# CS231 Project 8

## Summary

The goal of this project was to find the most common words in a text document of reddit comments and find trends related to how many times a set of words appeared over 8 years of reddit comments. To find the most common words, counts files that include each word and how often it appeared were found in Project 7 using a binary search tree and were read into a binary search tree again, then put into a priority queue. The priority queue removes items in order of how many times the word appears and prints the list to the terminal. In an extension, it reads the word-value pairs directly into a priority queue. Then to find trends, it reads the word-value pairs into a binary search tree and the finds each word in a list and returns its frequency (how many times it appears divided by the total number of words in the document) and prints a list of how many times it appears to the terminal. Then, a graph is made with the results. In extensions, it prints the results to files rather than printing to the terminal. The most common words found are pronouns articles and very simple verbs like is and have. Cat and dog are the most common pet words over all 8 years tested.

## Tasks

1) The first task was to create a FindCommonWords task that will print the words in a word count file (from project 7) in order from the highest number they occur to the least number of times they occur. It has fields for a word counter object (from project 7) and a priority queue heap that uses a comparator that takes in key value pairs that have a string (the word) and an integer (the count) and returns the difference between the counts to find which word occurs more times.

In findWords, I used the readWordCountFile of WordCounter from Project 7 to put all of the word-count pairs into a binary search tree. Then using the getPairs method, I made an arrayList of the word-count pairs. Then I looped over every pair in the arrayList and add them to the priority queue heap. Then it loops over the number of items in the heap and prints out the word-count pair returned when removed is called. This will print every word in the word count file in order from the word that occurs most to the word that occurs least. Here is an image of the ten most common words in the 2008 reddit comments file:

```
Key: the, Value: 715631
Key: to, Value: 433159
Key: a, Value: 383959
Key: i, Value: 350554
Key: of, Value: 331367
Key: and, Value: 320678
Key: that, Value: 290574
Key: it, Value: 277506
Key: is, Value: 259969
Key: you, Value: 246681
```

"the" is by far the most common word, almost twice more common that even the second most common word.

"deleted" is the first non pronoun, article, auxiliary or linking verb, word that appears in the list, the 27th occurring 74,414 times.

2) The second task was to create a findTrends class to "determine the frequency of a specific set of words in a set of words". It has the same fields as the findCommonWords class.

I made a findTrends method that starts by clearing the word counter object in case more than one list of words is analyzed with the same FindTrends object. Then it calls the readWordCountFile method of the WordCounter object. It makes an arrayList of Doubles called frequency and puts 0.0 in each location up to the size of the words arrayList passed in to the method. Then for each word in the arrayList, it calculates the frequency by calling the getFrequency method of the Word Counter object and then sets the index of the word in the frequency array list to the frequency of the corresponding word in the words array list. At the end it returns the frequency array list.

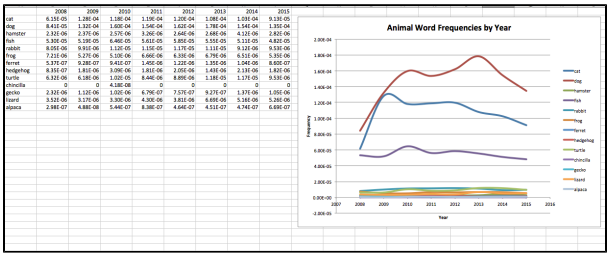
Then I used a secondary helper method called findTrend with an array of strings (for the args array) and an array list of words. It starts by making an arrayList frequency of word by Year with empty strings the length of the number of words in the words arrayList. Then looping over the 1st index of the args array up to and including the 2nd index of the args array, it calls findTrends from above on the current year file (args[0]i.txt). Then it loops over that arrayList and sets the corresponding index in the frequency of word by year to its current value plus the index of the findTrends arrayList. At the end, I print the ArrayList.

