Shiho Numakura

3/9/2018

CS 150

Professor Plotnick

Project 2: Creating a Searchable Collection

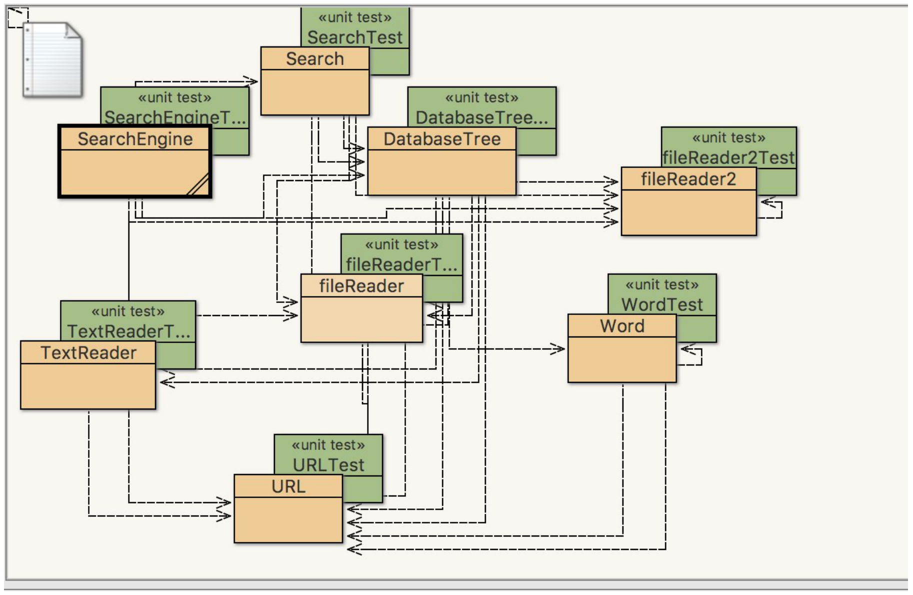
Introduction:

This project will analyse a file with multiple websites and create a database. Then, users can perform a search to print out the URL in order appropriately to the search. The goal of this project is to efficiently build a database and operate a reasonable search engine. In this process, we are to use either a treemap or a treeset along with a priority queue. When a search is performed, we assumed that there will only be one "or" as the keyword connector, representing "a disjunction between the clauses on the left and right" (*Plotnick,* 2). We also restricted that each keyword must be separated by a connector including "and," so the program will print out the websites that contained the most keywords. User is also allowed to have a "-" symbolizing a negation to that word. This negation has the highest priority so the url containing this word will have the lowest priority. Consequently, this program is a miniature search engine.

Approach:

The program will first go through the first input file which contains all websites for our database. Then, it will read the text file which represent each website's article. The second input file contains all "common words" such as "and", "the", and "is" which will not be stored in the database, for it is apparent that most articles will contains these words. All words in the articles are then stored as individual words disregarding repeating and the specified common words.

Finally, once the database is built, the program will ask the user for a search input. According to this user input, the program will print out the resulting URLs in order of its priority. The priority of this order depends on the number of matching words to the search and each website's reliability.



Approach; figure 1.

A class diagram for my project

As shown in Approach; figure 1, my program is made up of 8 classes. To begin coding, I first created a URL class which represents each website and stores its appropriate details in as instance variables. When calling for an URL object, the URL, reliability of its website, and the text file that the article is stored in, should be specified in its parameter. It will take in these parameters and assign to its instance variables. So, when the fileReader class is called given with a String of a file in its parameter, it can create an ArrayList of URL objects that it reads from its first input file taken in *Main()*.

To read the second input file taken from *Main()*, I created the fileReader2 class. In its parameter, it will take a String of a file and create a LinkedList of Strings with words specified in its file. So in this case, it will take in the file containing the common words and store all words in a LinkedList.

However, to finally build a database, I need to go through the actual articles given from the first input file. To do so, I created the TextReader which will take in a URL and create a LinkedList of Strings with all the words in it. However, in this process, it will eliminate repeating words or common words. So, the second parameter, a LinkedList of common words is necessary as well. In the process of reading the file using a Scanner (Class Scanner), we only wanted to take in the words excluding any punctuations such as periods or commas and even exclamation points. To do this, the syntex ‘replaceAll("[^a-zA-z]", "")’ (Class Pattern) was important.

