of the tone can be made.

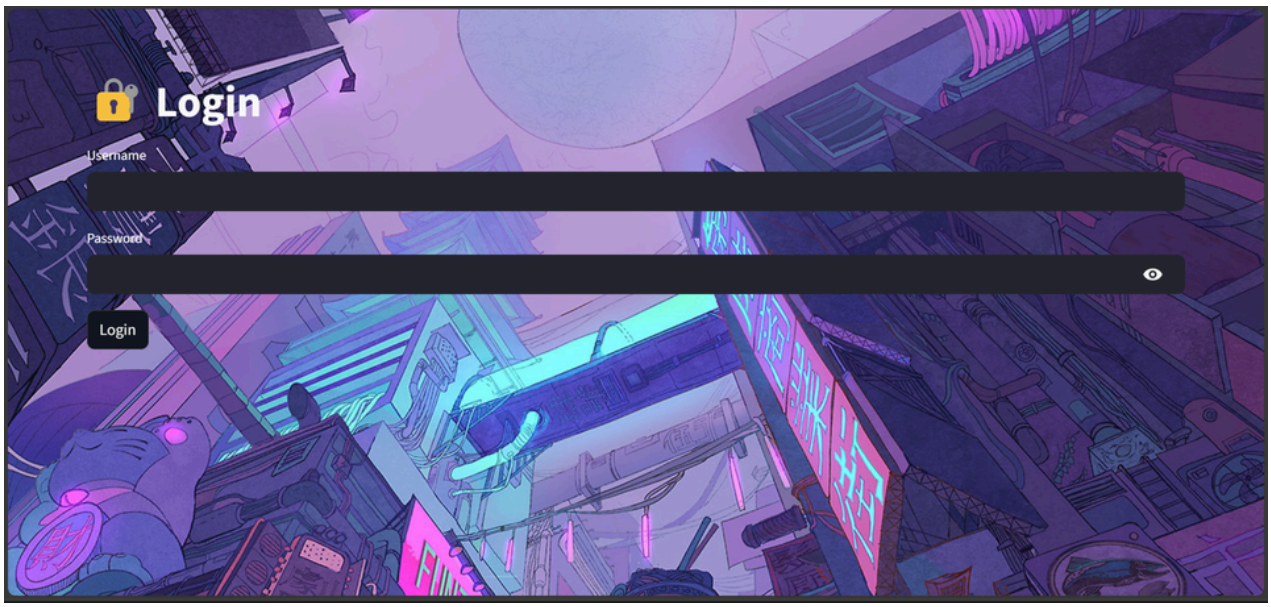
Spell checker:

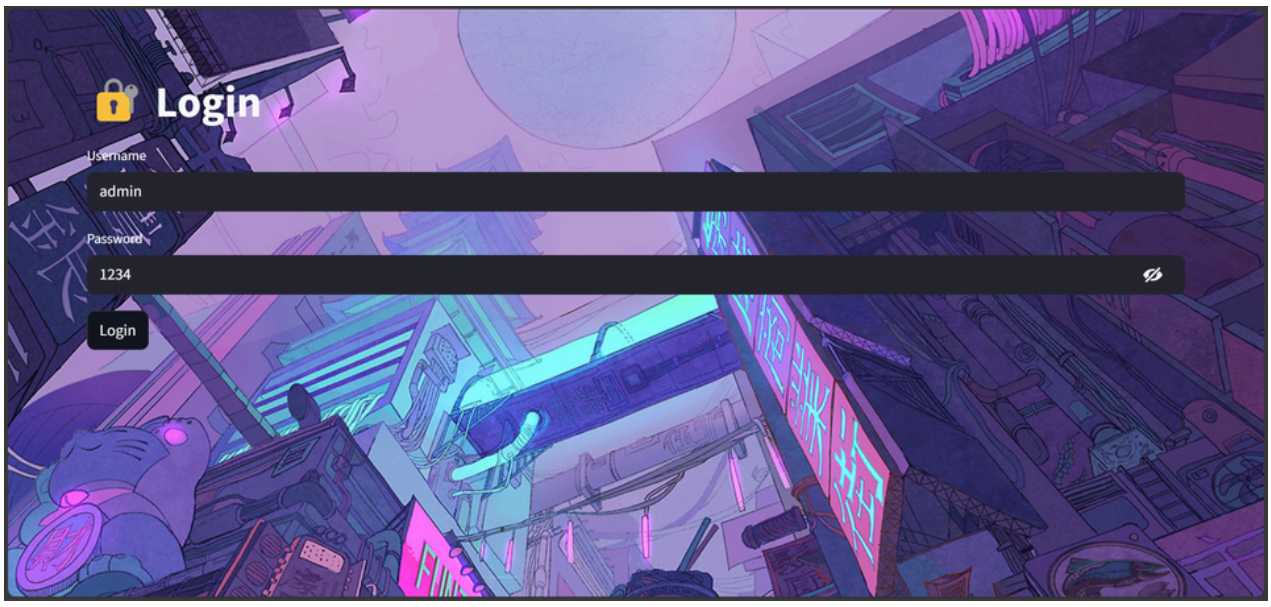
1. Tokenizing all the words and punctuations from input.
2. Uses Levenshtein distance to calculate the closest correct word to the input using a text file of words as reference.
3. Retains punctuation, capitalization and appends the correct words with the closest words.

