Project 2 Report

Hiep Le – CS 150 – 04/15/2018

1. Introduction

The project aims to create a prioritized collection of web site information that can be searched by keywords input by the user. When a user enters search words, a prioritized list of matching sites is returned to them. Each keyword is to be entered as a single word. There is also a list of common words which will be excluded from every search (Ranks NL, n.d). If the user enters a common word, there will be a message indicating that the word is part of the excluded list.

Connectors (and/or) can also be used by the user to make the results more accurate and specific. Additionally, the negation sign "-" can be used to specify results without the chosen word.

For each search, the top 5 sites will be displayed, ordered by the following criteria (in priority order)

- Number of unique words that match the file/URL
- Priority of website
- Alphabetical order of name of website

After the results of the search are shown, the program will ask users if they want to make another search. If the user answers anything but "Y", the program will exit. (Plotnick, 2018)

2. Approach

Classes

The program consists of 5 main classes

- Website stores all information about the site's name, URL, priority as well as list of words. It also has methods for searching of words in its contents.

- Database is a website container, allowing us to call methods on all websites in the program. The database has methods that will run search on all sites and return set of all websites with matching results.

- Search Engine takes in user input, analyzes the input accordingly and passes the desired keywords into the database. It will print out the result that the database returns into the terminal as well as the time needed to run the search

- Input File Reader is used to read all data necessary for program function. This includes number of websites, website's details and list of excluded words.

- Controller holds the main method and is used to run the program.

In addition to the aforementioned classes, we also have 4 test classes for every class aside from Controller class. Figure 1 shows the results of the unit tests.

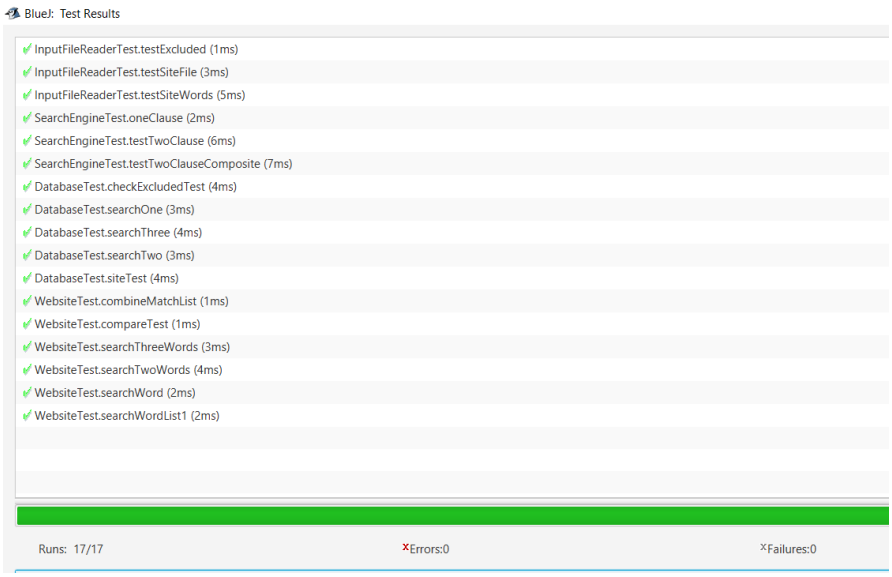

Figure 1. Unit Test Results

Design

When the program is first run, data for different websites will be imported into the database, with details specified by each input file. The program will maintain each website's name, URL and contents of its text. Each website will be tagged with a priority attribute which will determine its reliability. User can then input search words into the terminal, the words will be checked against the database and the top results will be returned. We will measure the time taken to run the search to analyze the performance of the program.

Algorithm and Data Structures

- TreeSet (Oracle, 2017) is used to store the words of each website object as well as to store the list of sites in the database. The TreeSet structure helps to avoid adding duplicate words to the list of words as it does not allow identical objects. Additionally, objects in a TreeSet are automatically sorted, which makes the addition, iteration and printing of the objects much simpler. The complexity for the basic operations (adding and contains) are O(log n), allowing our program to run very efficiently (Oracle, 2017).

- PriorityQueue (Oracle, 2017) is used to store websites returned as final results of a search. Since final results change every search and the head of the queue is guaranteed to be the best result for every search, we can keep popping and printing the head of the queue until the queue is empty and ready for the next search. Access to the head of queue is done in constant time, allowing the printing of results to be quick and efficient.

