Hamlet word counts project

Arjun Chandrasekhar

Overview

In this project you will write code that can analyze which words occur the most frequently in a given corpus of text. This is valuable for many natural language processing algorithms, as well as data visualization tools such as word clouds. In order to make the project difficulty worthy of CS 2, you will have to do some extra text parsing. But the main point of the project is for you to see how to use the Dictionary ADT to store key-value data.

Compiling, running, and testing your code

Use the following command to compile and run your code:

```
bash run.bash
```

Use the following command to check if your code passes all the unit tests:

```
gradle test
```

Collaboration policy

For this assignment, you must work with a partner. If there are an odd number of people, then I will allow one individual to complete the project alone. You must work with someone in the same section as you. You must with a new partner - you may not work with someone that you have worked with on a previous project.

For the coding portion of the assignment, all group members should contribute equally. The best way to ensure this is to use a technique called pair programming. In this method, one person is the “driver” and one person is the “navigator”. The navigator is responsible for telling the driver what changes to make to the code, but the driver is responsible for actually typing in the code. **Importantly, you should frequently switch roles.**

The assignment report must be completed individually, with no collaboration. You may (and frequently should) discuss how to design your program, but you may not collaborate on how to answer the questions in the report. You may ask for help with the report from the instructor or the tutor(s); it will be considered an honor code violation if you collaborate with literally anyone else (including your partner) for the report.

You may not look at another group's code, and you may not show your code to another group. You may discuss the assignment with other groups, and you may discuss your code in general terms, but any sort of direct code sharing is strictly prohibited.

At the end of your report you will need to describe what help you got. In particular, you will need to list the following information:

- If you got help from someone else in the class, state who.
- If you used any internet websites for help, you must state each website that you used. Note that if you use AI technology to complete the assignment, you may automatically be given an F in the course.

Counting words

You are given a very small piece of starter code. Your task is straightforward, but definitely not simple. You want to write a program that can go through a text file, count how often each word appears, and then print out a "histogram" that visualizes how often each word appears. Here is some information about the structure of the file:

- The file starts with some metadata about the text corpus.
- The file will have a special line with the text `**START OF TEXT**`
- The words that you want to count start after that.
- Some words will have punctuation. You will need to strip away the punctuation.

When you print the histogram, it should be ordered as follows:

- For each word, you should print out that word, followed by a vertical bar, followed by one asterisk for each time that the word appeared.
- The most frequently occurring words should occur first.
- If two words are tied for frequency, they should be ordered alphabetically.
- All of the vertical bars should line up, so that all of the asterisks start from the same output column.

For example, suppose we have the following text file:

```
This is a famous line from a Shakespeare play, because it is a classic
example of
the concept of iambic pentameter.

**START OF TEXT**

a horse!
a horse!
a kingdom for my horse!
```

You would want your code to produce the following output:

```
      a| ***
  horse| ***
    for| *
kingdom| *
     my| *
```

Here are some other example input files and their corresponding output:

```
This is the text of Robert Frost's poem "Fire and Ice", written in 1920
about the anxiety
surrounding post WWI society.

**START OF TEXT**

Some say the world will end in fire,
Some say in ice.
From what I've tasted of desire
I hold with those who favour fire.
But if it had to perish twice,
I think I know enough of hate
To say that for destruction ice
Is also great
And would suffice.
```

```
          i| ***
        say| ***
       fire| **
        ice| **
         in| **
         of| **
       some| **
         to| **
       also| *
        and| *
        but| *
     desire| *
destruction| *
        end| *
     enough| *
```

```
     favour| *
        for| *
       from| *
      great| *
        had| *
       hate| *
       hold| *
         if| *
         is| *
         it| *
       i've| *
       know| *
     perish| *
    suffice| *
     tasted| *
       that| *
        the| *
      think| *
      those| *
      twice| *
       what| *
        who| *
       will| *
       with| *
      world| *
      would| *
```

```
This is the fight song for the University of Alabama Crimson Tide football
team. This song was written in 1926. Read more about it here!
https://news.ua.edu/2018/09/how-a-football-championship-gave-us-yea-alabama
/

**START OF TEXT**

Yea, Alabama! Drown 'em Tide!
Every 'Bama man's behind you,
Hit your stride.
Go teach the Bulldogs to behave,
Send the Yellow Jackets to a watery grave.
And if a man starts to weaken,
That's a shame!
```

```
For Bama's pluck and grit have
Writ her name in Crimson flame.
Fight on, fight on, fight on men!
Remember the Rose Bowl, we'll win then.
So roll on to victory,
Hit your stride,
You're Dixie's football pride,
Crimson Tide, Roll Tide, Roll Tide!!
```

```
      on| ****
    tide| ****
      to| ****
       a| ***
   fight| ***
    roll| ***
     the| ***
     and| **
 crimson| **
     hit| **
  stride| **
    your| **
 alabama| *
    bama| *
  bama's| *
  behave| *
  behind| *
    bowl| *
bulldogs| *
 dixie's| *
   drown| *
      em| *
   every| *
   flame| *
football| *
     for| *
      go| *
   grave| *
    grit| *
    have| *
     her| *
      if| *
      in| *
```

```
 jackets| *
     man| *
   man's| *
     men| *
    name| *
   pluck| *
   pride| *
remember| *
    rose| *
    send| *
   shame| *
      so| *
  starts| *
   teach| *
  that's| *
    then| *
 victory| *
  watery| *
  weaken| *
   we'll| *
     win| *
    writ| *
     yea| *
  yellow| *
     you| *
  you're| *
```

