PYTHON JACKFRUIT PROBLEM

GENIE

TEXT ANALYSER AND CORRECTOR

PRESENTED BY:

Lahaarika Rajuu
PES2UG25AM138

Manasi Trivedi
PES2UG25AM152

M Sai Srivani
PES2UG25EC070

Nibedita Banerjee
PES2UG25CS330

PROBLEM STATEMENT

With the rise in digitization of organizations and services , the advent of text errors is inevitable. However, classical rule based correction approaches eliminate context . Thus, a more context aware corrections can be generated by the usage of ML with models such as T5, BERT, and Levenshtein distance.h the digitization of organizations and services., the advent of text errors is inevitable.. Classical rule based approach eliminates the context, hence incorrectly modifying the text. Leveraging machine learning and the classical approach of Levenshtein distance , Genie intends to correct the text more accurately.

Implementation of a text analyser and corrector using hybrid NLP techniques

Objectives:

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

The final deliverable should include:

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

CONTRIBUTIONS

Lahaarika Rajuu
PES2UG25AM138

Tone checker using Machine Learning by an encoder decoder model BERT (base_uncased).

Manasi Trivedi
PES2UG25AM152

Spell checker and corrector by a rule based approach using Levenshtein distance.

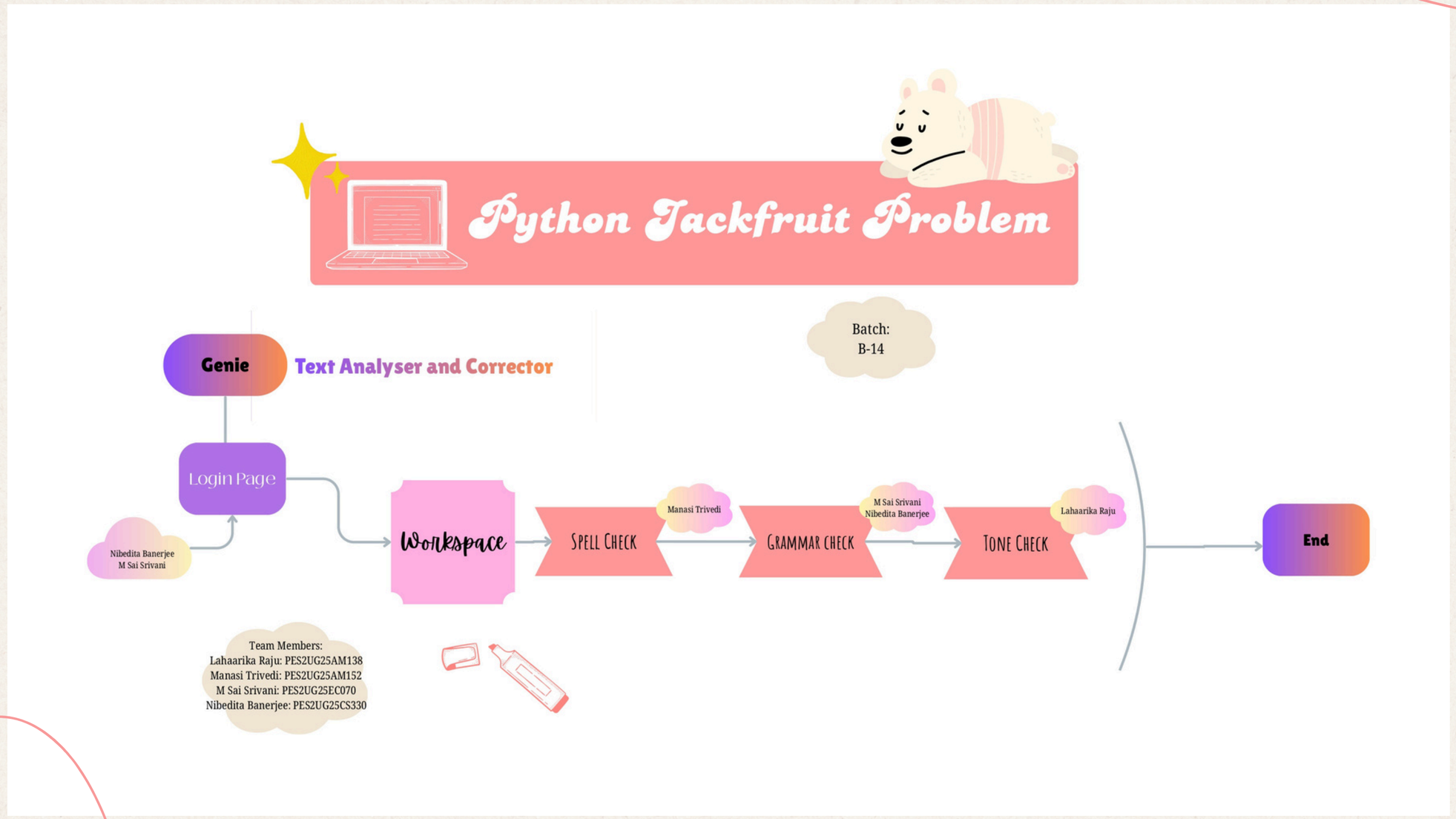
M Sai Srivani
PES2UG25EC070

Grammar checker and corrector using Machine Learning by a sequence to sequence model T5 (small),

Nibedita Banerjee
PES2UG25CS330

Front end user interface using Streamlit and integration of the project.

BLOCK DIAGRAM



UI/UX

```
import streamlit as st
from grammar_checker import grammar_check
from spelling_checker import spell_check
from tone_checker import tone_check
import base64

# ------------------------
# ADD BACKGROUND IMAGE
# ------------------------

import streamlit as st
# your other imports...

st.set_page_config(layout="wide")

# remove top padding
st.markdown("""
    <style>
        .block-container {
            padding-top: 0rem;
            padding-bottom: 0rem;
        }
    </style>
```

Done By:
Nibedita Banerjee

Spell Checker

```python
import nltk
from nltk.tokenize import word_tokenize
import Levenshtein
import string