- Comparable Interface (Oracle, 2017): Website class implements Comparable interface to make websites comparable by number of words match, priority, and alphabetical name.

- Scanner (Oracle, 2017) is used to read input files as well as users' input.

3. Methods

We analyze the complexity of searches with different number of connectors and varying number of sites. This is done through measuring the runtime of the different searching methods in the Search Engine class. Searching with and without "or" are considered two different cases. As such, we will carry out the experiment for several cases, with A, B, C and D representing different possible key words:

- Single word
- A and B, A and B and C, A and B and C and D
- A or B, A or B and C, A and B or C and D

For each case, we will vary the words chosen to run the search, adding negation signs and record the average run time in nanoseconds. The number of sites will start with 5 sites and then 10, 20 and 40 sites.

The run time for each case will then be compared in graphs to determine any possible relationship.

4. Data and Analysis

Firstly, we look at the impact of adding a negation sign in a search. From Figure 2, we can see that the impact of having negation is minimal. This is because the program will still have to look at the same number of words to see if a program contains or does not contain a word. As such, performance difference is minimal.

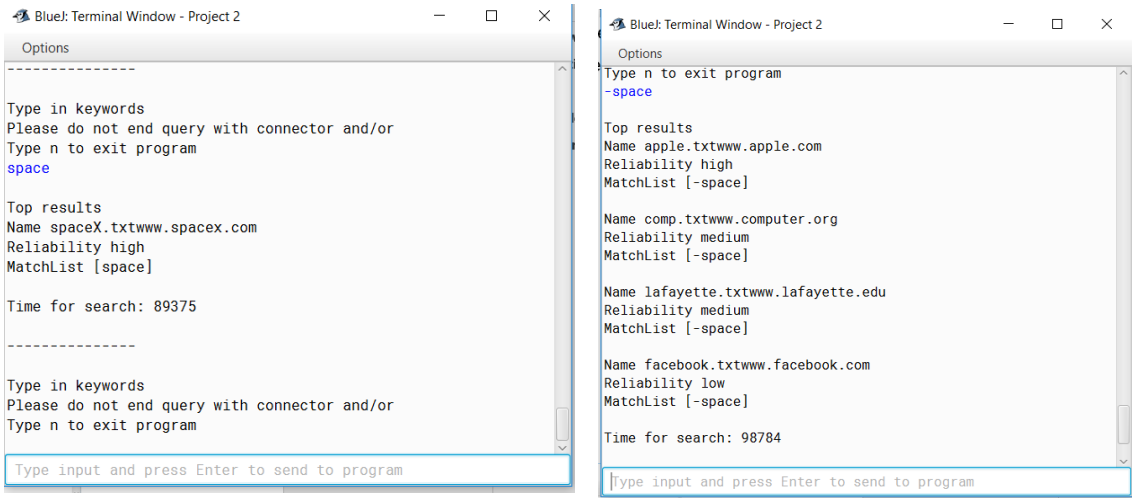

Figure 2. Search time comparison for word with and without negation

The impact of adding connectors on search time will also be analyzed. First, we consider clauses with no “or” connector.

	Search time/ ns
Single word	60574
A and B	186432
A and B and C	321583
A and B and C and D	350402

Table 1. Search time comparison with different number of “and”

