Jordan Dahmen

Ben Carroll

05 / 06 / 15

CS 475 spring 2015

A Concurrent Scraper for Reddit User and Comment Statistics

Table of Contents

Scraper Description

Our program concurrently scrapes and concurrently analyzes Reddit user and user comments to generate interesting metrics. This is accomplished by multi-threaded scraping of all posts made by a specified user or list of users, which are then returned to a series of multi-threaded user metrics. Currently, the program implements post frequency and word frequency analysis, but could be easily extended to implement other user metrics. In writing this program, we are trying to provide information that cannot be gleaned simply by viewing the user or users' Reddit homepage.

Program Design and Structure

Our program was designed to be extendable, and currently consists of a **MainMenu**, where a user can enter Reddit user names or a file containing a list of Reddit usernames. This then invokes the **ScraperManager,** which handles much of the concurrency, through the use of the Java ExecutorService framework and the Java Future interface. The ExecutorService handles thread creation, while the Future interface allows for the asynchronous return of objects. The **ScraperManager** creates a fixed number of Scraper threads, which process up to that number of Reddit users concurrently, re-using threads if necessary for additional users. Each **Scraper** loads and parses the user's comments page by page through the jsoup (HTML parsing) library and inserts them into a list of **Comments**, which is then returned to the Future in the **ScraperManager**. Reddit user comments are listed 25 to a page. The

Scraper uses a random port to connect to and retrieve each page of comments, until there are no more pages, and parses the HTML for the necessary information, pausing for a random number of seconds every three pages. This implements a crude form of rate-limiting, to avoid potentially being banned by Reddit for making too many requests. In our program, that network connection is certainly the limiting factor for speed.

Our program uses **Comment, Post, Sub, User, Word,** and **Day** objects to store data extracted from Reddit pages, while it is in different stages of processing. A **Comment** object stores a single comment by a single user, while a **Post** object is a comment that begins a discussion thread. The **Sub** and **Word** objects contain subreddits and individual words, respectively. **Day** objects keep track of how many posts were posted on what day of the week. All of these are encapsulated by the **User** object, which when complete, contains visited subreddits, word frequency, post rate, user comments and posts, and any other associated information.

Once the user's comments have been scraped, the **ScraperManager** creates a dynamically allocated thread pool to analyze posts and comments. Analyzed posts and comments are passed to the **analyzeComments** method in **AnalyzeComments**, where punctuation is stripped, and word frequency is computed. The comment is then retrieved from its Future to be added to its corresponding User object (an object representing the author). Once the comments and posts are matched to their users, all the users are retrieved from their Future, added to a list of users, and the list is returned to the MainMenu, where the pertinent information is displayed.

Program Design Issues

Originally, our ExecutorService code implemented runnable instead of callable, however, it quickly became apparent that we needed to return single comments, lists of comments, and eventually users and lists of users, in the AnalyzeComment and ScraperManager classes. This was successfully implemented with a little bit of trouble. In particular, certain code, when not shut down correctly would create a number of zombie processes.

We had originally planned to scrape the user's Reddit gilded page as well, but did not implement this, as doing so would require more requests to Reddit's servers, as discussed in the next section.

Possible Future Additions

Our program could be extended to check for "gilded" Reddit comments. Originally, we had planned to, but this would require an additional set of scrapers for each user. This would also require additional requests per user to Reddit's servers, and would be best implemented in a distributed fashion, so that not all the requests are coming from the same IP address.

In terms of additional metrics, there are many possibilities. Our program could be extended to calculate readability metrics for a user. Readability metrics rank text based on how difficult it is to read, and provide a guess as to the user's writing or education level. Finally, our program could be extended to calculate these same metrics on a subreddit basis in addition to the current user basis. It would also be reasonable to make a GUI interface to replace the current CLI interface. It could also be extended to calculate readability metrics for a user.

