Assignment 04: Text Analysis

CS101 - Intro To Computer Science

Fall 2023

In this assignment, you will implement a text analysis tool that counts the number of verbal ticks. In contrast to previous assignments, you must implement certain tasks using different methods.

Prerequisites:

- Methods
- Arrays

The files for this assignment are in a zip file called `04_text_analysis.zip`. There are four groups of files in the folder

- `TextAnalysis.java`: the base implementation of the assignment - includes comments with the functions you need to implement.
- `TextAnalysisTest.java`: unit tests - make sure you pass these unit tests before submitting the assignment.
- `.jar` files in the `lib` folder: libraries to enable the use of the unit tests.
- `trump_speech_010621.txt`: An example text file to run your `main` function on.

1. Project Setup

The zip file contains a folder called `04_text_analysis`, which in turn contains a VS Code project. Extract the folder from the zip file and open it with VS Code to begin implementing and running your assignment. The folder contains the skeleton code for your submission, as well as further instructions on the various methods you need to implement, in `src/edu/nyu/assignment4/TextAnalysis.java`.

Note that this skeleton code already contains a `Scanner` called `scanner`. Use this one in all the methods.

2. Text Analysis

The program must be able to open any text file specified by the user and analyze the frequency of verbal tics in the text. Since there are many different kinds of verbal tics (such as "like", "uh", "um", etc) the program must ask the user what tics to look for. A user can enter multiple tics, separated by commas. Any spaces entered by the user before or after each tic must be ignored. You can assume that all tics are single words.

We will do this task by implementing a variety of small methods to accomplish this larger task. Every method you need to implement is described in `TextAnalysis.java`. The `@param` and `@return` statements in the comments indicate what the expected parameters and return type should be. If you correctly implement these methods your code should "pass" the unit tests in `src/edu/nyu/assignment4/tests/TextAnalysisTest.java`.

A description of how to check if your code passes unit tests is later in the document.

After the program is fully implemented, it should allow the user to submit a text file and a sequence of tics separated by commas and it should return the following statistics:

- the total number of tics found in the text
- the density of tics (proportion of all words in the text that are tics)
- the frequency of each of the verbal tics
- the percentage that each tic represents out of all the total number of tics

This example shows suggested input/output of such a program, using a file `trump_speech_010621.txt` that is contained in the assignment zip file.

```
What file would you like to open?
trump_speech_010621.txt
What words would you like to look for?
uh,like, um,so

.............................Analyzing text.................................

Total number of tics: 31
Density of tics: 0.01

.............................Tic breakdown.................................

uh        / 0                 / 0% of all tics
like      / 17                / 55% of all tics
um        / 0                 / 0% of all tics
so        / 14                / 45% of all tics
```

3. Unit Tests

This assignment includes unit tests to make sure that your code conforms to our expected set of methods. Unit tests are used for a variety of reasons, but two key reasons are what we are doing here, to define how code should be invoked, and also to make sure that code continues to perform consistently over time.

The unit tests will only build and run AFTER you have defined the various methods required for this assignment. Unit tests can be run by following the instructions in Figure 1. Note that these unit tests are VERY similar to how your assignments are graded - the autograder we use is implemented by running similar unit tests and giving marks if you pass them.

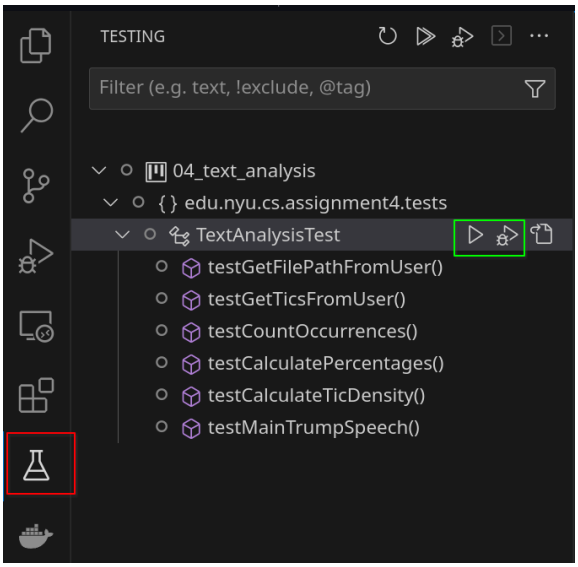

Figure 1: The red square indicates the Test tab and the green square indicates how to run or debug the tests. If you highlight the individual you should see play/debug buttons appear next to them. If you press those buttons you will run just a single unit test.

4. Debugging Your Code

Debugging is an important part of coding. In this assignment, each function has some corner cases, make sure you understand what's going on and fix your code. You can simply click the button with the bug symbol to debug the code. Remember to set a checkpoint before you run your code!

Here is a summary of a few things you need to care about. For the corner cases. Please read the comments carefully to find more details.

- `Always check if the input is null if it is possible. Do not trust users.`
- `Users' input could contain repeated tokens, ignore the repeated ones.`
- `Users' input could have no token, let them try again until you get valid input.`

- `Java allows programmers to divide a number over double 0, but you need to avoid it.`
- `When you iterate an empty String array with "for", be sure you won't go into the loop.`

5. Submission

Submit the following files:

- `TextAnalysis.java`