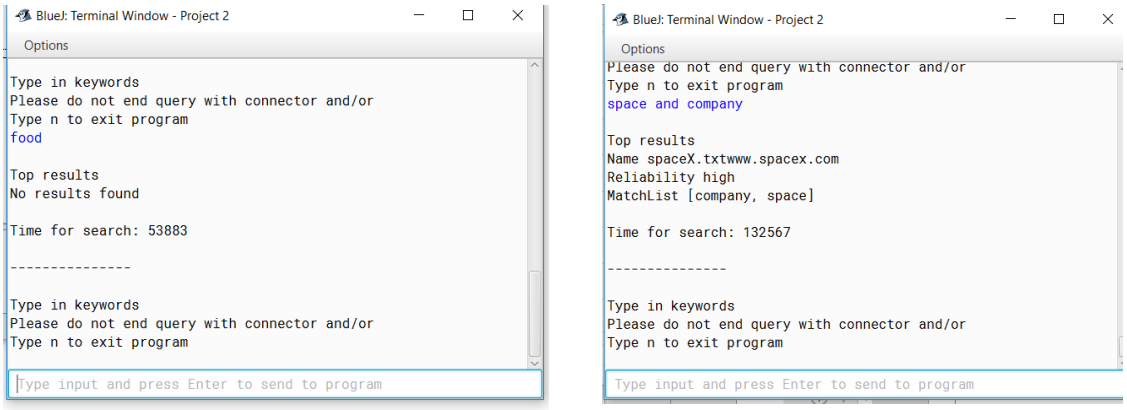

Figure 3. Screenshots of search with different number of “and”

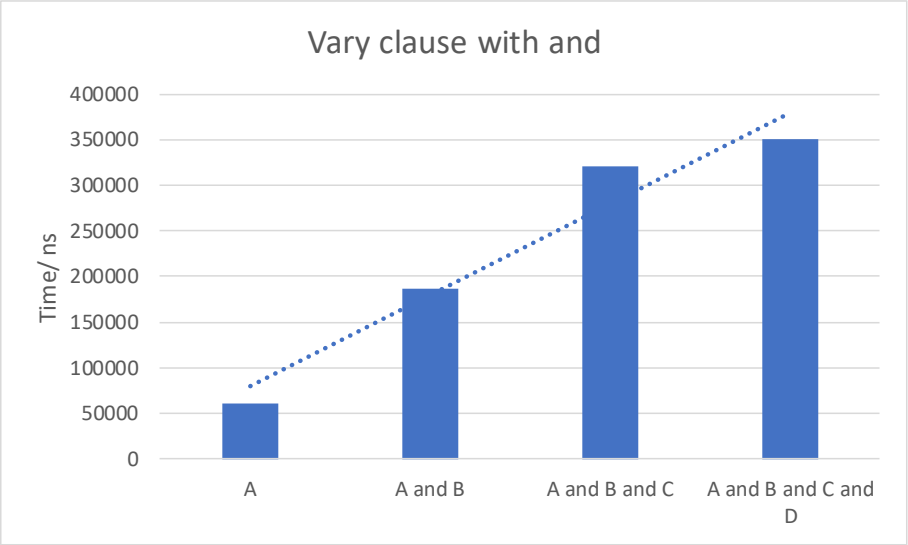

Figure 4. Search time comparison with varying number of "and"

From Figure 4, it can be seen that the search runtime increases with increasing number of "and" connectors. This is because each new keyword requires the program to iterate through the database one more time, searching for matches. The run time seems to increase linearly at O(n) complexity at first but it is difficult to have a definite conclusion as the increase in runtime from 2 "ands" to 3 "ands" does not seem to cause a linear increase in runtime anymore.

Now we consider the case when we vary the number of "and" in a statement with "or"

	Search time/ ns
Single word	60574
A or B	1070367
A or B and C	1048942
A and B or C and D	791122

Table 2. Search time comparison with varying "and" with "or"

