

BOSTON UNIVERSITY

Twitter Keyword Search Project Report

Tianyi Tang
U00278643
tty8128@bu.edu

December 12, 2019

Abstract

The goal of the project is to provide users with top relevant twitters from local or output twitter databases by input keywords. Both command line application and a website application based on Spring Boot is developed.

This document begins with a brief introduction of the problem requested, and some fundamental algorithms used in the projects. The second part is application implementation, the third part includes result from the application, the forth part includes different kinds of test cases and test results, and the last part is the summary and future work.

CONTENTS

1	Introduction	4
1.1	Objective	4
1.2	Heap Sort	4
1.3	Bucket Sort	5
2	Implementation	6
2.1	First generation: Spring Boot web application	6
2.2	Second Generation: Command Line Tool	9
2.3	Third Generation: Command Line Tool	11
3	Result	14
3.1	First Generation	14
3.2	Third Generation	18
4	Speed Test	23
4.1	Keywords with low relevance rate	23
4.1.1	Test Case 1	23
4.1.2	Test Case 2	23
4.2	Keywords with high relevance rate	24
4.2.1	Test Case 3	24
4.2.2	Test Case 4	24
4.3	Long Sentences	25
4.3.1	Test Case 5	25
4.3.2	Test Case 6	25
5	Summary	27
5.1	Work Breakdown	27

1 INTRODUCTION

1.1 OBJECTIVE

The project is about writing a keyword searching engine that takes a CSV file with date/twitter pairs as input, and provides users with top relevant twitters using the input from users. The system should return the most relevant twitters first.

There are three core parts for the projects. The first core part is to read CSV file from internal storage or from input absolute path. The second part is to read the keyword from the input. The third part is to sort twitters by relevance.

The first two parts can be easily finished by internal libraries of Java. The third one need the algorithm to finish the comparison. First of all a similarity value is needed to do the comparison. The way to generate the similarity value is split the twitter to words, and check whether the word is inside of the keywords set. After calculating the similarity, these values will be used to sort twitters. Two ways for sorting are implemented in the project. One is heap sort, and the other is bucket sort.

1.2 HEAP SORT

Heap sort can be implemented by PriorityQueue in Java and it can be used to sort.

Heap sort divides its input into a sorted and an unsorted region, and it iteratively shrinks the unsorted region by extracting the minimum element and moving that to the sorted region. The improvement consists of the use of a heap data structure rather than a linear-time search to find the minimum.[5]

Although somewhat slower in practice on most machines than a well-implemented quick sort, it has the advantage of a more favorable worst-case $O(n \log n)$ run-time. Heap sort is an in-place algorithm, but it is not a stable sort.

The Heapsort algorithm involves preparing the list by first turning it into a min heap. The algorithm then repeatedly swaps the first value of the list with the last value, decreasing the range of values considered in the heap operation by one, and sifting the new first value into its position in the heap. This repeats until the range of considered values is one value in length.

Heap sort works as follows:

- 1 Call the buildMinHeap() function on the list. Also referred to as heapify(), this builds a heap from a list in $O(n)$ operations.
- 2 Swap the first element of the list with the final element. Decrease the considered range of the list by one.
- 3 Call the siftDown() function on the list to sift the new first element to its appropriate index in the heap.
- 4 Go to step (2) unless the considered range of the list is one element.

The buildMinHeap() operation is run once, and is $O(n)$ in performance. The siftDown() func-

tion is $O(\log n)$, and is called n times. Therefore, the performance of this algorithm is $O(n + n \log n) = O(n \log n)$. [1]

The `PriorityQueue` class is a part of the `java.util` package and is a generic implementation of a priority-based queue in Java. It is based on the priority heap. The elements of the priority queue are ordered according to the natural ordering, or by a `Comparator` provided at queue construction time, depending on which constructor is used. [2]

1.3 BUCKET SORT

Bucket sort, or bin sort, is a sorting algorithm that works by distributing the elements of an array into a number of buckets. Bucket sort can be implemented with comparisons and therefore can also be considered a comparison sort algorithm. The computational complexity depends on the algorithm used to sort each bucket, the number of buckets to use, and whether the input is uniformly distributed.

Bucket sort works as follows:

- 1 Set up an array of initially empty "buckets".
- 2 Scatter: Go over the original array, putting each object in its bucket.
- 3 Sort each non-empty bucket.
- 4 Gather: Visit the buckets in order and put all elements back into the original array.

