

# ChatGPTChatGPT



```
from tkinter import filedialog
import string
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statistics import mean

# Constants for column names
CNST_TEXT_LINE_NAME = "Input Text"
CNST_TEST_LANGDETECT_LANG_LINE_NAME = "LangDetect: Detected Language"
CNST_TEST_LANGDETECT_PROB_LINE_NAME = "LangDetect: Confidence Score"
CNST_TEST_LANGID_LANG_LINE_NAME = "LangID: Detected Language"
CNST_TEST_LANGID_PROB_LINE_NAME = "LangID: Confidence Score"

# Output file names
CNST_LANG_DETECTION_RESULTS_FILE_NAME = "Language_Detection_Results.csv"
CNST_LANG_DETECTION_DIFFERENCES_FILE_NAME = "Detection_Differences.csv"

# Functions for detection
def detect_language_with_langdetect(line):
    from langdetect import detect_langs
    try:
        langs = detect_langs(line)
        for item in langs:
            return item.lang, item.prob
    except:
        return "Unknown", 0.0

def detect_language_with_langid(line):
    from langid import classify
    lang, prob = classify(line)
    return lang, prob

def write_data_to_csv(lines, langdetect_lang_results, langdetect_prob_results, langid_lang_results,
langid_prob_results):
    data = {
        CNST_TEXT_LINE_NAME: lines,
        CNST_TEST_LANGDETECT_LANG_LINE_NAME: langdetect_lang_results,
        CNST_TEST_LANGDETECT_PROB_LINE_NAME: langdetect_prob_results,
        CNST_TEST_LANGID_LANG_LINE_NAME: langid_lang_results,
        CNST_TEST_LANGID_PROB_LINE_NAME: langid_prob_results
    }
    df = pd.DataFrame(data)
    df.to_csv(CNST_LANG_DETECTION_RESULTS_FILE_NAME, sep="|", index=False)
```

```
def extract_differences(file_name):
    df = pd.read_csv(file_name, sep="|")
    differences = df[df[CNST_TEST_LANGDETECT_LANG_LINE_NAME] !=
df[CNST_TEST_LANGID_LANG_LINE_NAME]]
    differences.to_csv(CNST_LANG_DETECTION_DIFFERENCES_FILE_NAME, sep="|", index=False)

# Visualization for Non-Technical Users
def show_results_for_non_tech_users(file_name):
    df = pd.read_csv(file_name, sep="|")
    sns.set_theme(style="whitegrid")

    print("\nLanguage Detection Results Summary")
    print("=" * 50)

    # Display simple text-based insights
    detected_languages_langdetect = df[CNST_TEST_LANGDETECT_LANG_LINE_NAME].value_counts()
    detected_languages_langid = df[CNST_TEST_LANGID_LANG_LINE_NAME].value_counts()

    print("\nMost Common Languages Detected (LangDetect):")
    print(detected_languages_langdetect)

    print("\nMost Common Languages Detected (LangID):")
    print(detected_languages_langid)

    # Create visualizations
    fig, axes = plt.subplots(2, 2, figsize=(16, 12))

    # Pie chart for LangDetect
    axes[0, 0].set_title("LangDetect - Detected Languages", fontsize=14)
    detected_languages_langdetect.plot.pie(
        autopct='%1.1f%%',
        startangle=90,
        ax=axes[0, 0],
        colors=sns.color_palette("pastel"),
    )
    axes[0, 0].set_ylabel("")

    # Pie chart for LangID
    axes[0, 1].set_title("LangID - Detected Languages", fontsize=14)
    detected_languages_langid.plot.pie(
        autopct='%1.1f%%',
        startangle=90,
        ax=axes[0, 1],
        colors=sns.color_palette("pastel"),
    )
    axes[0, 1].set_ylabel("")

    # Bar chart for LangDetect confidence
    langdetect_mean_probs = df.groupby(CNST_TEST_LANGDETECT_LANG_LINE_NAME)
[CNST_TEST_LANGDETECT_PROB_LINE_NAME].mean()
```

```

langdetect_mean_probs.sort_values().plot.barh(ax=axes[1, 0], color="skyblue")
axes[1, 0].set_title("LangDetect - Mean Confidence Score", fontsize=14)
axes[1, 0].set_xlabel("Mean Confidence")

# Bar chart for LangID confidence
langid_mean_probs = df.groupby(CNST_TEST_LANGID_LANG_LINE_NAME)
[CNST_TEST_LANGID_PROB_LINE_NAME].mean()
langid_mean_probs.sort_values().plot.barh(ax=axes[1, 1], color="skyblue")
axes[1, 1].set_title("LangID - Mean Confidence Score", fontsize=14)
axes[1, 1].set_xlabel("Mean Confidence")

plt.tight_layout()
plt.savefig("Language_Detection_Visualization.png")
plt.show()

print("\nCharts saved as 'Language_Detection_Visualization.png'.")
print("Detailed results are saved in 'Language_Detection_Results.csv'.")
print("Any differences are saved in 'Detection_Differences.csv'.")

# Main Workflow
file_path = filedialog.askopenfilename()
with open(file_path, 'r', encoding='utf-8') as file:
    lines = file.readlines()

langdetect_lang_results = []
langdetect_prob_results = []
langid_lang_results = []
langid_prob_results = []
raw_text = []

for line in lines:
    line = line.translate(str.maketrans("", "", string.punctuation)).rstrip()
    raw_text.append(line)

    langdetect_lang, langdetect_prob = detect_language_with_langdetect(line)
    langdetect_lang_results.append(langdetect_lang)
    langdetect_prob_results.append(langdetect_prob)

    langid_lang, langid_prob = detect_language_with_langid(line)
    langid_lang_results.append(langid_lang)
    langid_prob_results.append(langid_prob)

write_data_to_csv(raw_text, langdetect_lang_results, langdetect_prob_results, langid_lang_results,
langid_prob_results)
extract_differences(CNST_LANG_DETECTION_RESULTS_FILE_NAME)
show_results_for_non_tech_users(CNST_LANG_DETECTION_RESULTS_FILE_NAME)

explain code line by line