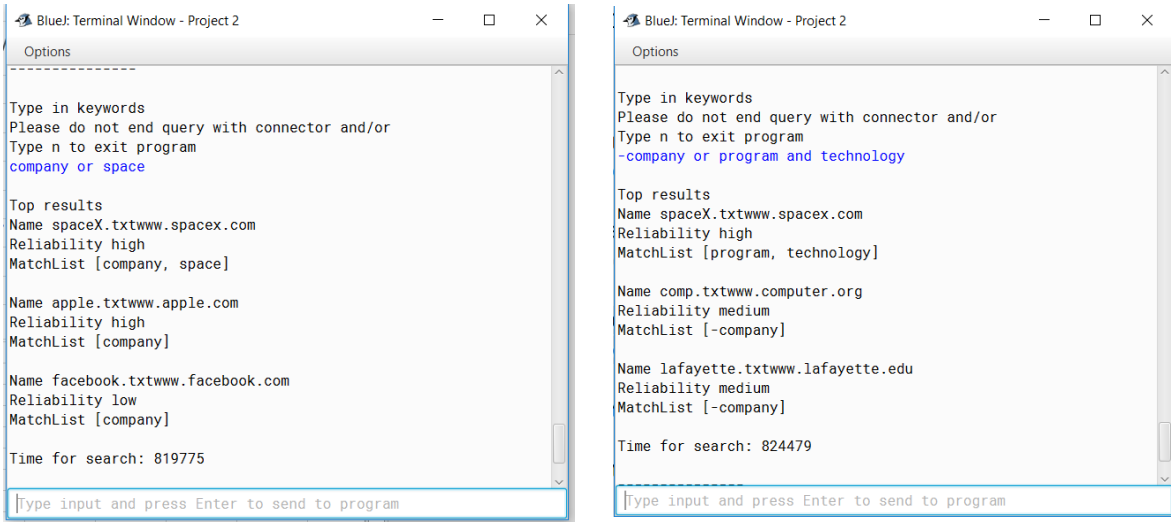

Figure 5: Screenshots of search with different number of "and" with "or"

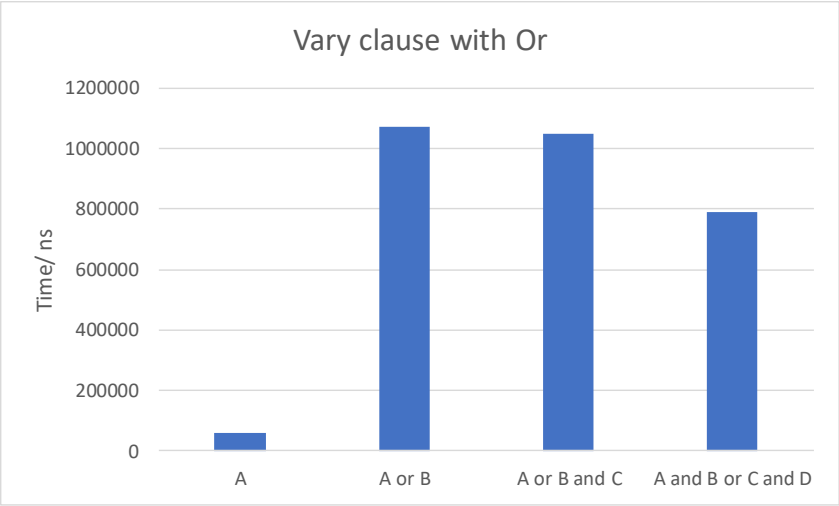

Figure 6: Search time comparison with varying number of "and" with "or"

Compared to varying "and" in search statement without "or", varying "and" in statements with "or" does not seem to produce any recognizable impact. This is possibly because the "or" dominates the "and" in terms of impact. Having an "or" requires the search program to merge two results queues together, a demanding operation. As such, any impact by the increasing the number of "and" are insignificant compared to this demanding function. In the last cases, the run time even decreases. This is because with increasing number of "and", there are fewer matching websites and thus the merge() function will have fewer websites to work with, causing a decrease in the time required for every search.

Next, we analyze the complexity of search when we vary the number of input sites.

Number of sites	5	10	20	40
Single	60574	78403	217501	282666
A and B	186432	395561	463128	1031026
A and B and C	312583	371186	433621	614938
A and B and C and D	350402	398555	521286	729972
A and B or C and D	791122	1679747	2032973	6360629
A or B	1070367	1785799	3185018	10031012
A or B and C	1049842	2166822	2878832	3605383

Table 3: Search time comparison with various number of sites

In general we observe that the search time increases with increasing number of sites. As with the previous analysis, we will split the cases into two main cases for further analysis: with and without "or".