Finally, now that all needed data are analyzed to create a database, I created a simple class called Word. This Word class will simply store a String and a ArrayList of URL in its instance variable. Then, I began to code the DatabaseTree class. In creating this database, I decided to create a TreeMap (Class TreeMap<K,V>) so the key would be the word itself, and the value being an ArrayList of URL objects that contain that key. The reason I decided not to use a TreeSet (Class TreeSet<E>) was because I wanted to store all the URLs for each word in one location. In its parameter, should have a fileReader object and a fileReader2 object of the input files. So using the ArrayList of URLs retrieved by the fileReader object, a TextReader object is called and adds the words into the TreeMap. When adding to the TreeMap, it makes sure the word isn’t already in the tree. If the word is already in the tree, it will simply take the URL of that LinkedList of String it currently went through and adds into the ArrayList in the Word object with the appropriate word. However, when the word isn't in the tree, it simply calls a Word object and assign the URL to its ArrayList.

Now that all database is built, I began to work on my Search Class. This class will take in the DatabaseTree to perform the search, and a String of what the user has typed in to search. To compute what the user has inputted, there is a case where the user used the connector "or". If there are no "or" connector in the search, we simply operate a search of each keywords using the database. However, if there is a "or" connector used in their search, it will perform two different searches for the clause on the left and right of the connector. Then, it will bring the list of URLs together and print out accordingly. This searching process became especially complicated when each URL's priority had to be appropriately changed for every keyword in the search. Therefore, I first went through the keywords without a negation since negation has the highest priority among the keywords to search. Then, while every URL already has an appropriate numerical priority less than one accordingly to its reliability of the url, for every keyword that was in that url, I added one to its priority. So, the whole numbers in priority shows the number of keywords that the article contains. However, once priority of the URLs for all the regular keywords are done, the program will go over all the urls that contain the keywords with a negation and puts it back to its initial priority when it only stores its reliability priority. This algorithm will allow all keywords with a negation to have a lower priority than others but still print at the end according to its website priority. When there is an "or" connector, this process is done separately for the two clauses then combined, taking the higher priority if there was an overlap in these result.

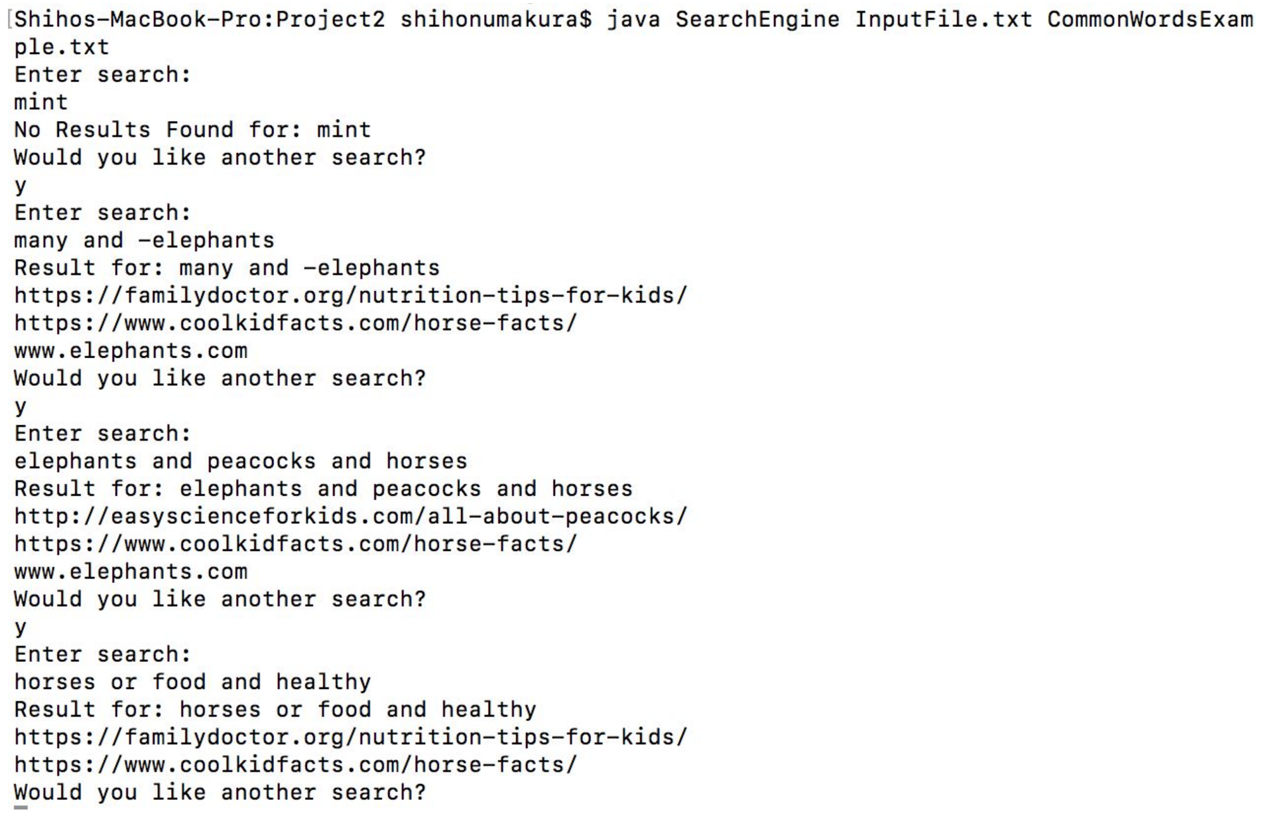
Once the finalized ArrayList of URLs are ready to be printed, they are put into a PriorityQueue (Class PrioirtyQueue<E>) to print in order. This will automatically sort the URLs according to the priority set for the search. It will then print out at most the first five results of the search into the console.