Code to write

You've been given two starter files called `Dictionary.java` and `WordCounter.java`.

`Dictionary.java`

This file contains an implementation of the Dictionary ADT. It has the following methods:

- `void put(K key, V value)`: Adds a key-value pair to the dictionary.
- `V remove(K key)`: Removes a key-value pair from the dictionary by key. Returns the value associated with the key, or null if the key does not exist.
- `V get(K key)`: Retrieves the value associated with the specified key. Returns the value if the key is found, or null if the key does not exist.
- `boolean containsKey(K key)`: Checks if the dictionary contains a key.
- `int size()`: Returns the number of key-value pairs in the dictionary.

- `boolean isEmpty()`: Returns true if the dictionary is empty.
- `Collection<K> keyList()`: Returns a Collection of all the keys in the dictionary.
- `Collection<V> valueList()`: Returns a Collection of all the values in the dictionary.

It contains an `ArrayList`-based implementation of the dictionary ADT, but you only need to know what methods it implements rather than how they are implemented.

`WordCounter.java`

There are two methods that you need to fill in.

`computeWordCounts` method

This method takes as input a `String` specifying the name of the text file to be read in. It reads through the file and it returns a `Dictionary` object. The keys of the dictionary are the words in the file. The values in the dictionary are the frequencies of each of the words.

`wordCountHistogram` method

This method takes a `Dictionary` object as input. The keys of the dictionary are words, and the values are the word frequencies. It returns a `String` containing the word count histogram. **The histogram needs to be formatted *exactly* as described above.**

- Words should be sorted by count, and if two words are tied for the same count they should be sorted alphabetically.
- Every word should have enough leading whitespace on its line so that the vertical bars line up.

You may write your own sorting code, or you may use any built-in methods that Java has for sorting. Both of these approaches have their own pros and cons. Either way, be prepared to explain your approach in the project report. **You must justify why your sorting uses on average $O(n \log n)$ comparisons, where N is the number of words/frequency pairs.**

Test input files

We have included four input files: `Iambic-Pentameter.txt`, `Fire-and-Ice.txt`, `Alabama-Fight-Song.txt`, and `Hamlet.txt`. The first three are pretty simple and straightforward (and small enough that they were included above). The last one contains the entire text of Hamlet.

For all of the files, there is some metadata about the text at the top of the file. All of this metadata can be skipped. Each file then has the following line:

```
**START OF TEXT**
```

This line signals the point where we should start counting words. However, there are some nuances for counting words:

Punctuation

You need to strip away punctuation - with one exception. Apostrophes (the `'` character) will often be a part of the word, and in fact the word can start and/or end with an apostrophe.

You will also see text like this `Horatio.--Welcome,`. You are responsible for making sure your program can recognize that the two relevant words are `Horatio` and `Welcome`.

Case-sensitivity

If the same word appears twice but with different capitalization, we should treat them as the same word. For example, `the` and `The` should not be counted as different words.

Hamlet character names

The file `Hamlet.txt` contains a play script, so for every line it will list the character who said the line. You will need to ignore these words. Fortunately, all of the character names are written in `ALL CAPS`.

Hamlet narrations

There are some parts of the text that do not contain dialogue but rather describe the characters moving into and out of the scene. Text like this:

```
[They exit.]
```

```
[Flourish. Enter Claudius, King of Denmark, Gertrude the
Queen, the Council, as Polonius, and his son Laertes,
Hamlet, with others, among them Voltemand and
Cornelius.]
```

These are not part of the dialogue of Hamlet, and they should be ignored.

Hamlet Acts and Scenes

There are lines that denote the start of an act or the start of a scene:

```
ACT 1
```

```
=====
```

```
Scene 1
=======
```

You should ignore these when you are counting word frequencies.

