Project 1b: Search Engine

CS Y11

Monday 18th February, 2019

CLICK HERE FOR FILES NEEDED

You are provided with the full code for WordPair.java and SearchEngineTester.java. You do not need to modify WordPair. You will need to implement the following methods in **SearchEngine.java.** The intended use for this class is:

1. Read in the words from a file.
2. Strip out any common words (i.e., the, a, an) (the file with common words is given to you).
3. Display the most occurring words in the input file to another output file or on console.

1 Part I

1.1 Getting words from the input file

void getWordsFromFile(String filename) throws IOException

This method constructs an ArrayList containing WordPairs for each word in the file. Wordpairs contain two instance variables. One is "word" which is a string of the word. The other is "count" which is the current count of this word, how many times its been found. This starts at 1. This class throws IOException because you will deal with file I/O in this method. You have to handle that with try/catch Your algorithm will be:

```
for every word in the file
    search for the word in the arrayList
        if the word is present
            increment its count
    if word is not already in the arrayList
        add word to the arrayList
```

Important:
Make sure that you do not ignore punctuation while saving your words in WordPair. For example,

- "Apple" and "Apple!" should be counted as different words in the ArrayList (with different entries)
- Similarly "state", "state," and "state." should be counted as different words.

(Although this may not be logically correct, it will make it much simpler for you to implement)

Note:

- Adding a word into the ArrayList should be ***case insensitive*** and should just keep the String in the same as its first occurrence.
- All the input files should be put in the same location as the .java files.

To test: Whether reading the file succeeded. Try writing your own small files. Make sure the SearchEngine's ArrayList contains the correct WordPairs read.

1.2 Remove a given set of common words

void removeCommon(String omitFilename) throws IOException

This method will read **each word** from the specified file and remove that word from the ArrayList. Your method need not be efficient (nested for loops is fine). Also, removing a word is case insensitive like in the method before.

Things to Test: Make sure that the ArrayList has been properly filtered to now exclude common words.

2 Part II

2.1 Print Words

public void printWords(ArrayList wordList, boolean printToFile) throws IOException

This method is called to print the words in the WordPair Arraylist. This method takes in an ArrayList of WordPairs and a boolean printToFile. If printToFile is false, it should print the ArrayList of WordPair with their counts on the screen. If printToFile is true, it should output the ArrayList of WordPair with their counts to a file myOutput.out. If it exists, it should be overwritten.

Note: Print a newline after printing all the words in the console or the file.

Things to Test: make sure the correct action is chosen for the corresponding printToFile boolean. Then, make sure the content is correct. It is possible for the Tester method to open the file that was just written (through the printWords method call). Make sure to check for consistency between the wordList and the printed words. You may need to verify the print statements to terminal manually.

2.2 Find n most frequent words

public ArrayList topNWords(int n, int length)

This method will find the top n occurring words whose lengths are at least "length" (i.e the value of the integer "length" passed as the parameter) , as determined by their frequency in the arraylist and returns this list of words as well as their counts as an ArrayList. In the event of a tie, use the first occurring word with that count.

public ArrayList topNWords(int n, char beginsWith)

This method will find the top n occurring words which begin with the character 'beginsWith' (i.e the value of the character beginsWith passed as the parameter) [**Comparison should be case insensitive**] , as determined by their frequency in the arraylist and returns this list of words as well as their counts as an ArrayList. In the event of a tie, use the first occurring word with that count.

Hint: Here is one way to get the top n words from an unsorted Arraylist (we refer to it as the original list here). You can iterate through the list looking for the word with the highest count. Once you have saved this word and its count into another ArrayList, you can make its count negative of its original in the original list so it wont be the one with the highest rank again in the next round of search. You repeat this process until you have located the top n words. When you are done finding the top n words in the original list, you will need to iterate through the original array list again and change the negated counts back to their original values. For instance, if the original arraylist is the following (with word and its count) and the length parameter passed is 1.

```
dog     5
cat     222
snake   10
dolphin 200
```

We want to locate the top 2 words in the list (n=2). In the first round of search, your code will locate cat which has the highest count, and make its count negative of its original count int the array list. It becomes

```
dog     5
cat     -222
snake   10
dolphin 200
```

In the second round of search, the list becomes the following.

```
dog     5
cat     -222
snake   10
dolphin -200
```

The last step is to go through the original array list again to make all the counts positive again. So the array list changes back to its original form.

You are required to be able to have the method topNWords execute more than once and produce the same results (ArrayList should have the same values). In the case of a tie (two words are equally frequent), you should select the word which occurred first in the original text file.

Note 1: You are free to implement any other helper algorithm/method as well to get the topNwords.

Note 2: Please note that topNWords method does not print/display anything.

Note 3: If the number of words that satisfy a criteria (i.e begin with given character or length >= given length) are less than n, return all those words. For example n = 10 but you have only 5 words that satisfy given criteria, return those 5 words.

2.3 Finding the word count

int findWordCount(String word)

This method will take a string and search for it in the ArrayList. If the word is found, its count is returned. If the word isnt found, then 0 is returned. Also, searching for a word is **case insensitive** like in the methods removeCommon and getWordsFromFile.

Things to Test: Think about what needs to be tested to ensure correctness.

3 Testing

You can run the main method in SearchEngineTester.java to run your program. You are free to change anything and add more tests to this class.

4 Possible Extensions

1. Create a nice output message that explains how this program works.
2. Edit the output so that it uses colored letters.
3. Find lots of other txt files and give the use options of what to read.
4. Anything you think would be great or that you came up with through receiving feedback from others!

Acknowledgments to UCSD for this project idea and materials.