Test Design

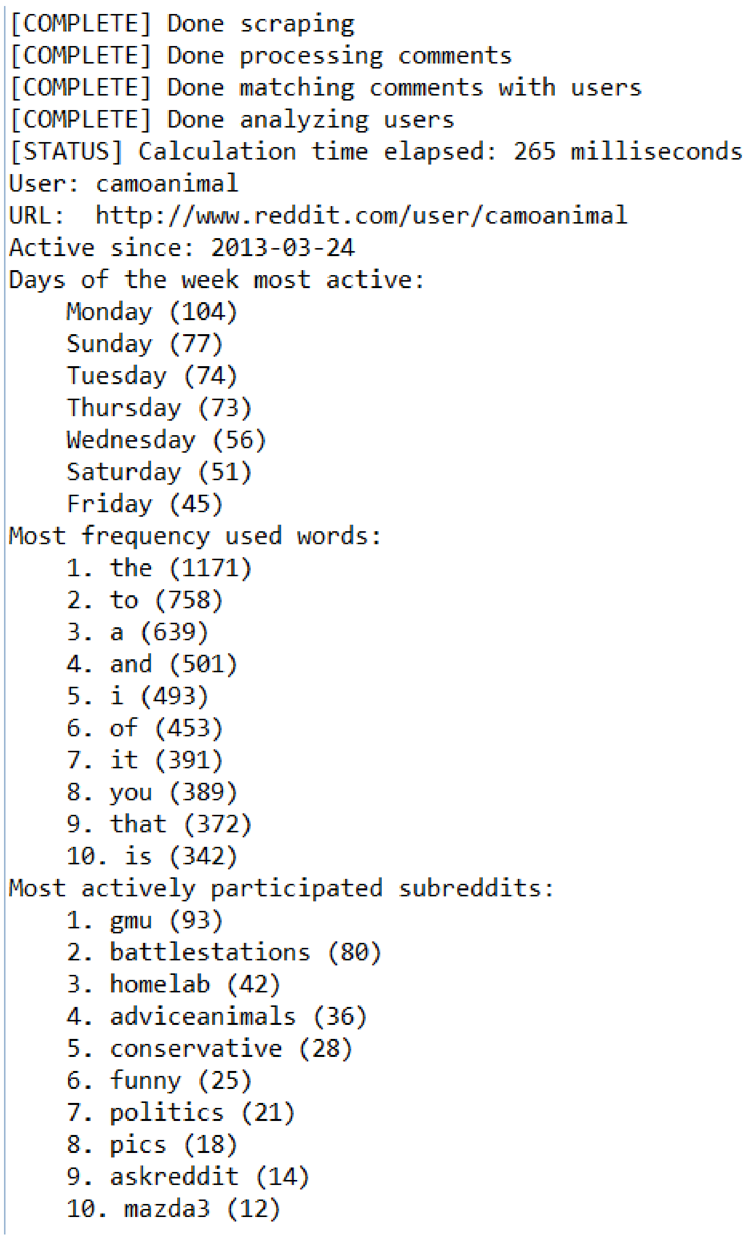
Originally, we had Junit test cases for the Comment and User classes. However, these became cumbersome once we started pulling in HTML data from actual users. With this, it was easier to test with print statements in the code. During the development process, we ran the code hundreds of times with dozens of Reddit user pages, when both testing and implementing new code. We assumed that this was a sufficient number of test cases to find major bugs within the code.

Our program has been tested extensively on real Reddit users (mostly picked at random). As the program is now, we have not seen any errors from our code, although it is possible that there are edge cases that might occur when certain input is parsed or analyzed, that we do not know about.

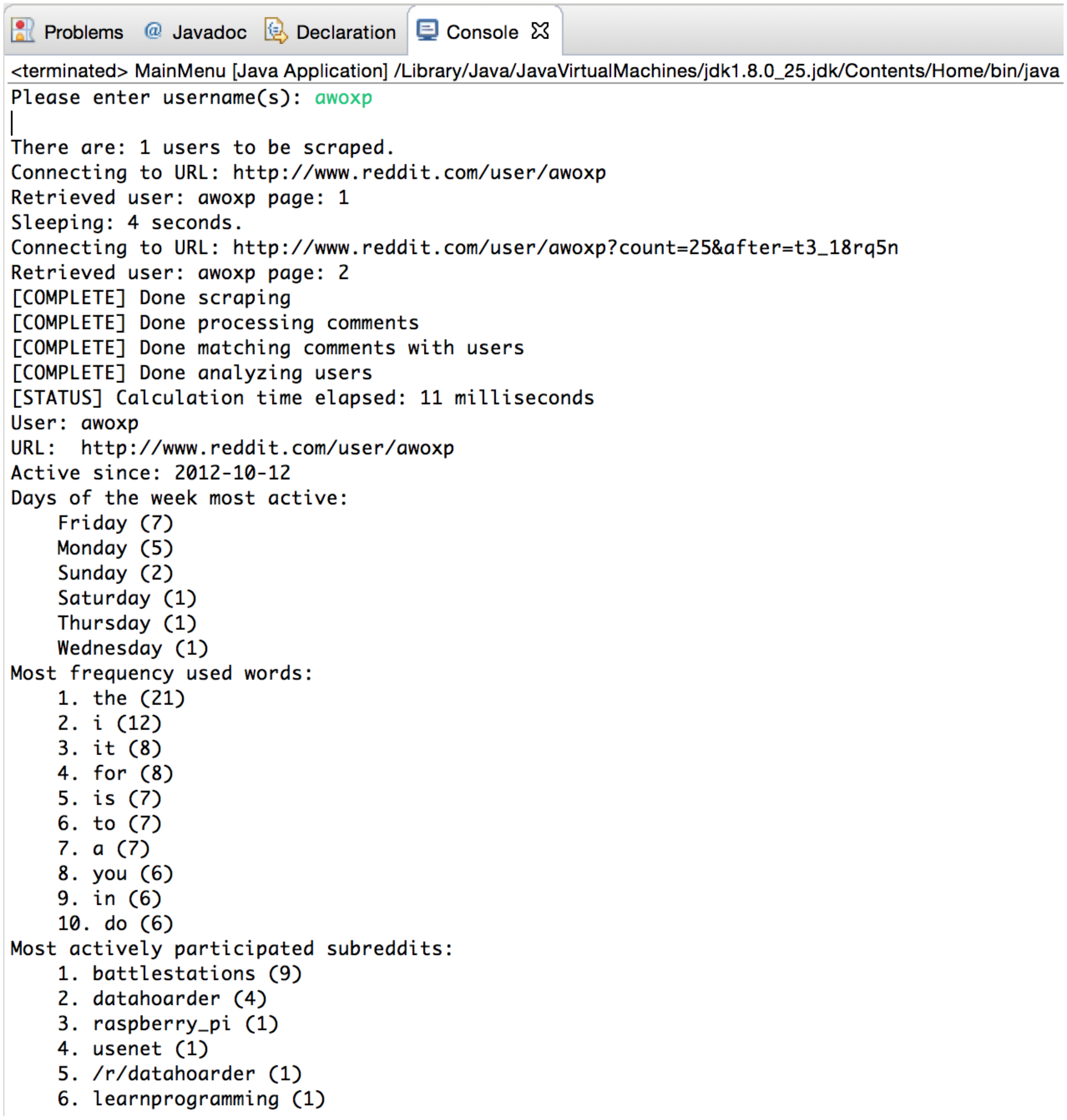
Server-side errors are somewhat common, especially when run on the GMU network. HTTP 429 (Too Many Requests) errors occur frequently when Reddit is rate-limiting requests. Reddit appears to rate-limit based on IP address; therefore, all people accessing Reddit from the GMU network appear as one IP address or a pool of IP addresses and are treated as a single “entity.” This error is mostly gone if run from a private network, or better still, a VPS with a dedicated IP address. When Reddit is under peak load, it is still possible to see HTTP 503 (Service unavailable) errors occur. Reddit does not appear to have the server-capacity to meet peak demand.