Bucket sort is mainly useful when input is uniformly distributed over a range. In the normal case the time complexity is $O(n)$. When the input contains several keys that are close to each other (clustering), those elements are likely to be placed in the same bucket, which results in some buckets containing more elements than average. The worst-case scenario occurs when all the elements are placed in a single bucket. The overall performance would then be dominated by the algorithm used to sort each bucket, which is typically $O(n^2)$ insertion sort, making bucket sort less optimal than $O(n \log(n))$ comparison sort algorithms like quick sort. [4]

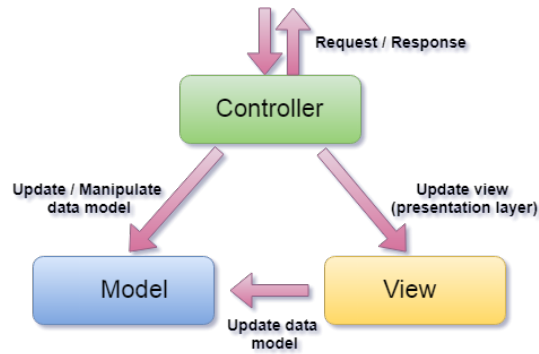


Figure 2.1: Model-View-Controller design pattern

2 IMPLEMENTATION

There are three generations of applications. The first generation is the Spring Boot website. It offers users with user-friendly and tidy user interface, but as students do not have permission to access the MySQL database on SCC(Boston University Shared Computing Cluster), this application cannot be tested on SCC and is only deployed on Amazon EC2 Ubuntu virtual machine.

The second generation is the command line application that uses bucket sort and heap sort to sort the through twitter dataset. However, every time if a user need to search the dataset, the program will iterate through the whole dataset.

To improve the speed of searching, in the third generation of application the program is storing the reversed index to the HashMap, so the searching speed can be boosted up.

2.1 FIRST GENERATION: SPRING BOOT WEB APPLICATION

The Spring Framework is an application framework and inversion of control container for the Java platform. Spring Boot is Spring's convention-over-configuration solution for creating stand-alone, production-grade Spring-based Applications.[3]

The Spring Boot is using the MVC(Model-View-Controller) design-pattern shown in figure2.1. MVC is a software design pattern commonly used for developing user interfaces which divides the related program logic into three interconnected elements. This is done to separate internal representations of information from the ways information is presented to and accepted from the user. Following the MVC architectural pattern decouples these major components allowing for code reuse and parallel development.

The most important part in MVC is the model part. Model is the central component of the pattern. It is the application's dynamic data structure, independent of the user interface. Inside of the model, the most important part is the data storage. In the industrial application,

persistent data in Spring Boot should be stored into database that Spring supported. The project chose MySQL as the solution. MySQL is an open-source relational database management system (RDBMS). MySQL can import certain types of external files directly. The project creates a database named 'tw_search', and imports CSV file into the MySQL database by 'load' SQL and terminate fields with comma, terminate lines with '\n' and ignore the first row of the CSV file. All the data from the CSV file is stored into 't_twitter' table with the first column named 't_date' and second column named 't_string'.

To read from the MySQL database, MyBatis is chose to be the persistent framework. A persistence framework is a middleware that assists and automates the storage of program data into databases, especially relational databases. MyBatis is a Java persistence framework that couples objects with stored procedures or SQL statements using an XML descriptor or annotations. The project uses the MyBatis to generate TwitterData object out of the twitter database. The Date type filed in TwitterData named twDate and String type filed named twString will automatically be fulfilled when functions in Mapper interface is called.

After loading the CSV file into the database, a temporary database is created for external files from users. 't_twitter_temp' table is created for users to store their CSV file into the system. The field inside of the temporary table is exactly the same as the table of 't_twitter', which contains only Date and String.

For the view part, HTML, CSS, JavaScript and Thymeleaf are used and written to build the clean and tidy user interface. Material design logic is realized in the aesthetic web page, and intuitive functions are set as friendly as possible.

The website are divided into four main pages: main page, search page, result page and contact page. The main page is to introduce the service of the website, the search page is to let users enter their inputs to search, and the contact page is to let users send their feedback.

In the search page, there are two options for users to choose, the first is to search the internal database, and the second is to search an uploaded file. All the functions inside of the view, which can also be concluded as front end part, needed to be realized in the controller.

Controller is the part that handle the front end request and implement the sorting algorithm. When users choose different functions in the front end, controller will handle the request to different function in the back end. In the external search, the file will firstly be uploaded to the virtual machine and then loaded into the temporary database table. Then both external and internal search will use MyBatis to generate a List object of TwitterData objects using annotation. These data will be sent to the bucket sort and the heap sort.