I put in a list of animals (cat, dog, hamster, fish, rabbit, frog, ferret, hedgehog, turtle, gecko, lizard, alpaca). Here is the terminal output with the frequencies from the reddit comments in 2008-2015:

```
kaylas-mbp:Project8 kaylafreeman$ java FindCommonWords counts_reddit_comments_ 2008 2015 cat dog hamster fish ra
bbit frog ferret hedgehog turtle chincilla gecko lizard alpaca
cat,6.1519118049524444E-5
dog,8.405228338161769E-5
hamster,2.3248503914064468E-6
fish,5.299466661436747E-5
rabbit,8.0475590471761636E-6
frog,7.212997368209746E-6
ferret,5.365039364784108E-7
hedgehog,8.3456167896641695E-7
turtle,6.318824140745727E-6
chincilla,0.0
gecko,2.3248503914064468E-6
lizard,3.5170813613584171E-6
alpaca,2.980577424880068E-7
cat,6.1519111804952444E-5,1.2818406245149004E-4,1.1806974447537388E-4,1.1891344486613104E-4,1.1966649029762222E-4,
1.0780911402197008E-4,1.0255984153580734E-4,9.126657046724159E-5
dog,8.405228338161769E-5,1.3221193720001334E-4,1.5969100177892724E-4,1.5352579156331634E-4,1.6204684560817018E-4
,1.7038561990956078E-4,1.5440822331915196E-4,1.3463013732536818E-4
hamster,2.3248503914064468E-6,2.3679021248894566E-6,2.5712780755083193E-6,3.2597631945027354E-6,2.638085517890018
3E-6,2.680418667367403 4E-6,4.121342361946993E-6,2.8192229140007568E-6
fish,5.299466661436747E-5,5.187414448855767E-5,6.455371298511943E-5,5.607245439432830 4E-5,5.847756231322874E-5,5
.548717147868201 5E-5,5.111411963840005 6E-5,4.816581310522488 5E-5
rabbit,8.0475590471761636E-6,9.911013017578549E-6,1.1225823793072986E-5,1.1454445669572111E-5,1.1724824523955637E
-5,1.1072383653985305E-5,9.119062122698084E-6,9.532817177075759E-6
frog,7.212997368209746E-6,5.272854216248686E-6,5.100746751414877E-6,6.655349855443084E-6,6.326519899384395E-6,6.
788723914547552E-6,6.513615801927718E-6,5.351757011691654E-6
ferret,5.365039364784108E-7,9.276317602659725E-7,9.407114910396 29E-7,1.4487836420012157E-6,1.2213358879120454E-6
,1.3527346545592907E-6,1.8421785283004348E-6,8.601038054504444E-7
hedgehog,8.345616789664169E-7,1.8064407963074202E-6,3.0938955705303353E-6,1.8109795525015196E-6,2.051844291692232
62E-6,1.4278865798125846E-6,2.1317288079036167E-6,1.8157747003953824E-6
turtle,6.318824140745727E-6,6.176074614402396E-6,1.0222398202630635E-5,8.443692163538335E-6,8.891325263999691E-6
,1.1823902906518244E-5,1.1653450816539 77E-5,9.532817177075759E-6
chincilla,0.0,0.0,0.0,4.180939960176129E-8,0.0,0.0,0.0,0.0,0.0
gecko,2.3248503914064468E-6,1.1229226571640 72E-6,1.0243302902431515E-6,6.791173321880699E-7,7.572282505054682E-7
,9.268737447906251E-7,1.3737807873156641E-6,1.0512379844394 32E-6
lizard,3.5170813613584171E-6,3.1734770745941164E-6,3.3029425685391416E-6,4.3010764371911094E-6,3.8105679702855 82E
-6,6.688521347543 16E-6,5.163528090255427E-6,5.256189922197 16E-6
alpaca,2.980577424880068E-7,4.882272422452487E-8,5.435221948228967E-7,8.375780430319528E-7,4.641076374065773E-7,4
.509115515197636E-7,4.737175128674704E-7,6.689696246145 68E-7
kaylas-mbp:Project8 kaylafreeman$ java FindCommonWords counts_reddit_comments_ 2008 2015 cat dog hamster fish ra
bbit frog ferret hedgehog turtle chincilla gecko lizard alpaca
```

It shows the animals in just 2008, then in 2008-2015.

