Project #3 - Final Project

Information Retrieval

Dr. Sabine Bergler

https://github.com/vincent-fugnitto/IR-FinalProject

Michel Polisena-Petrelli, 40005141
Stephen Sugumar, 26561489
Vincent Fugnitto, 27207999

26th November, 2017

Design Decisions

Overview

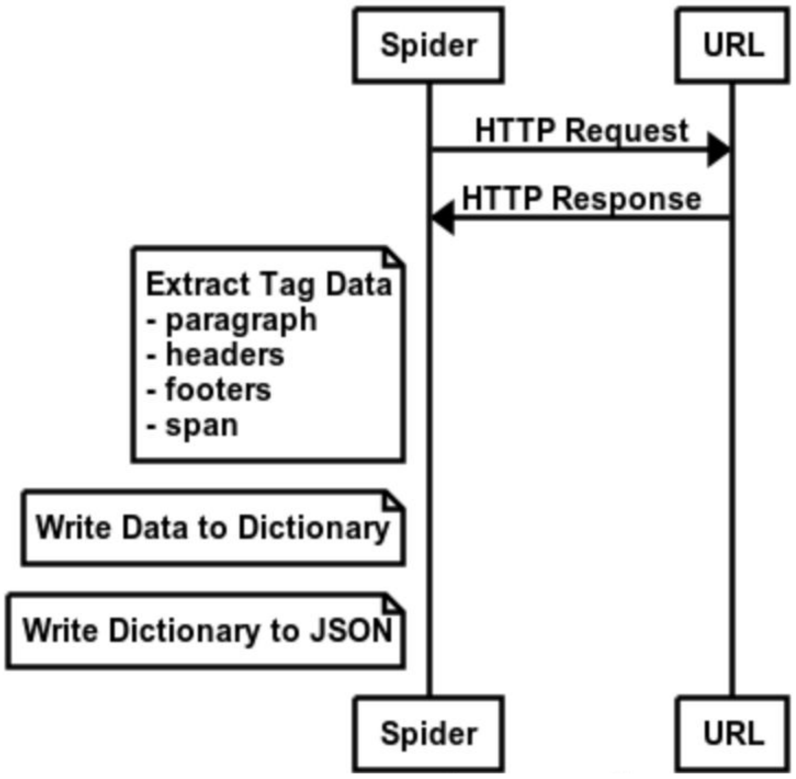
For our sentiment analysis search system, we utilized Python and various Python libraries to successfully create both a complete web crawler, inverted index, sentiment analysis and search system. We decided to use Python as a programming language because of its code simplicity, readability, and native libraries. Furthermore, the Python framework Scrapy was used for our web crawler, while the aFinn library was used for sentiment scoring.

Spider

Initially, our web crawler was written using Requests and BeautifulSoup. The Requests library was used to perform HTTP requests to various URLs present on each page crawled. BeautifulSoup was then used to scrape the data of each page by various textual tags. However, implementation of the web crawler was updated to the use of the Scrapy python framework. When refactoring from Requests and BeautifulSoup to Scrapy, we witnessed a greater increase in processing speed, fewer lines of code thus less potential errors/complexity, and most importantly easier data extraction. Scrapy also automatically crawls and follows the embedded URLs of each page to crawl, it handles crawling web pages concurrently to speed up processing, has the ability to set a max bound of URL pages visited, has native support for extracting data based on HTML tags, and has native support for writing results to JSON. We decided for these reasons that Scrapy was the ideal choice for the project.

Our custom Scrapy spider works by inheriting the CrawlSpider which by default can be used to follow a set of rules for web crawling. Our spider works by starting a set of URLs which will be used on initialization and following links within a webpage to iteratively scrape. For each web page, we decided that we will scrape based on common HTML textual tags being headers (h1 to h6 tags), p tags (paragraph), footers, and span tags. For each of these tags, we ensured that only word occurrences would be recorded using a regex. In the end, we stored results in a dictionary of URL keys and text values. Scrapy handled writing the results to a JSON file which would later be used when writing the inverted index.

Extracting Web Page Data



Inverted Index & Search

The creation of the inverted index takes place in the “inverted_index.py” class. Using a dictionary for the index, the code parses the JSON file created by the scraper. The inverted index stores the terms, their sentiment values, and their postings along with each posting’s frequency for that term. The dictionary is sorted after parsing is completed.