To sort all the twitters, a value should be generated to compare the similarity. A HashSet is used to store all the keywords users input, and each storing will be $O(1)$ time complexity. In the comparison process, the twitter will be splitted to words by `split()` function in Java, and each word will be checked by `contains()` function of HashSet, which also only use $O(1)$ time complexity. If the word is inside of the HashSet, then the similarity value will plus one. One of the other benefit of using HashSet besides speed is that if there are any duplicate keywords, they will automatically be deduplicated.

To reduce the time of searching with several methods, the project rewrites the constructor

of `TwitterData`, and will split the twitter string to string array automatically while object is generated by MyBatis. So the `split()` time will be inside of the time of loading data, but not inside of sorting data. And after sorting data, the whole similarity calculation will be $O(m)$ time complexity where 'm' is the size of the keywords set, as both the `contains()` and `add()` function of `HashSet` uses $O(1)$ time complexity.

In the bucket sort, because the length of twitter will be no longer than 140 words, the similarity value will not be larger than 140, so 140 buckets will be build for sorting. Iterate through all twitter data, and store the twitter to the bucket where the index equals to the similarity value. After the data iteration, iterate from the end of the bucket to the start, and add all the `TwitterData` objects inside of the bucket to the answer list one by one. If the answer list is larger or equal to the needed size, then break the loop. Also, remove the item at end of the list if the size of list is larger than the needed size. The total time complexity will be $O(mn)$ where 'm' is the number of keywords and 'n' is the size of the database.

In the heap sort, a min heap which size is the needed answer size will be created by `PriorityQueue` with overridden `Comparator` using similarity value. While iterating through all `TwitterData`, the similarity value of the new `TwitterData` will be compared to the top object in the min heap. If the size of heap has already reached the answer size and the new `TwitterData` has higher similarity value than the old one, then the min heap will `poll()` out the object with the minimum similarity value, and add the new `TwitterData` inside. The whole process will use $O(mn\log(k))$ time complexity where the 'k' is the needed answer size, 'm' is the keywords size and 'n' is the size of the twitter dataset.

After the sorting process, the taken time and answer list will be generated, and send by to the front end view part through Thymeleaf template engine. The result will be shown in a table. The figure 2.2 shows the architecture of the application.

The Spring Boot application was finally deployed on the Amazon EC2 virtual machine, which can be accessed here: <http://www.tty8128.com>.