Methods:

To run this program, it takes in an input file of the websites to build a database, and another file containing all the common words to disregard. A DatabaseTree object is first created, then asks for the user for a search input. Taking this search input, it calls a Search which will print out the results. The program will keep asking the user for another search and as long as the user answers "y", they can continue searching. However, any other answer than a "y" will close the program.

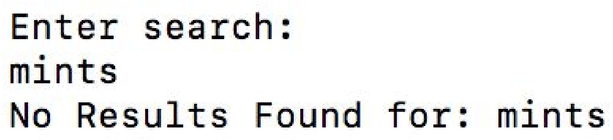
Data and Analysis

Data and Analysis; Figure 1 : result for continuous searches



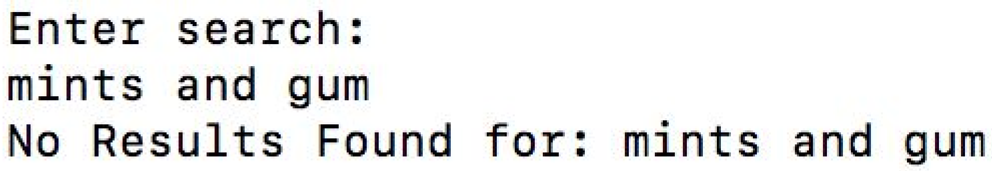
```
[Shihos-MacBook-Pro:Project2 shihonumakura$ java SearchEngine InputFile.txt CommonWordsExam
ple.txt
Enter search:
mint
No Results Found for: mint
Would you like another search?
y
Enter search:
many and -elephants
Result for: many and -elephants
https://familydoctor.org/nutrition-tips-for-kids/
https://www.coolkidfacts.com/horse-facts/
www.elephants.com
Would you like another search?
y
Enter search:
elephants and peacocks and horses
Result for: elephants and peacocks and horses
http://easyscienceforkids.com/all-about-peacocks/
https://www.coolkidfacts.com/horse-facts/
www.elephants.com
Would you like another search?
y
Enter search:
horses or food and healthy
Result for: horses or food and healthy
https://familydoctor.org/nutrition-tips-for-kids/
https://www.coolkidfacts.com/horse-facts/
Would you like another search?
```

Data and Analysis; Figure 2.1 : result for a search with no results with five files



```
Enter search:
mints
No Results Found for: mints
```

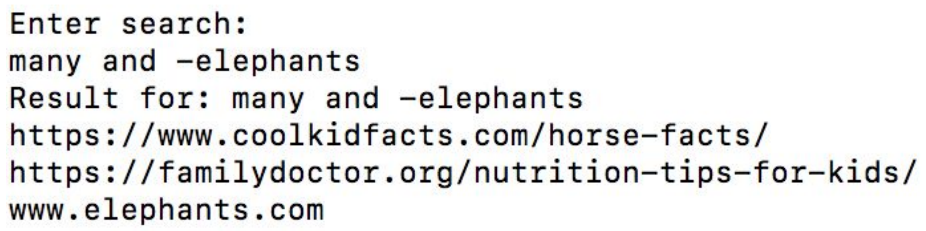
Data and Analysis; Figure 2.2 : result for a search with no results with five files



```
Enter search:
mints and gum
No Results Found for: mints and gum
```

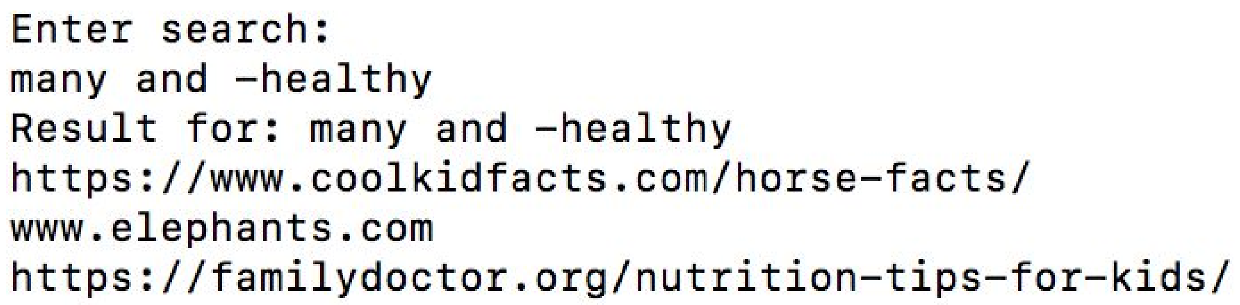
Figures 2.2 shows that no matter how many keywords are inputted, it will print the same result.