```



Sure! Let's break down the code step by step:

## Imports

python

```
from tkinter import filedialog
import string
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statistics import mean
```

1. `filedialog`: This is from the `tkinter` library and is used to open a file dialog to let the user select a file for processing.
2. `string`: Provides a string-related utility, specifically for punctuation removal.
3. `pandas`: A data manipulation library, used for creating and handling data frames.
4. `matplotlib.pyplot`: A plotting library for creating visualizations, specifically used here to generate charts.
5. `seaborn`: A statistical data visualization library built on top of Matplotlib, used for making visually appealing charts.
6. `mean`: From Python's `statistics` module, provides a function to calculate the mean value, although this isn't directly used in the code.

## Constants

python

```
CNST_TEXT_LINE_NAME = "Input Text"
CNST_TEST_LANGDETECT_LANG_LINE_NAME = "LangDetect: Detected Language"
CNST_TEST_LANGDETECT_PROB_LINE_NAME = "LangDetect: Confidence Score"
CNST_TEST_LANGID_LANG_LINE_NAME = "LangID: Detected Language"
CNST_TEST_LANGID_PROB_LINE_NAME = "LangID: Confidence Score"
```

- These are constant variables used for column names in the data frame that will store language detection results. It helps to avoid hardcoding strings repeatedly and makes the code more maintainable.

python

```
CNST_LANG_DETECTION_RESULTS_FILE_NAME = "Language_Detection_Results.csv"
CNST_LANG_DETECTION_DIFFERENCES_FILE_NAME = "Detection_Differences.csv"
```

- These constants specify the output file names where the results and differences will be saved.