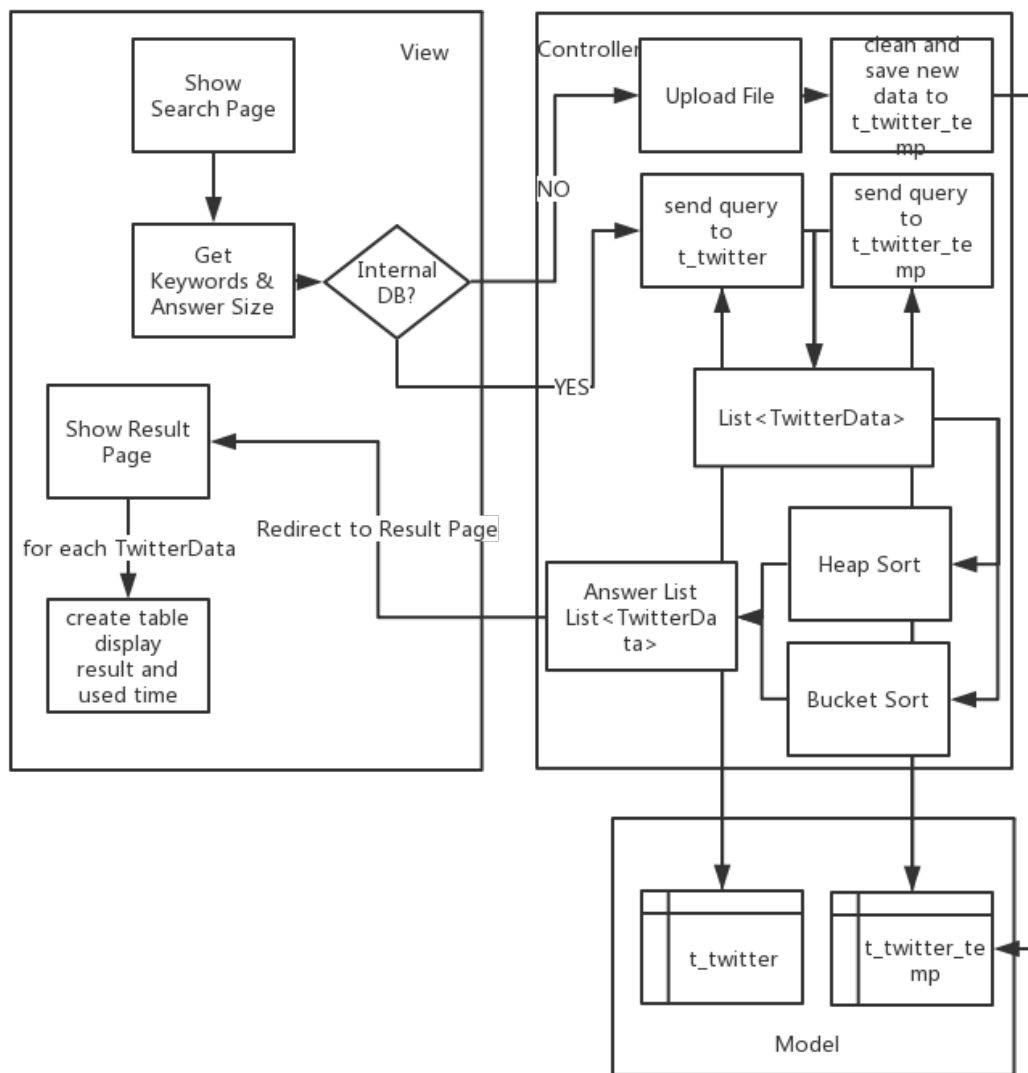


Figure 2.2: Architecture of the Spring Boot application

2.2 SECOND GENERATION: COMMAND LINE TOOL

Though the Spring Boot application can provide users with user-friendly user interface and good user experience, it has several problems.

- The application can not be run on the SCC for testing because students do not have the permission to access the MySQL database on the SCC.
- Initialize Object from database using the provided CSV (which includes over 1 million lines) via MyBatis may cause OOM(out of memory) problem on Amazon EC2, as the

ram of EC2 is limited, so the dataset needed to be limited.

The second problem actually can be handled by divide and conquer, reading objects from MySQL several times will solve it. But as the first problem can not be dealt, the project can only move to the next generation, which is the command line tool.

The command line tool is a Maven project written in Java. In the second generation, the whole workflow is similar to the Spring Boot application without database and front end web pages. The difference is that all the data are directly read by split lines with commas, and results are printed out by `System.out.println()` function. The similarity evaluation, bucket sort and heap sort functions are exactly the same as the ones in Spring Boot, so they will not be introduced again. The figure2.3 shows the workflow of the command line tool.

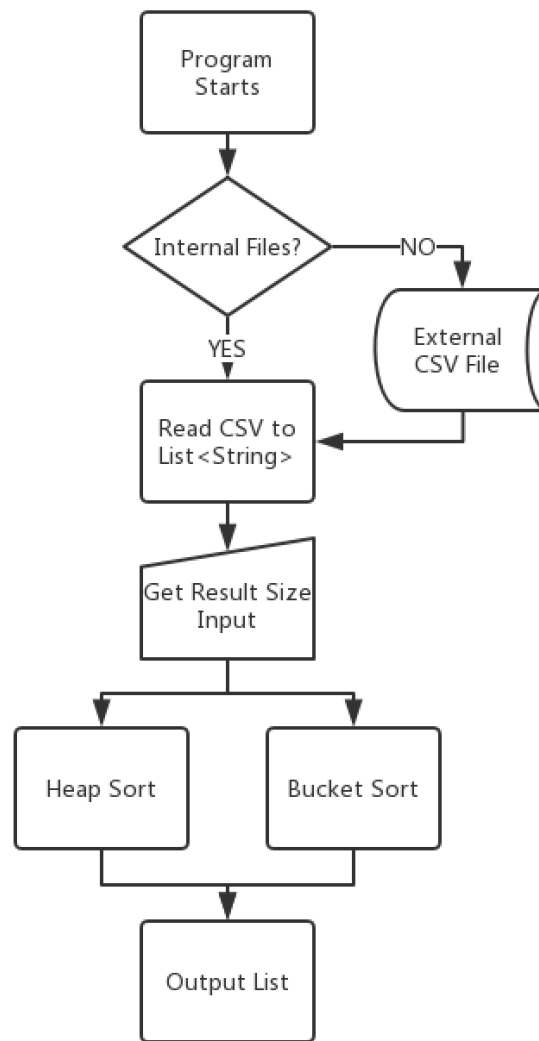


Figure 2.3: Second Generation: Command line tool workflow

2.3 THIRD GENERATION: COMMAND LINE TOOL

If users are trying to search several times, the second generation will be slow in speed because it has to iterate through the whole dataset again and again. To accelerate the program in multiple times searching, in the third generation, the application stored reversed index to the system before searching.

The reversed index will store all the words in twitter to a huge key-value pair sets, which in implementation of Java, would be a HashMap, as HashMap provide $O(1)$ time complexity read and store function and is the fastest Map in Java.