Data and Analysis; Figure 3.1 : result for a search with at least one negation with five files



```
Enter search:
many and -elephants
Result for: many and -elephants
https://www.coolkidfacts.com/horse-facts/
https://familydoctor.org/nutrition-tips-for-kids/
www.elephants.com
```

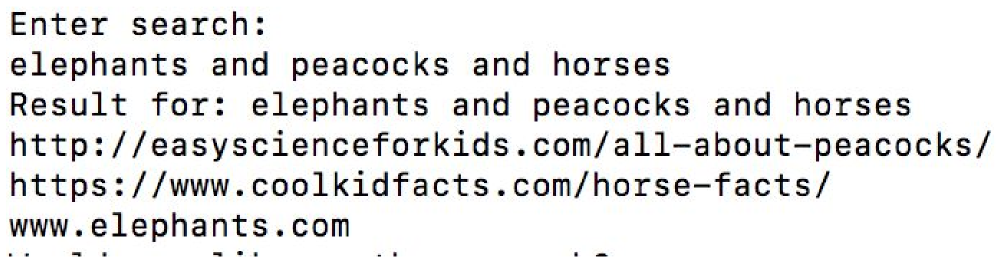
Data and Analysis; Figure 3.2 : result for a search with at least one negation with five files



```
Enter search:
many and -healthy
Result for: many and -healthy
https://www.coolkidfacts.com/horse-facts/
www.elephants.com
https://familydoctor.org/nutrition-tips-for-kids/
```

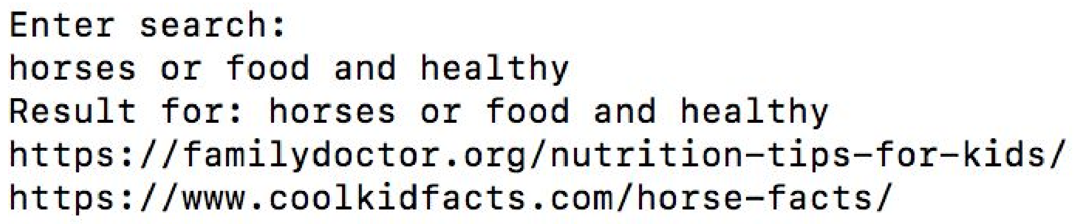
As shown in figure 3.2, when the keyword “healthy” is negated, the result shows the website containing that word at the end of the list.

Data and Analysis; Figure 4 : result for a search that uses at least three keywords with five files



```
Enter search:
elephants and peacocks and horses
Result for: elephants and peacocks and horses
http://easyscienceforkids.com/all-about-peacocks/
https://www.coolkidfacts.com/horse-facts/
www.elephants.com
```

Data and Analysis; Figure 5 : result for different combinations of connectors with five files



```
Enter search:
horses or food and healthy
Result for: horses or food and healthy
https://familydoctor.org/nutrition-tips-for-kids/
https://www.coolkidfacts.com/horse-facts/
```

As shown in Figures 1-5, the results are shown in order appropriately to the search input.

Conclusion:

Throughout this process, I realized the complexity Google company is facing to analyze the data and deliver web users the appropriate information. Although this program runs successfully with the given space and data, to think to compute in a bigger scale would be unimaginable. However, it is interesting how data can be stored in many different ways. For this project, I used a TreeMap but I am also aware the numerous ways to code one program is infinite. Overall, my program has no bug and runs successfully.

References :

(2017, December 19), Retrieved April 16, 2018, Class ArrayList<URL>, fromhttps://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

(2017, December 19), Retrieved April 16, 2018, Class LinkedList<E>, from https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html

(2017, December 19), Retrieved April 16, 2018, Class Pattern, from https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html

(2017, December 19), Retrieved April 16, 2018, Class PriorityQueue<E>, from https://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html

(2017, December 19), Retrieved April 16, 2018, Class Scanner, from https://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html

(2017, December 19), Retrieved April 16, 2018, Class TreeMap<K,V>, from https://docs.oracle.com/javase/7/docs/api/java/util/TreeMap.html

(2017, December 19), Retrieved April 16, 2018, Class TreeSet<E>, from https://docs.oracle.com/javase/7/docs/api/java/util/TreeSet.html

Plotnick, L. *CS150: ProjectII - Creating a Searchable Collection*