Analysis of Results

After a user (or users) is entered and scraped, assuming no HTTP 429 or 503 errors are encountered, a listing of the user’s username, Reddit URL, first post date, days of the week the poster is most active, top 10 frequently used words, top 10 most frequently visited subreddits, and post frequency per day of the week is displayed, as seen in the screenshots below.



```
[COMPLETE] Done scraping
[COMPLETE] Done processing comments
[COMPLETE] Done matching comments with users
[COMPLETE] Done analyzing users
[STATUS] Calculation time elapsed: 265 milliseconds
User: camoanimal
URL:  http://www.reddit.com/user/camoanimal
Active since: 2013-03-24
Days of the week most active:
    Monday (104)
    Sunday (77)
    Tuesday (74)
    Thursday (73)
    Wednesday (56)
    Saturday (51)
    Friday (45)
Most frequency used words:
    1. the (1171)
    2. to (758)
    3. a (639)
    4. and (501)
    5. i (493)
    6. of (453)
    7. it (391)
    8. you (389)
    9. that (372)
    10. is (342)
Most actively participated subreddits:
    1. gmu (93)
    2. battlestations (80)
    3. homelab (42)
    4. adviceanimals (36)
    5. conservative (28)
    6. funny (25)
    7. politics (21)
    8. pics (18)
    9. askreddit (14)
    10. mazda3 (12)
```



```
<terminated> MainMenu [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents/Home/bin/java
Please enter username(s): awoxp

There are: 1 users to be scraped.
Connecting to URL: http://www.reddit.com/user/awoxp
Retrieved user: awoxp page: 1
Sleeping: 4 seconds.
Connecting to URL: http://www.reddit.com/user/awoxp?count=25&after=t3_18rq5n
Retrieved user: awoxp page: 2
[COMPLETE] Done scraping
[COMPLETE] Done processing comments
[COMPLETE] Done matching comments with users
[COMPLETE] Done analyzing users
[STATUS] Calculation time elapsed: 11 milliseconds
User: awoxp
URL:  http://www.reddit.com/user/awoxp
Active since: 2012-10-12
Days of the week most active:
    Friday (7)
    Monday (5)
    Sunday (2)
    Saturday (1)
    Thursday (1)
    Wednesday (1)
Most frequency used words:
    1. the (21)
    2. i (12)
    3. it (8)
    4. for (8)
    5. is (7)
    6. to (7)
    7. a (7)
    8. you (6)
    9. in (6)
    10. do (6)
Most actively participated subreddits:
    1. battlestations (9)
    2. datahoarder (4)
    3. raspberry_pi (1)
    4. usenet (1)
    5. /r/datahoarder (1)
    6. learnprogramming (1)
```

Overall, we are relatively happy with the scraper as it is. It provides user and comment statistics that are not readily seen on the user's Reddit page. It uses concurrency to both scrape and process comment and user data, and can be extended further in the future.

References

CS 475 class notes: Parallel Framework:

https://cs.gmu.edu/~rcarver/cs475/ParallelFramework.pdf

Java Executor:

https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html

Java ExecutorService:

https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ExecutorService.html

Java Future interface:

https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Future.html

Code Listing

The code for ScraperManager.java, AnalyzeComment.java, and AnalyzeUser.java has been included starting on the following page. These three java classes contain much of the concurrent code within our program. The code for the other java files that makeup our program can be viewed in the included zip file.