While iterate through all twitter strings in the List, these strings will firstly be splitted into words. For each word, the application will check whether there is such a key before in the HashMap, if there is not, then make the new value a second HashMap. If there is, then use the `get()` to get that old Map out as the second Map. The second map is called frequency map, in the frequency map the index of the twitter string will be stored as the key, and the value will be the appearance time of the certain words. For example, when the application iterate through the 153 index, and found twitter string "hello world hello world", it will firstly split the string to "hello", "world", "hello" and "world". Then it will check whether "hello" is inside of the first HashMap. If it is not, then put "hello" as a new key into it, with the value of a brand new HashMap as the frequency map. In the frequency Map, put the key 153, which is the index of the twitter, and the value 2, which is the appearance time of that word "hello" in that twitter, to the frequency HashMap.

The first HashMap will be called reversed index map. Using the reverse index map, the application can check the similarity value quickly. A similarity map will be built after users input the keywords. After program get all the keywords, it will iterate every single one of them. And then program will use that keyword to check the reversed index map, that whether there exists such a key. If there is, then program can get all the twitter indexes inside of it out and give them a similarity value, which is the appearance time in the map. The data will be stored to the similarity map, where key is the index, and value is similarity. When the program move to the second keyword, the indexes it gets out from the reversed index map may be exist inside the similarity map. Then the similarity of that index should add the old one. Until all the keywords are iterated, the similarity values will be the final values. Finally, the program will implement bucket sort and `Collections.sort()` of Java(which implements quick when the data size is small and merge sort when the data size is large) to sort the list that contains index of twitters, which in other word, keys of frequency map, via these similarity values inside of the map, and get the target result with size 'k' out. The whole workflow is shown on figure2.4.

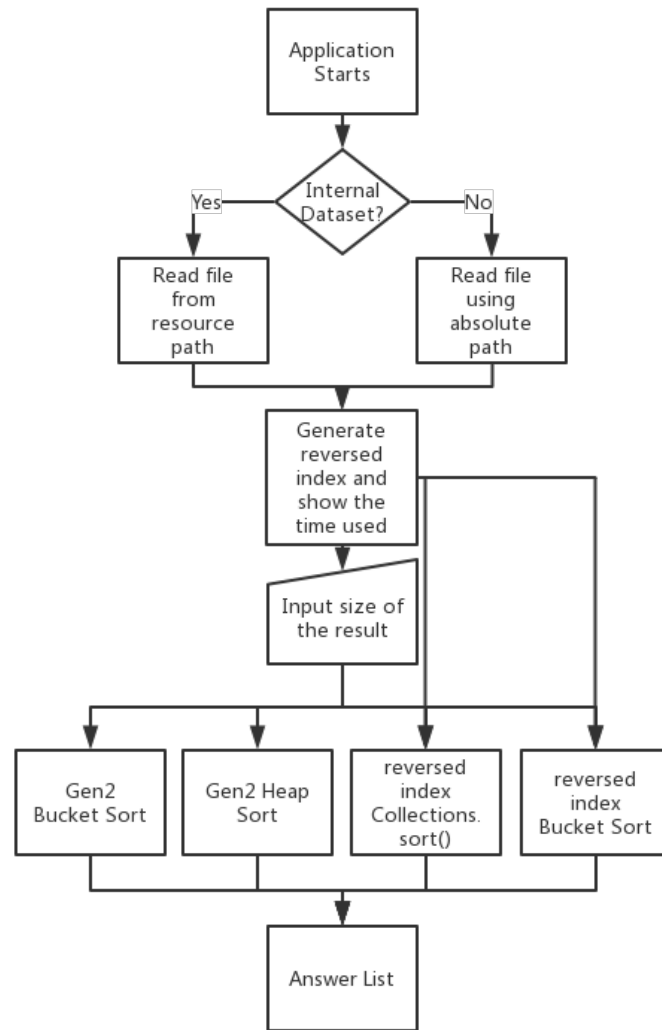


Figure 2.4: Third Generation: Command line tool workflow

In this implementation, the reverse index process will use $O(mn)$ time complexity, where m is the average length of the twitter string and n is the overall dataset size. However, the search process will be lightening fast. Consider the length of keywords is ' m ', the size of final result is ' k ', and the average number of twitter that related to each keyword is ' r ', then get all the related twitter will be $O(mr)$ time complexity, and the sort process will only use $O(mr)$ time complexity with bucket sort, $O(mr \log(mr))$ for the `Collections.sort()`. Because using the reversed index to search avoid the huge ' n ' (size of dataset), the third generation method is far more faster then the second generation. However, in the worst case where all the twitters have the keywords inside, this method can be really slow as ' r ' will be as huge as ' n '.