Testing your code

We have provided JUnit tests for `Iambic-Pentameter.txt`, `Fire-and-Ice.txt`, and `Alabama-Fight-Song.txt`. **Your need to provide one more text file and write your own tests in the same style as the ones provided.** Your text file should contain at least 25 words. You should manually count up the words, determine what the correct Dictionary values and word count histogram should be, and write tests accordingly in the exact same style as the ones given to you.

Analyzing a text of your choosing

In addition to the text file that you provide for testing, provide another text file that is from the real world and that represents something important or meaningful to you. It could be a famous speech, the script of a movie/play, the lyrics to your favorite song, etc. Run your code to generate a histogram for that text file. **In your project report, write a paragraph discussing whether you find anything interesting in the word frequencies for that text.** See if you can use computer science to translate quantitative observations into qualitative analysis about the meaning behind the text.

Note that since the project report is done individually, you and your partner will both upload different text files and use your (shared) code to analyze the texts separately.

Project Report

Answer all of the following questions. Make your answers correct (duh), complete, and *concise*. **Don't forget that each person must complete the report separately and independently. You may NOT collaborate with ANY other student (including your project partner) on the report.**

Put all of the original questions in bold. *Put your answers to the questions in italics.* Make it easy for me to give you credit for answering all of the questions. An easy way to accomplish this is to copy/paste all of the questions into your own document, so that you can keep track of which ones you have and haven't answered.

If you are submitting a rework, put your modified answers in red.

1. (1 point) What is a dictionary?
2. (4 points) What are the runtimes for `put`, `remove`, `get`, and `containsKey`, in an `ArrayList`-based implementation of the Dictionary ADT?
3. (1 point) How did your program skip through the metadata at the top and know when to actually start reading words? Specifically describe what kind of loop you used to do this.
4. (1 point) How did your program strip away irrelevant punctuation symbols?
5. (1 point) How did your program make sure not to treat different capitalizations of the same word as different words?
6. (1 point) How did your program recognize character names that were not actually part of the dialogue, and ignore these?
7. (1 point) How did your program manage to ignore narration delineated by square brackets? Keep in mind that some of the narrations span multiple lines, so you should be specific about how you managed to ignore the entire narration.
8. (1 point) How did your program recognize and skip past the lines of text denoting acts?
9. (1 point) How did your program recognize and skip past the lines of text denoting scenes?
10. (10 point) Describe the approach you used to sort the words by frequency and break ties alphabetically. If you wrote your own sorting algorithm, describe which one you used, and how you modified it to deal with key-value pairs as opposed to a list of atomic elements. If you used built-in Java methods describe what you had to do to structure your data so that it could fit into the built-in java methods. If you created any custom classes or data structures for sorting purposes, describe them here. Make sure to clearly explain how your program knows to first compare key-value pairs by word frequency, and only use alphabetical comparisons as a tiebreaker. Finally, justify why your sorting approach uses on average $O(n \log n)$ key-value pair comparisons.
11. (1 point) How did you make sure that every line in the histogram `String` had the correct amount of horizontal padding at the start, to make sure that all of the vertical bars lined up?
12. (10 point) Describe any helper methods that you wrote. State the name and purpose of each helper method. If you did not write any helper methods then your code is almost certainly not modular enough, and you should rethink whether some of your code can be refactored into separate methods.
13. (10 points) Describe the real-world body of text that you uploaded, and discuss all of the interesting and/or unexpected patterns that you found in the word frequencies for that

body of text. Try to focus on patterns that help us quantitatively understand the qualitative meaning behind the text.

14. (1 point) Was there anything that this project helped you understand better that you did not fully understand when we covered it in class?
15. (1 point) Were there any unexpected obstacles that you encountered on this project that were not related to the complexity of the concepts that were covered and the tasks that you had to complete? Do you have suggestions for how I can modify the project writeup or delivery to make sure these obstacles don't impede future students?
16. (1 point) What lingering questions do you have? Is there anything about this project that you would like to go over in class or in student hours?
17. (1 point) Describe the division of labor that you had with your partner. Explain which parts of the code each person contributed on their own, and explain which parts of the code you completed together.
18. (1 point) If you got help from any individual other than your partner or the instructor, state who helped you and describe the nature of the help that they provided.
19. (1 point) If you got help from any websites, list each of them here.
20. (1 point) Type out the honor code followed by your name (or explain why you cannot put your name to the honor code for this assignment).

Put your reflection in a document titled `Hamlet-Word-Counts-Report.pdf`.

Grading rubric

`computeWordCounts` **method:** 30 points

`wordCountHistogram` **method:** 20 points

Unit testing: 15 points
- **Designing manual test for a small corpus**: 10 points
- **Providing a real text that is meaningful to you:** 5 points

Commenting: 10 points

Programming style: 5 points

Project report: 50 points