I copied and pasted this into an excel document, added a header, then made a graph of the results:

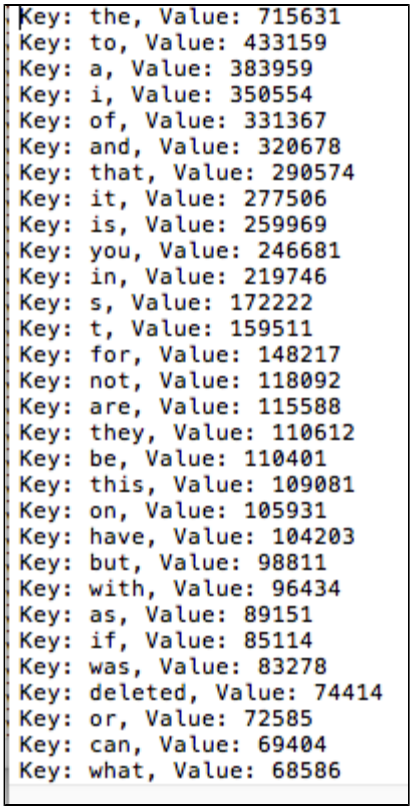



Dog, cat, and fish are by far the most common animal names uses throughout all 8 years. The three words are closest in 2008 then vary but generally grow apart until 2015. Dog and cat become generally more frequent between 2008 and 2015, while fish remains constant. Cat almost catches up to dog in 2009, but not quite. The rest of the words are all much less common and remain constant in their frequentness over then 8 years.

## Extensions

1) The first extension that I completed was to move the findCommonWords output to as text file rather than printing to the terminal so it is easier to look through and understand. To do this, I made a new method called writeFindWordsFile. It starts by clearing the WordCount object, then calling readWordCountFile of the WordCount object. It makes an ArrayList of the pairs then adds each to the priority queue heap. Then I made a BufferedWriter object and in a try loop, I made a new bufferedWriter object and looped over the number of items in the priority queue heap, and writes the key value pair returned by removing from the priority queue heap). Then it catches exceptions and will close the file when necessary.

Here is the text file made using this method:




(I tried to upload the file but its too big)

2) The second extension that I completed was to try to make the findWords process faster. I made another method that would create a text file, called findWords that requires an integer as a parameter to distinguish it from the first. It starts by initiating a String object and a BufferedReader object. In a try loop, it makes a BufferedReader object, then uses it to read the next line of the file (to eliminate the header). Then it reads each line of the word counts file and parses it, making the two pieces into a KeyValuePair object, then adding each KeyValuePair to the priority heap. Then it checks for exceptions and closes the file. In findWords, it prints as it removes, in writeFindWordsNewFile, it makes a text file using this new method, so then it writes the removes to a text file just like the previous extension did. I made sure that the file created was the same as above:

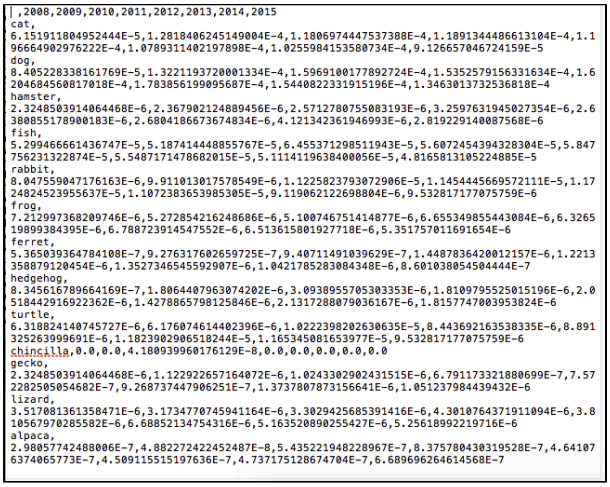
```
Key: the, Value: 715631
Key: to, Value: 433159
Key: a, Value: 383959
Key: i, Value: 350554
Key: of, Value: 331367
Key: and, Value: 320678
Key: that, Value: 290574
Key: it, Value: 277506
Key: is, Value: 259969
Key: you, Value: 246681
Key: in, Value: 219746
Key: s, Value: 172222
Key: t, Value: 159511
Key: for, Value: 148217
Key: not, Value: 118092
Key: are, Value: 115588
Key: they, Value: 110612
Key: be, Value: 110401
Key: this, Value: 109081
Key: on, Value: 105931
Key: have, Value: 104203
Key: but, Value: 98811
Key: with, Value: 96434
Key: as, Value: 89151
Key: if, Value: 85114
Key: was, Value: 83278
Key: deleted, Value: 74414
Key: or, Value: 72585
Key: can, Value: 69404
Key: what, Value: 68586
```