3 RESULT

3.1 FIRST GENERATION

The Spring Boot web application is deployed on Amazon EC2 and can be accessed via <http://www.tty8128.com>

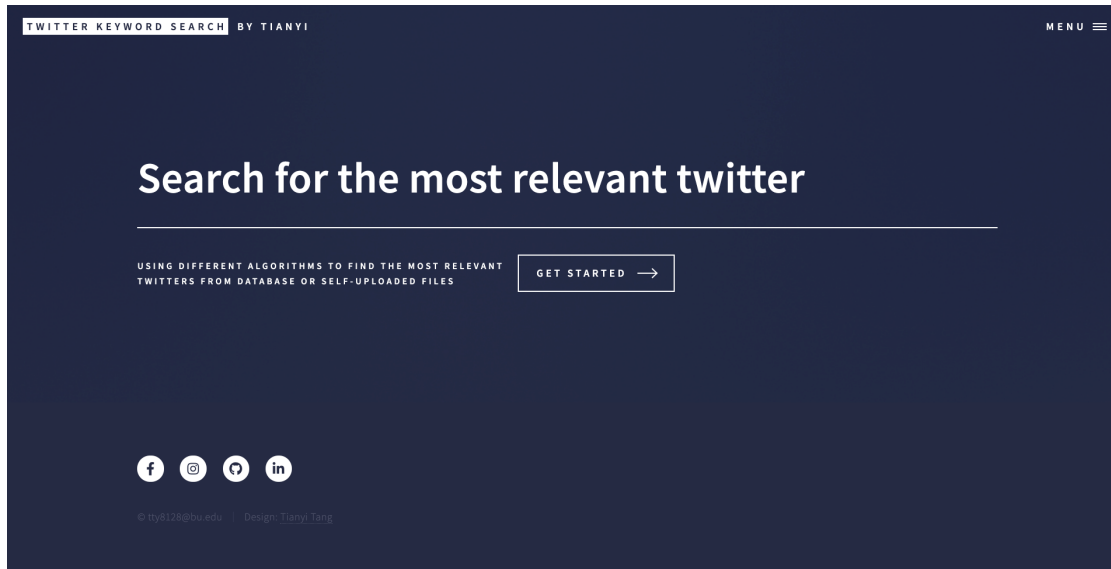



Figure 3.1: First Generation: Main Page

TWITTER KEYWORD SEARCHBY TIANYI

MENU

Search



Search the huge internal database by inputting your keywords below with space separated keywords, for example, "answer love cute have", and then put the size of twitter list that you would like to see in the result page, for example, "10". You can also search your own files by uploading CSV file with Col1:Date(yyyyMMdd), Col2:Twitter Text format. The first row will be ignored automatically.

WARNING: The file size is limited to 20Mb, or you may get an error from the website.
No matter how large your file might be, you will only get a dataset with size of 200000.
Also, please be patient as the dataset is very huge and searching will take you a long time.

Search Internal Dataset

KEYWORDS

Input your key words and separate them using spaces...

Size

INTERNAL SEARCH

CLEAR

Search External Dataset

Choose File

No file chosen

KEYWORDS

Input your key words and separate them using spaces...

Size

EXTERNAL SEARCH

CLEAR

© tny8128@bu.edu | Design: Tianyi Tang

Figure 3.2: First Generation: Search Page

TWITTER KEYWORD SEARCH
BY TIANYI
MENU

Advice?

If you have any advice for my project, please inform me!

NAME
EMAIL

MESSAGE

SEND MESSAGE
CLEAR

Email

Phone

Address

f
@
p
in

© tty8128@bu.edu | Design: Tianyi Tang

Figure 3.3: First Generation: Contact Page

TWITTER KEYWORD SEARCH
BY TIANYI
MENU

Result

Bucket Sort

15ms

DATE	TWITTER
Mon Nov 10 00:00:00 UTC 2003	navy ship to get bowen home away from home
Thu Jun 03 00:00:00 UTC 2004	beware work from home scam asic says
Tue Jun 15 00:00:00 UTC 2004	home staff to get more work
Thu Jul 01 00:00:00 UTC 2004	work from home move for council admin staff
Mon Aug 23 00:00:00 UTC 2004	delirious swedes create home from home
Wed Mar 16 00:00:00 UTC 2005	first sod to be turned on nursing home work
Fri Apr 22 00:00:00 UTC 2005	industrial unrest possible over nursing home work
Tue Jun 21 00:00:00 UTC 2005	work continues on new nursing home
Fri Jul 22 00:00:00 UTC 2005	police stop work over planned work changes
Wed Feb 19 00:00:00 UTC 2003	commonwealth bank cuts fixed home loan rates