## Functions for Language Detection

python

```
def detect_language_with_langdetect(line):
    from langdetect import detect_langs
    try:
        langs = detect_langs(line)
        for item in langs:
            return item.lang, item.prob
    except:
        return "Unknown", 0.0
```

- `detect_language_with_langdetect(line)`: Detects the language of a given line of text using the `langdetect` library.
  - `detect_langs(line)` returns a list of detected languages with their associated probabilities.
  - If successful, it returns the language and probability of detection; otherwise, it returns "Unknown" and a confidence of 0.

python

```
def detect_language_with_langid(line):
    from langid import classify
    lang, prob = classify(line)
    return lang, prob
```

- `detect_language_with_langid(line)`: Detects the language of a line of text using the `langid` library.
  - `classify(line)` returns the language and its associated probability.

## Write Data to CSV

python

```
def write_data_to_csv(lines, langdetect_lang_results, langdetect_prob_results, langid_lang_r
    results, langid_prob_results):
    data = {
        CNST_TEST_LINE_NAME: lines,
        CNST_TEST_LANGDETECT_LANG_LINE_NAME: langdetect_lang_results,
        CNST_TEST_LANGDETECT_PROB_LINE_NAME: langdetect_prob_results,
        CNST_TEST_LANGID_LANG_LINE_NAME: langid_lang_results,
        CNST_TEST_LANGID_PROB_LINE_NAME: langid_prob_results
    }
    df = pd.DataFrame(data)
    df.to_csv(CNST_LANG_DETECTION_RESULTS_FILE_NAME, sep="|", index=False)
```

- `write_data_to_csv`: Writes the results of language detection to a CSV file.
  - Takes in the raw lines of text and the detected languages/probabilities from both language detectors.
  - The data is organized into a dictionary, which is then converted into a DataFrame using `pandas`.
  - The DataFrame is saved as a CSV file (`Language_Detection_Results.csv`) with a pipe (`|`) separator.

## Extract Differences

python

```
def extract_differences(file_name):
    df = pd.read_csv(file_name, sep="|")
    differences = df[df[CNST_TEST_LANGDETECT_LANG_LINE_NAME] != df[CNST_TEST_LANGID_LANG_LINE_NAME]]
    differences.to_csv(CNST_LANG_DETECTION_DIFFERENCES_FILE_NAME, sep="|", index=False)
```

- `extract_differences`: Compares the language detection results from both `LangDetect` and `LangID`.
  - It reads the CSV file where the results are stored.
  - Filters the rows where the languages detected by `LangDetect` and `LangID` are different.
  - Saves these differences to another CSV file (`Detection_Differences.csv`).

## Show Results for Non-Technical Users (Visualization)

python

```
def show_results_for_non_tech_users(file_name):
    df = pd.read_csv(file_name, sep="|")
    sns.set_theme(style="whitegrid")
```

- `show_results_for_non_tech_users`: This function generates visual reports that can be easily interpreted by non-technical users.
  - Reads the CSV file with the language detection results.
  - Sets a simple visual theme for the plots using `seaborn`.

python

```
print("\nLanguage Detection Results Summary")
print("=" * 50)
```

- Prints a summary header to the console.

python

```
detected_languages_langdetect = df[CNST_TEST_LANGDETECT_LANG_LINE_NAME].value_counts()
detected_languages_langid = df[CNST_TEST_LANGID_LANG_LINE_NAME].value_counts()

print("\nMost Common Languages Detected (LangDetect):")
print(detected_languages_langdetect)

print("\nMost Common Languages Detected (LangID):")
print(detected_languages_langid)
```

- **Prints the most common detected languages:** The `value_counts()` function counts how many times each language is detected by `LangDetect` and `LangID`.

python

```
fig, axes = plt.subplots(2, 2, figsize=(16, 12))

# Pie chart for LangDetect
axes[0, 0].set_title("LangDetect - Detected Languages", fontsize=14)
detected_languages_langdetect.plot.pie(
    autopct='%1.1f%%',
    startangle=90,
    ax=axes[0, 0],
    colors=sns.color_palette("pastel"),
)
axes[0, 0].set_ylabel('')
```

- **Pie chart for LangDetect:** Creates a pie chart of the languages detected by `LangDetect`. It shows the percentage of each language detected.