Then, to see which worked best, I ran them on the same file (2008) and kept track of how long it took to do each:

```
kaylas-mbp:Project8 kaylafreeman$ java FindCommonWords counts_reddit_comments_20
08.txt counts_reddit_comments_2008_times.txt
Directly Into PQHeap: 39268
Make Binary Tree: 38652
kaylas-mbp:Project8 kaylafreeman$ java FindCommonWords counts_reddit_comments_20
08.txt counts_reddit_comments_2008_times.txt
Directly Into PQHeap: 38254
Make Binary Tree: 38343
kaylas-mbp:Project8 kaylafreeman$ java FindCommonWords counts_reddit_comments_20
08.txt counts_reddit_comments_2008_times.txt
Directly Into PQHeap: 38889
Make Binary Tree: 39104
```

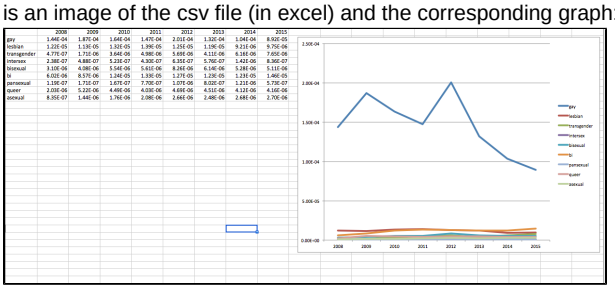
The two options actually take approximately the same amount of time, so not much was saved by adding KeyValuePairs directly into the priority queue heap.

3) The third extension that I completed was to write the results of findTrend to a CSV file so it can be opened as a CSV file. I edited findTrend so that at the end I initialized a BufferedWriter object to null. In a try loop, I made a new bufferedWriter, I wrote a header by first writing the numbers in the range of the first index in the args array up to and including the second index of the args array, separated by commas. This makes a header with the list of years that the frequency information was collected at. Then I write every item in the words list followed by the corresponding string in the frequency of words by year. This makes a full csv file with a header line that can be turn into a graph. It looks like this:



```
|,2008,2009,2010,2011,2012,2013,2014,2015
cat,
6.1519118049524444E-5,1.2818406245149004E-4,1.1806974447537388E-4,1.1891344486613104E-4,1.1
966649029762222E-4,1.0789311402197898E-4,1.0255984153580734E-4,9.126657046724159E-5
dog,
8.4052283381617694E-5,1.3221193720001334E-4,1.5969100177892724E-4,1.5352579156331634E-4,1.6
2046845600817018E-4,1.7838561990956874E-4,1.5440822331915196E-4,1.3463013732536818E-4
hamster,
2.3248503914064468E-6,2.367902124889456E-6,2.5712780755083193E-6,3.2597631945027354E-6,2.6
380855178900183E-6,2.6804186673674834E-6,4.1213423619469936E-6,2.8192291400875686E-6
fish,
5.2994666661436747E-5,5.1874144488557674E-5,6.4553712985119434E-5,5.6072454394328304E-5,5.847
756231322874E-5,5.5487171478682015E-5,5.1114119638400056E-5,4.8165813105224885E-5
rabbit,
8.0475590471761636E-6,9.9110130175785496E-6,1.1225823793072906E-5,1.1454445669572111E-5,1.17
248245239556637E-5,1.1072383653985305E-5,9.1190621226988044E-6,9.5328171770757596E-6
frog,
7.2129973682097466E-6,5.2728542162486866E-6,5.1007467514148776E-6,6.6553498554430846E-6,6.3265
198993843956E-6,6.7887239145475526E-6,6.5136158019277186E-6,5.3517570116916546E-6
ferret,
5.3650393647841086E-7,9.2763176026597256E-7,9.4071149103962976E-7,1.4487836420012157E-6,1.2213
358879120454E-6,1.3527346545592907E-6,1.0421785283084348E-6,8.6010380545044446E-7
hedgehog,
8.3456167896641696E-7,1.8064407963074202E-6,3.0938955705303353E-6,1.8109795525015196E-6,2.0
518442916922362E-6,1.4278865798125846E-6,2.1317288079036167E-6,1.8157747003953824E-6
turtle,
6.3188241407457276E-6,6.1760746144023966E-6,1.0222398202630635E-5,8.4436921635383356E-6,8.891
3252639996916E-6,1.1823902906518244E-5,1.1653450816539776E-5,9.5328171770757596E-6
chinchilla,0.0,0.0,0.0,4.1809399601761296E-8,0.0,0.0,0.0,0.0,0.0,0.0,0.0
gecko,
2.3248503914064468E-6,1.1229226571640726E-6,1.0243302902431515E-6,6.7911733218806996E-7,7.57
2282505054682E-7,9.2687374479062516E-7,1.3737807873156641E-6,1.0512379844394326E-6
lizard,
3.5170813613584716E-6,3.1734770745941164E-6,3.3029425685391416E-6,4.3010764371911094E-6,3.8
105679702855826E-6,6.6885213475431676E-6,5.1635208902554276E-6,5.2561899221971676E-6
alpaca,
2.9805774248800676E-7,4.8822724224524876E-8,5.4352219482289676E-7,8.3757804303195286E-7,4.64107
6374065773E-7,4.5091155151976366E-7,4.7371751286747046E-7,6.6896962646145686E-7
```