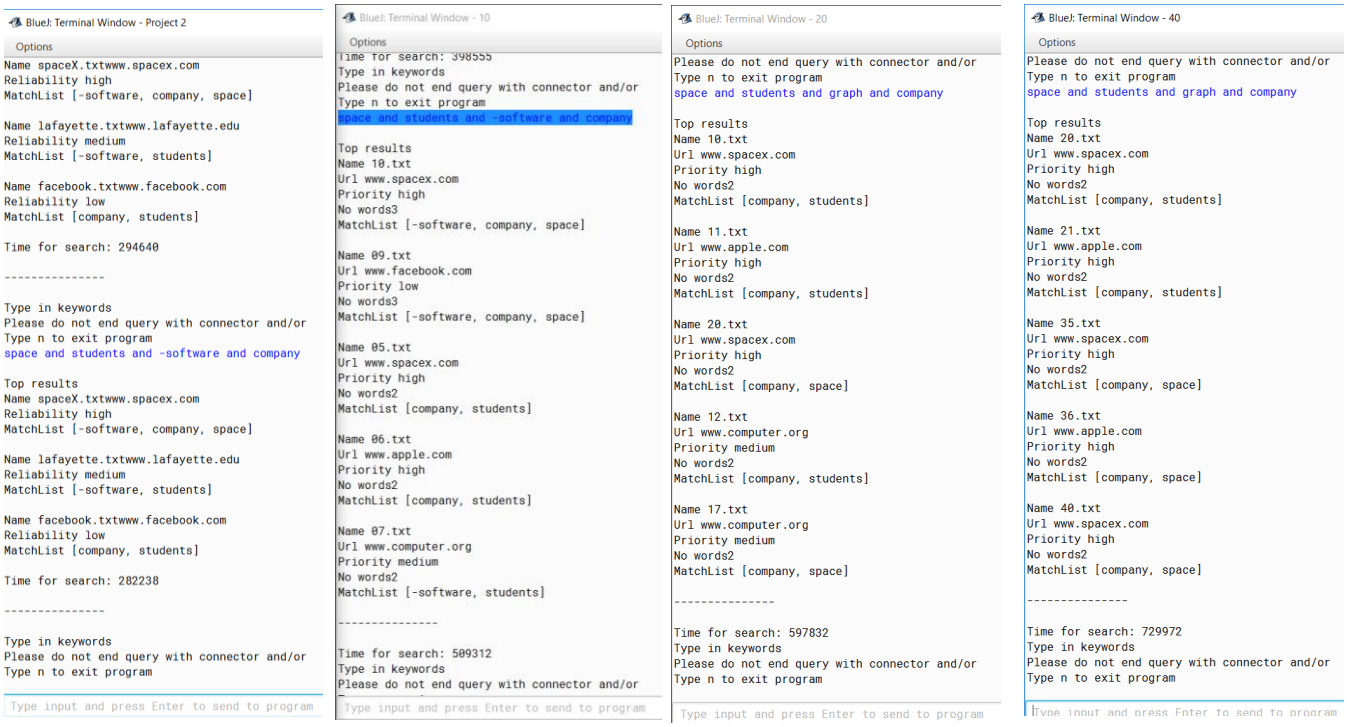

Figure 7: Screenshots of search of one case (A and B and C and D) with different input sizes

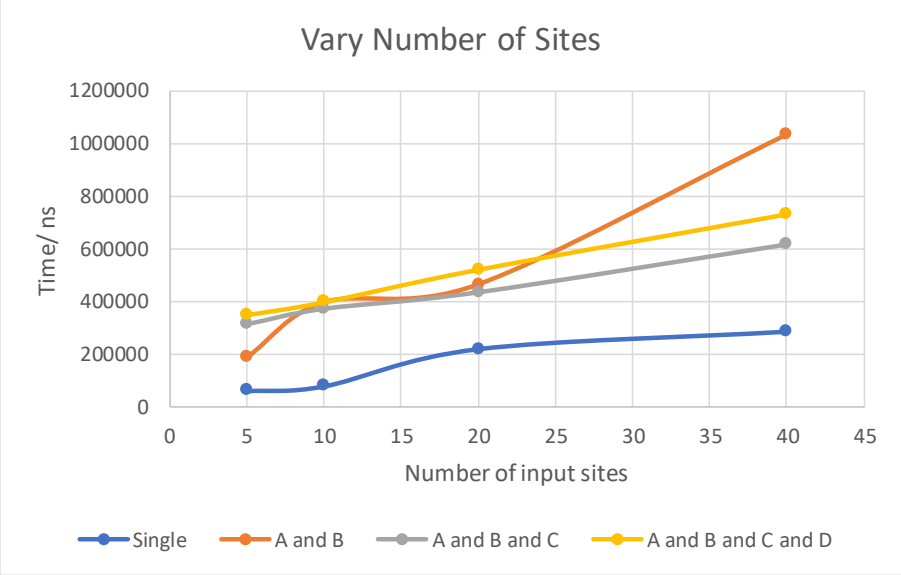

Figure 8: Search time with various input sizes in searches without "or"

Figure 8 suggests that when we vary the number of sites in searches with no "or" connector, the run time generally increases. This is understandable as the program needs more time to look through all files with increasing number of input sites. The exact complexity is difficult to evaluate since our sample size is too small and computer performance is not always constant, but a good estimate can range from O(log n) to O(n) time performance.