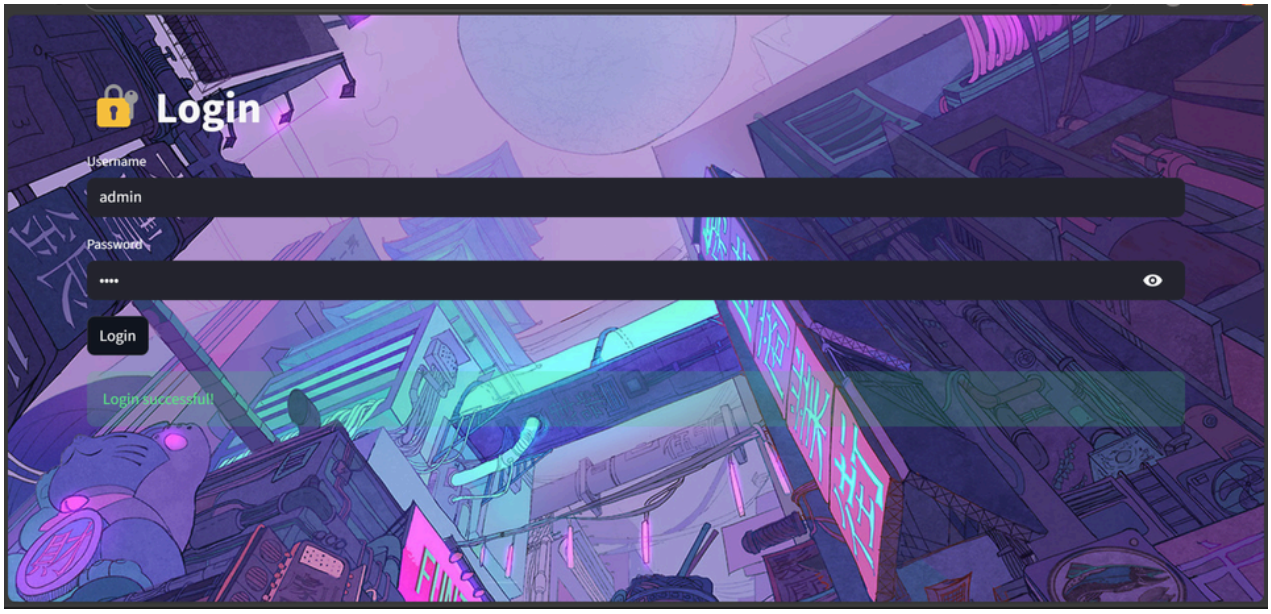
UI Design:

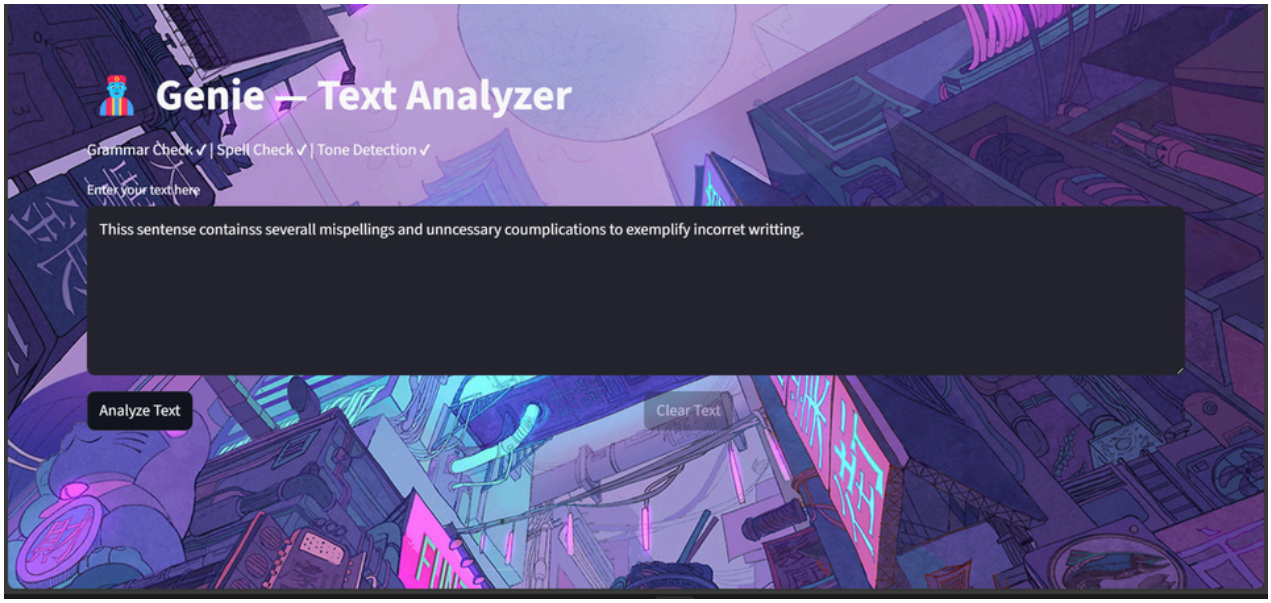
1. Firstly, a mostly python based approach was chosen which was Streamlit.
2. Text boxes and buttons were added in correspondence with its constructs to ensure a responsive, interactive user interface was developed.
3. A login page is followed by a home page where the user is prompted to enter their text.
4. The user can then press the analyze text button to receive their analyzed corrected text in the same page along with the tone.

SAMPLE INPUT/OUTPUT






Login
Username
admin
Password
Login
Login successful!


Genie – Text Analyzer
Grammar Check ✓ | Spell Check ✓ | Tone Detection ✓
Enter your text here
Thiss sentense containss severall mispellings and unncessary coumplications to exemplify incorret writting.
Analyze Text
Clear Text


Grammar Correction
This sentence contains several misspellings and unnecessary complications to exemplify incorrect writing
Spell Correction
This sentence contains several misspellings and unnecessary complications to exemplify incorrect writing .
Tone Detection
formal

CHALLENGES FACED

Spell checker:

1. Lack of publicly available large and compatible datasets.
2. Inaccurate predictions due to usage of only Levenshtein distance.
3. Inability to completely rectify spelling errors due to finding only the closest match causing incorrect rectifications.

Grammar checker:

1. Lack of datasets in required format or required size.
2. Error propagation in multi model approach leading to extremely low accuracy.
3. Bulkiness of text generation models caused significant issues in training.

Tone checker:

1. Lack of suitable labelled datasets which caused errors while training.
2. The model's tendency to memorise instead of learning due to mild difference in parameters.
3. unshuffled datasets causing lower accuracy.

UI Design:

1. Compatibility issues arose between the front end and backend code mainly due to differences in specifications of inpits and outputs.

Integration:

1. The diversity in functions posed issues due to repeated variable names or difference in input and return parameter.
2. Storage of models and trainung arguments under functions.

THANK YOU