Heap Sort

142ms

DATE	TWITTER
Wed Mar 16 00:00:00 UTC 2005	first sod to be turned on nursing home work
Fri Jul 22 00:00:00 UTC 2005	police stop work over planned work changes
Tue Jun 15 00:00:00 UTC 2004	home staff to get more work
Mon Aug 23 00:00:00 UTC 2004	delirious swedes create home from home
Fri Apr 22 00:00:00 UTC 2005	industrial unrest possible over nursing home work
Tue Jun 21 00:00:00 UTC 2005	work continues on new nursing home
Mon Nov 10 00:00:00 UTC 2003	navy ship to get bowen home away from home
Thu Jun 03 00:00:00 UTC 2004	beware work from home scam asic says
Thu Jul 01 00:00:00 UTC 2004	work from home move for council admin staff
Thu Nov 10 00:00:00 UTC 2005	police work to confirm azahari death

•

TRY AGAIN →

© ty8128@bu.edu
Design: Tianyi Tang

Figure 3.4: First Generation: Result Page

3.2 THIRD GENERATION

Because the third generation is an updated version of the second generation, it also includes the functions of the second generation, so the result from the second generation will not be mentioned in this report.

```
(base) [tty8128@scc1 TwitterKeywordSearch]$ java -jar twSearchCmd-2.0-SNAPSHOT.jar
-----
|                               Twitter Keyword Searcher                               |
|                               Developed by Tianyi Tang                               |
|                                                                                       |
|                               return the top relevant twitter with your keywords       |
|                                                                                       |
-----

Enter 1 to enter your own CSV file with absolute path,
enter anything else to use default CSV

Loading Default CSV File
1041776 twitters loaded

Start to create reversed index map
Reverse Index time: 4338ms

Please Input Your Keywords.
Split them using spaces, and end with enter
```

Figure 3.5: Third Generation: Loading Page

```
Please Input Your Keywords.  
Split them using spaces, and end with enter  
home work  
Please enter the size of the result:  
10  
  
*****  
  
Using the bucket sort...  
The time cost is 127 ms  
The answer generated is:  
  
navy ship to get bowen home away from home  
beware work from home scam asic says  
home staff to get more work  
work from home move for council admin staff  
delirious swedes create home from home  
first sod to be turned on nursing home work  
industrial unrest possible over nursing home work  
work continues on new nursing home  
police stop work over planned work changes  
why i work to save santas home  
  
*****
```

Figure 3.6: Third Generation: Bucket Sort

```
*****

Using the heap sort...
The time cost is 571 ms
The answer generated is:

union appeals against fair work order for crew to return to work
police stop work over planned work changes
home staff to get more work
airport staff sleeping at work cant afford to go home
industrial unrest possible over nursing home work
work continues on new nursing home
navy ship to get bowen home away from home
beware work from home scam asic says
work from home move for council admin staff
work to begin on $109 million fremantle dockers training home

*****
```

Figure 3.7: Third Generation: Heap Sort

```

*****

Using the reverse index...
The time cost is 7 ms
All Relevant Twitters Found

*****

Using the bucket sort to sort relevant twitters...
The time cost is 4 ms
The answer generated is:

regional home owners grant wont work reiq
work from home move for council admin staff
work continues on new nursing home
fair work launches crackdown on unpaid work
fair work ombudsman probes poultry producers work
work on pathfinder telescope on the home straight
delirious swedes create home from home
yahoo work from home ban puzzles business leaders
police stop work over planned work changes
airport staff sleeping at work cant afford to go home

*****

```

Figure 3.8: Third Generation: Bucket Sort with Reversed Index

```
*****

Using the Java default sort to sort relevant twitters...
The time cost is 8 ms
The answer generated is:

regional home owners grant wont work reiq
work from home move for council admin staff
work continues on new nursing home
fair work launches crackdown on unpaid work
fair work ombudsman probes poultry producers work
work on pathfinder telescope on the home straight
delirious swedes create home from home
yahoo work from home ban puzzles business leaders
police stop work over planned work changes
airport staff sleeping at work cant afford to go home

If you want to test again, enter 1.
Enter anything else will exit the program.
█
```

Figure 3.9: Third Generation: Collections.sort() with Reversed Index

4 SPEED TEST

To keep the test environment the same, all the test will be done on the same computer. The hardware and software environment is listed in table 4.1. Notice, the size of dataset for the Spring Boot is limited to 200'000 to avoid out of memory problems.