python

```
# Pie chart for LangID
axes[0, 1].set_title("LangID - Detected Languages", fontsize=14)
detected_languages_langid.plot.pie(
    autopct='%1.1f%%',
    startangle=90,
    ax=axes[0, 1],
    colors=sns.color_palette("pastel"),
)
axes[0, 1].set_ylabel('')
```

- **Pie chart for LangID:** Similarly, creates a pie chart for the languages detected by `LangID`.

python

```
# Bar chart for LangDetect confidence
langdetect_mean_probs = df.groupby(CNST_TEST_LANGDETECT_LANG_LINE_NAME)[CNST_TEST_LANGDETECT_PROB_LINE_NAME].mean()
langdetect_mean_probs.sort_values().plot.barh(ax=axes[1, 0], color="skyblue")
axes[1, 0].set_title("LangDetect - Mean Confidence Score", fontsize=14)
axes[1, 0].set_xlabel("Mean Confidence")
```

- **Bar chart for LangDetect confidence:** A horizontal bar chart showing the mean confidence score for each detected language in `LangDetect`.

python

```
# Bar chart for LangID confidence
langid_mean_probs = df.groupby(CNST_TEST_LANGID_LANG_LINE_NAME)[CNST_TEST_LANGID_PROB_LINE_NAME].mean()
langid_mean_probs.sort_values().plot.barh(ax=axes[1, 1], color="skyblue")
axes[1, 1].set_title("LangID - Mean Confidence Score", fontsize=14)
axes[1, 1].set_xlabel("Mean Confidence")
```

- **Bar chart for LangID confidence:** Similar to the previous one, but for `LangID`.

python

```
plt.tight_layout()
plt.savefig("Language_Detection_Visualization.png")
plt.show()
```

- **Final adjustments and saving:** Adjusts the layout, saves the visualization as a PNG file, and displays it.

python

```
print("\nCharts saved as 'Language_Detection_Visualization.png'.")
print("Detailed results are saved in 'Language_Detection_Results.csv'.")
print("Any differences are saved in 'Detection_Differences.csv'.")
```

- Prints messages to indicate the results have been saved and provide file names.

## Main Workflow

python

```
file_path = filedialog.askopenfilename()
with open(file_path, 'r', encoding='utf-8') as file:
    lines = file.readlines()
```

- Opens a file dialog and asks the user to select a file. Then, it reads the contents of that file line by line.

python

```
langdetect_lang_results = []
langdetect_prob_results = []
langid_lang_results = []
langid_prob_results = []
raw_text = []
```

- Initializes empty lists to store the results of language detection.

python

```
for line in lines:
    line = line.translate(str.maketrans('', '', string.punctuation)).rstrip()
    raw_text.append(line)
```

- Iterates over each line of the file, removes punctuation, and strips any trailing whitespace. The cleaned line is stored in `raw\_text`.



python

```

langdetect_lang, langdetect_prob = detect_language_with_langdetect(line)
langdetect_lang_results.append(langdetect_lang)
langdetect_prob_results.append(langdetect_prob)

langid_lang, langid_prob = detect_language_with_langid(line)
langid_lang_results.append(langid_lang)
langid_prob_results.append(langid_prob)

```

- For each line, it detects the language using both `LangDetect` and `LangID`, then appends the results (language and confidence) to the corresponding lists.

python

```

write_data_to_csv(raw_text, langdetect_lang_results, langdetect_prob_results, langid_lang_results, langid_prob_results)
extract_differences(CNST_LANG_DETECTION_RESULTS_FILE_NAME)
show_results_for_non_tech_users(CNST_LANG_DETECTION_RESULTS_FILE_NAME)

```

- After processing all lines, it writes the results to a CSV file, extracts the differences between `LangDetect` and `LangID` results, and shows the results through visualizations.

---

This code essentially provides an interactive way to load a text file, detect the language of each line using two different language detection libraries, store the results, and present them in an accessible visual format for non-technical users.