Yash Shah

Pocket Dict Mini Translator: English to French Pocket Dictionary/Mini Translator

Brief Description

Yes.

The aim of this application is to provide users with quick translations of small phrases and expressions. The core functionality of the application will allow users to input text and receive possible translations for that expression should they exist in the data. Some extra features that may extend the functionality of the app are filtering results by word category, tense (for verbs) or gender (for nouns).

Datasets and Tools

The core implementation will rely on using datasets with English text and their respective French translations.

Tools such as kaggle.com and datasetsearch.research.google will be useful in gathering relevant and sufficient amounts of data.

Data files on Kaggle are usually in CSV form hence using Python libraries such as `pandas` will help read and organise the data. Furthermore, VSCode provides an extension to visualise CSV files as a normal table in comparison with other IDEs which display CSV files as plaintext. Hence, I will be able to inspect the data manually to see if there are things I need to take into account.

I may also use a another dataset (in the form of a `.txt` file), which I will attempt to merge with the other data creating one single dataset. Although this may cause problems with duplicate values.

I also aim to make a User interface for the application, most likely using the `Tkinter` library which would make the User experience better as supposed to a command line application.

Leave until you have the rest working

General Implementation Outline

The Application will need to do the following:

- Read a Datasets

In [8]:
```
import pandas as pd
import numpy as np

data = pd.read_csv("./eng_-french.csv")
data = data.rename(columns={'English words/sentences': 'EN',
                            'French words/sentences': 'FRE'})
```

- Parse it into 2 separate list, one with English text and the other with French text.

In [9]:
```
en = data.EN
fr = data.FRE
```

- Implement a basic search algorithm to find possible matches in the English list.
- Store the index of a possible match in a list to be referenced later as English and French lists will be the same length, the index for an item in the English text list will correspond with the time in the French text list.

In [10]:
```
### A posible approach ###

# Simple linear search - not efficient given worse case is O(n)
# Time complexity: O(n)
# Space complexity: O(1)
def find_matches(search):
    results = [] # index of matching searches
    # Search algorithm
    search = search # clean `search` string

    # Use for loops instead of list comprehension to make the code more rea
    # each string needs to be cleaned hence for loops would be better (unle
    for i in len(en):
        em_text = en[i] # clean `en_text` string
        if search in em_text:
            results.append(i)

    return results
```

O(n) is actually OK.

- If possible, filter the results by word category, tense or gender if the user requires them to be.

In [11]:
```
# test with hard coded string before changing to user input
indices = find_matches("hello")

for index in indices:
    french_text = fr[index]  # clean `french_text`

    # add filtering for french text here: Nouns, verbs, word category
    # and/or
    # display results on screen
```

Testing

I aim to test the core functionality of the application by using assertions and the `pytest` library. Additionally, inspecting a subset of data manually to predict expected behaviour will also be useful.

In [ ]:
```
import pytest

def test_find_matches():
    assert find_matches("hello") == # expected list of indices
```

One problem that I may encounter is testing the UI (should I have time to implement the UI).

A possible UI test may look like the following. However, considering I have not used Tkinter, I am not sure if things like accessing a labels text through a property will work.

In [ ]:
```
def test_display():
    label = # get data from on screen label
    assert label == "Expected Output"
```

Great project but
be warned that translation is a
difficult topic.
Simple word mappings are Ok but it rapidly
gets very complicated after that.
So be sure to get something working
soon as you can, then iteratively improve it

Not much for me to add more than that
as you seem to have a comprehensive
grasp of the project and its challenges.
Good luck building your project.