Name	2018 Macbook Pro 15'
System	MacOS
CPU	Intel i7-8750H
GPU	Radeon Pro 555X 4 GB
RAM	16GB

Table 4.1: The testing environment

4.1 KEYWORDS WITH LOW RELEVANCE RATE

4.1.1 TEST CASE 1

Keywords	Return Size
falconio external court	10

Table 4.2: Input 1

Web Application (with 200'000 twitters)		Command Line Tool (with 1'000'000 twitters)	
bucket sort	32ms	bucket sort	263ms
heap sort	268ms	heap sort	454ms
		create reversed index map	4097ms
		create frequency map	11ms
		(Reversed Index)bucket sort	4ms
		(Reversed Index)Collections.sort()	3ms

Table 4.3: Result 1

4.1.2 TEST CASE 2

Keywords	Return Size
tasmanian rugby toledo	10

Table 4.4: Input 2

Web Application (with 200'000 twitters)		Command Line Tool (with 1'000'000 twitters)	
bucket sort	29ms	bucket sort	101ms
heap sort	203ms	heap sort	380ms
		create reversed index map	5290ms
		create frequency map	2ms
		(Reversed Index)bucket sort	0ms
		(Reversed Index)Collections.sort()	1ms

Table 4.5: Result 2

4.2 KEYWORDS WITH HIGH RELEVANCE RATE

4.2.1 TEST CASE 3

Keywords	Return Size
for to have need go	30

Table 4.6: Input 3

Web Application (with 200'000 twitters)		Command Line Tool (with 1'000'000 twitters)	
bucket sort	27ms	bucket sort	111ms
heap sort	290ms	heap sort	639ms
		create reversed index map	5450ms
		create frequency map	47ms
		(Reversed Index)bucket sort	32ms
		(Reversed Index)Collections.sort()	36ms

Table 4.7: Result 3

4.2.2 TEST CASE 4

Keywords	Return Size
all need as can could would will	50

Table 4.8: Input 4

Web Application (with 200'000 twitters)		Command Line Tool (with 1'000'000 twitters)	
bucket sort	35ms	bucket sort	110ms
heap sort	346ms	heap sort	701ms
		create reversed index map	3846ms
		create frequency map	12ms
		(Reversed Index)bucket sort	6ms
		(Reversed Index)Collections.sort()	6ms

Table 4.9: Result 4

4.3 LONG SENTENCES

4.3.1 TEST CASE 5

Keywords	Return Size
state rail says psych testing as rigorous holly good new stuff having hello night yeah	50

Table 4.10: Input 5

Web Application (with 200'000 twitters)		Command Line Tool (with 1'000'000 twitters)	
bucket sort	42ms	bucket sort	122ms
heap sort	344ms	heap sort	800ms
		create reversed index map	5425ms
		create frequency map	19ms
		(Reversed Index)bucket sort	12ms
		(Reversed Index)Collections.sort()	16ms

Table 4.11: Result 5

4.3.2 TEST CASE 6

Keywords	Return Size
create submit party morning test return size keyword world tab sleep	50

Table 4.12: Input 6

Web Application (with 200'000 twitters)		Command Line Tool (with 1'000'000 twitters)	
bucket sort	34ms	bucket sort	106ms
heap sort	307ms	heap sort	758ms
		create reversed index map	3367ms
		create frequency map	6ms
		(Reversed Index)bucket sort	4ms
		(Reversed Index)Collections.sort()	4ms

Table 4.13: Result 6

5 SUMMARY

From the testing result, it is very clear that the speed of the bucket sort is faster than the heap sort. Also, reversed index is a huge step forward if the users are searching the same database for several times.

In the future, there are still possibilities to improve the application. First of all, for now there is no caching system for the normal heap sort. It means that if a new twitter is put into the min heap, the heapify process will cause extra time complexity to re-calculate the similarity value of the twitters on its heapify path. These values can be stored into the HashMap to avoid the re-calculation. However, if use the twitter string as the key, the get() function from HashMap will actually cause higher time complexity in my test. It should be effected by the hash collision due to the large amount of data. Using the index of List<TwitterData> to store similarity value to an array may solve this problem, though it has not been tested yet.

Also, the Spring Boot application can use some local lite database like SQLite to avoid the MySQL permission problem on SCC. Changing the database can also make the web application run on the SCC successfully.

5.1 WORK BREAKDOWN

- Tianyi Tang: Finished the project individually.

REFERENCES

- [1] En.wikipedia.org. heapsort.
- [2] Geeksforgeeks - priorityqueue in java.
- [3] Spring.io. spring by pivotal.
- [4] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. Introduction to algorithms.
- [5] SKIENA, S. S. The algorithm design manual.