FARMER'S FRIDGE

Code Exercise

Goal: Build an application that performs simple frequency analysis on a user-supplied text document.

EVALUATION

You do not need to complete all of the steps but what is completed should run and demonstrate the desired functionality. Spend enough time on the exercise to be able to speak through your solution and enough code to demonstrate that your solution would function. We will evaluate your solution for how it addresses the problem and how well you can describe your approach. You will be sharing your screen through Zoom to walk us through the exercise, Zoom whiteboard will also be available. Be prepared to answer questions to expand on your solution (a favorite question is how you would scale your solution to handle a large volume of requests).

STEPS

1. For this exercise we created a document with a made up language - it isn't necessary to understand the grammar rules of this language to complete the exercise. Ignore any non-letter characters in the document.

2. The application should accept a text document (a sample document has been provided) from the user, count how often each word is used in it, and report the top 25 most frequently used.

3. Exclude stop words from your counts. Allow the user to decide whether to exclude stop words from their analysis. A file of stop words has been provided.

4. In order to make the results more useful, the analysis should allow the user to choose to extract the stems of the words so that different inflections of the same word are all counted in the same bucket. The rules can be found at the bottom of the page.

5. **BONUS** - Not all words that end with the given suffixes are root words, come up with an algorithm to help determine when the resulting word is an actual word and not a word that happens to end in those letters.

6. **BONUS** - Save the most recent 10 frequency analyses (original text, stop words setting, and resulting word frequencies), allowing the user to navigate back to view a previous

analysis for comparison. These persisted analyses should survive a restart of the server process.

Write your own code, instead of using an existing code library or project

Use whatever language and framework(s) you find familiar and appropriate for the task. When you're satisfied with your solution, please submit:

- The code (Github repo is acceptable)
- Include output files from the application (it might be helpful to include output from each step of the exercise)
- README an overview of your solution,libraries/frameworks used, setup instructions and execution instructions
- Bonus: Submit as a zipped git repo, so we can see your commit history.
- The hours spent on the exercise

Grammar Rules

- The following are common suffixes in the language, remove the suffix to find the root word -
    - “L”
    - “LZ”
    - “EVM”
    - “ZQ”

- These suffixes require adding back letters to determine the root word -
    - remove “ZL” add “A”
    - remove “PZL” add “AZ”
    - remove “EZL” add “R”

Definitions

Stop Word - A stop word is a word that is filtered out before analysis because it may skew the results.

Stem Word - For this exercise a stem word is the word without a suffix. As a English example “bike” would be the stem word of “biker”.

Root Word - Same as Stem Word

Inflection - An inflection changes a word’s tense or case. In another English example “es” is the inflection in the word “classes”.