Among the searches with “and”, the steepest increase is in the “A and B” case, implying that this is the case where the number of sites has the most impact on. This can be because the test for “A and B” only has one matching case (Both A and B has to be in the text) compared to other cases such as “A and B and C” (which accepts A and B and C, A and B, B and C, C and A all as match cases). As such, these other cases already have to process more matches and the increase in number of sites do not cause a significant increase in the time taken.

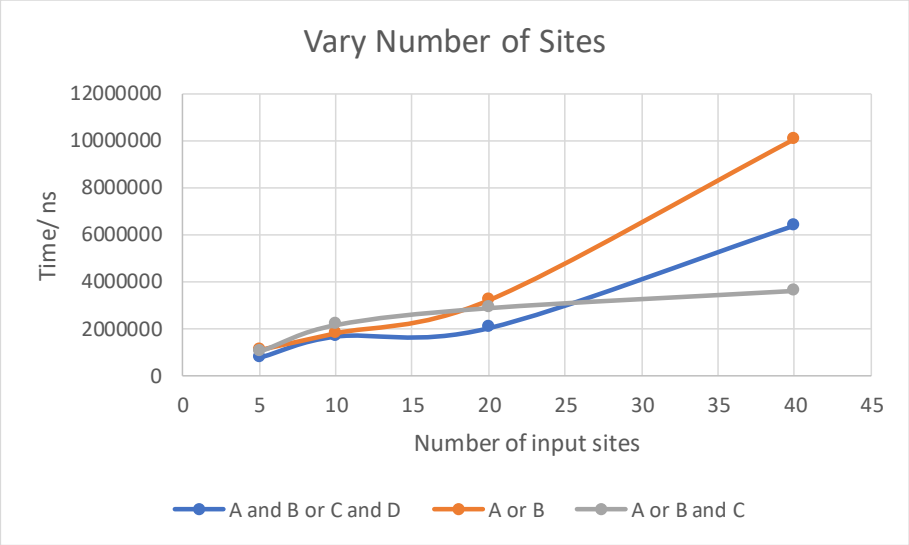

Figure 9: Search time with various input sizes in searches with “or”

Similar to figure 8, figure 9 also shows that increasing number of input sites will contribute to increasing runtime for all the searches with “or”. However, for some cases, such as “A or B and C”, the changes are minimal with each increase in number of sites. As explained previously, this can be due to the merge() method being very demanding performance-wise, thus any additional impact of an increased number of sites will not be as significant. Also, as noted above, the “A or B” case continues to take the most time. This is because the search for A or B will return the most number of matching sites, hence the merge() method will take a longer time.

5. Conclusion

In conclusion, the search runtime will depend on both the number of connectors and number of sites. As the two quantities increase, the program will require more time to return the desired results. The use of the negation site (“-“) has a minimal impact on program performance because it takes the same time to check if a website contains or does not contain a word. Both connectors “and” and “or” will increase search runtime because multiple searches are required for one user input. Out of the two, the connector “or” will significantly increase runtime because the program has to maintain two list of results for two clauses and then merge the results together.

6. References

Linda Plotnick. *CS 150: Project 2 – Creating a Searchable Collection.* (2018). Lafayette College.

Oracle. Class TreeSet, *Java™ Platform*, Standard Edition 8 API Specification. (2017). Retrieved April 15, 2018, from https://docs.oracle.com/javase/7/docs/api/java/util/TreeSet.html

Oracle. Class PriorityQueue, *Java™ Platform*, Standard Edition 8 API Specification. (2017). Retrieved April 15, 2018, from https://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html

Oracle. Interface Comparable, *Java™ Platform*, Standard Edition 8 API Specification. (2017). Retrieved April 15, 2018, from https://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html

Oracle. Class Scanner, *Java™ Platform*, Standard Edition 8 API Specification. (2017). Retrieved April 15, 2018, from https://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html

Ranks NL. *Stopwords Lists*. (n.d.). Retrieved March 29, 2018, from https://www.ranks.nl/stopwords