4) The last extension that I completed was to test FindTrends on another set of words. I decided to do a list of words related to sexuality to see how their frequency has changed in the last 8 years. I used gay, lesbian, transgender, intersex, bisexual, bi, pansexual, queer and asexual. Here is an image of the csv file (in excel) and the corresponding graph:



| | 2008 | 2009 | 2010 | 2011 | 2012 | 2013 | 2014 | 2015 |
|---|---|---|---|---|---|---|---|---|
| gay | 1.44E-04 | 1.87E-04 | 1.64E-04 | 1.47E-04 | 2.01E-04 | 1.32E-04 | 1.04E-04 | 8.92E-05 |
| lesbian | 1.22E-05 | 1.13E-05 | 1.32E-05 | 1.39E-05 | 1.25E-05 | 1.19E-05 | 9.21E-06 | 9.75E-06 |
| transgender | 4.77E-07 | 1.71E-06 | 3.64E-06 | 4.98E-06 | 5.69E-06 | 4.11E-06 | 6.16E-06 | 7.65E-06 |
| intersex | 2.38E-07 | 4.88E-07 | 5.21E-07 | 4.30E-07 | 6.35E-07 | 5.76E-07 | 1.42E-06 | 8.36E-07 |
| bisexual | 3.10E-06 | 4.08E-06 | 5.54E-06 | 5.61E-06 | 8.26E-06 | 6.14E-06 | 5.28E-06 | 5.11E-06 |
| bi | 6.02E-06 | 8.57E-06 | 1.24E-05 | 1.33E-05 | 1.27E-05 | 1.22E-05 | 1.23E-05 | 1.46E-05 |
| pansexual | 1.10E-07 | 1.71E-07 | 1.67E-07 | 7.70E-07 | 1.07E-06 | 8.02E-07 | 1.21E-06 | 5.73E-07 |
| queer | 2.05E-06 | 5.22E-06 | 4.66E-06 | 6.03E-06 | 4.68E-06 | 4.51E-06 | 4.12E-06 | 4.16E-06 |
| asexual | 8.35E-07 | 1.44E-06 | 1.76E-06 | 2.08E-06 | 2.66E-06 | 2.48E-06 | 2.68E-06 | 2.70E-06 |

Gay is by far the most common word used, although its frequency has decreased since 2012. I wonder if this is because there has been a lot of talk of eliminating "gay" as an insult or because people now use other words to discuss sexuality. Lesbian is the only other word to generally decrease in frequency over the 8 years, and only very slightly. The rest of the words increase in frequency slightly, possibly because there is more knowledge and use of these words in modern culture.

5) The fifth extension I completed was to make a findWords text file for each of the reddit comment counts file (for each year from 2008-2015). Interestingly, they all had the same top 10 words, with the and i always being the top two, and the rest occasionally switching positions.

## Help

Profs: Stephanie

Students: Jos