def closest_words(word, word_list):
    same_start = [w for w in word_list if w.startswith(word[0])]
    suggestions = same_start if same_start else word_list
    ranked = sorted(suggestions, key=lambda w: Levenshtein.distance(word, w))
    return ranked[:3]


def spell_check(in_text):
    # Load dictionary properly
    with open('words_final.txt', 'r') as f:
        words = [w.strip().lower() for w in f if w.strip()]

    highlighted = ""
    final_sen = []

    # Tokenize
    tokens = word_tokenize(in_text)
```

Done By:
Manasi Trivedi

Grammar Checker

```python
import torch
from transformers import T5TokenizerFast, T5ForConditionalGeneration

# ------------------------------------------------
# Load model ONCE (top-level)
# ------------------------------------------------
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

try:
    tokenizer = T5TokenizerFast.from_pretrained("saved_grammar_model")
    model = T5ForConditionalGeneration.from_pretrained("saved_grammar_model")
except Exception as e:
    print("❌ Could not load saved grammar model:", e)
    raise

model.to(device)
model.eval()

# ------------------------------------------------
# Function ONLY does inference (NO model loading!)
# ------------------------------------------------
def grammar_check(text):
    if not text:
        return "⚠️ No input text provided."
```

Done By:
M Sai Srivani

Tone Checker

```python
def tone_check(grammar_output):
    from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments
    from datasets import load_dataset, Dataset
    import pandas as pd
    import torch
    import pandas as pd
    from datasets import load_dataset

    import os
    os.environ["TRANSFORMERS_NO_ADVISORY_WARNINGS"] = "true"
    os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"
    os.environ["TRANSFORMERS_VERBOSITY"] = "error"
    import warnings
    warnings.filterwarnings("ignore", category=UserWarning, module="transformers.modeling_utils")
    warnings.filterwarnings("ignore")

    import logging
    logging.getLogger("transformers").setLevel(logging.ERROR)
    logging.getLogger("torch").setLevel(logging.ERROR)

    TRAIN = False

    new = pd.read_csv("pls.csv")
    new.head(10)
```

Done By:
Lahaarika Rajuu

Thank you