Since storing each web pages’ length and sentiment value inside the inverted index for each term would be redundant, each page has those values stored within a text file called doc_stats.txt. It contains the URL, the frequency and the sentiment value for each web page.

The search function will simply iterate through the terms and intersect a set of postings from each term. The sentiment value calculated for the query will be used to tell whether the results must be ordered in ascending or descending order, in term of sentiment value.

Usages

Spider

The spider can be executed using:

- **optional -max {value}:** if present specifies max page bound, else default = 10

```
$ python spider.py
$ python spider.py -max {value}
```

Inverted Index

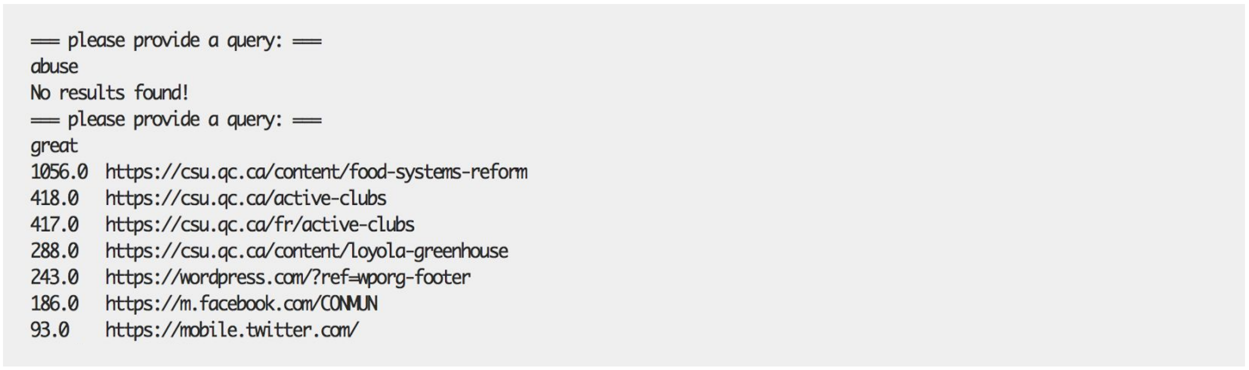
The inverted index can be executed with the optional -build parameter:

- **optional -build:** specifies whether or not to rebuild the index. If present, inverted index is rebuilt

```
$ python inverted_index.py
$ python inverted_index.py -build
```

Searching

When the Inverted Index is executed, the terminal will prompt the user for queries and display appropriate results

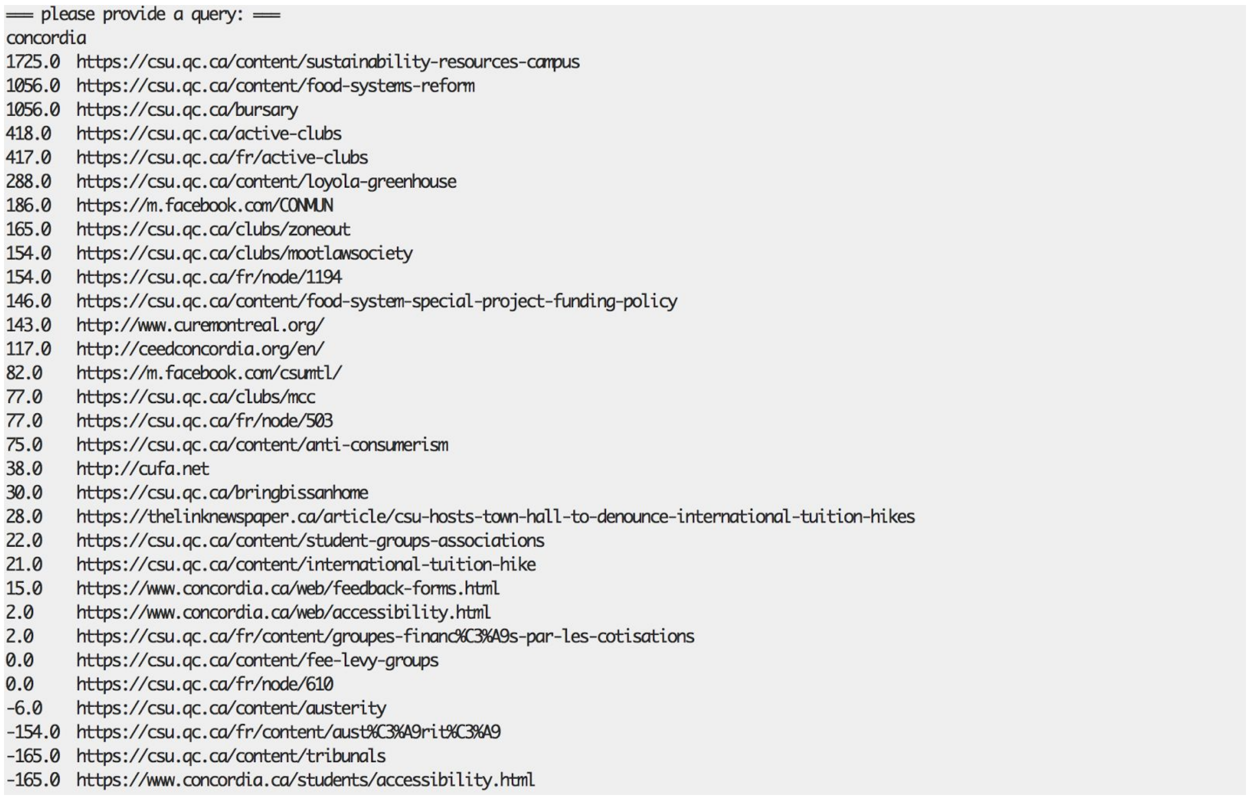


```
== please provide a query: ==
abuse
No results found!
== please provide a query: ==
great
1056.0  https://csu.qc.ca/content/food-systems-reform
418.0   https://csu.qc.ca/active-clubs
417.0   https://csu.qc.ca/fr/active-clubs
288.0   https://csu.qc.ca/content/loyola-greenhouse
243.0   https://wordpress.com/?ref=wporg-footer
186.0   https://m.facebook.com/CONMUN
93.0    https://mobile.twitter.com/
```

Sample Search Results

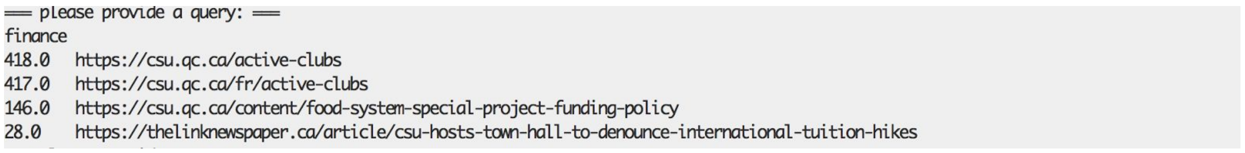
Results generated with a spider with a max bound of 50 pages

Query: **concordia** (neutral sentiment)



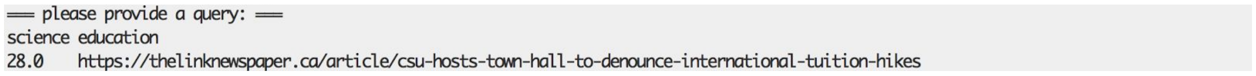
```
== please provide a query: ==
concordia
1725.0  https://csu.qc.ca/content/sustainability-resources-campus
1056.0  https://csu.qc.ca/content/food-systems-reform
1056.0  https://csu.qc.ca/bursary
418.0   https://csu.qc.ca/active-clubs
417.0   https://csu.qc.ca/fr/active-clubs
288.0   https://csu.qc.ca/content/loyola-greenhouse
186.0   https://m.facebook.com/CONMUN
165.0   https://csu.qc.ca/clubs/zoneout
154.0   https://csu.qc.ca/clubs/mootlawsociety
154.0   https://csu.qc.ca/fr/node/1194
146.0   https://csu.qc.ca/content/food-system-special-project-funding-policy
143.0   http://www.curemontreal.org/
117.0   http://ceedconcordia.org/en/
82.0    https://m.facebook.com/csumtl/
77.0    https://csu.qc.ca/clubs/mcc
77.0    https://csu.qc.ca/fr/node/503
75.0    https://csu.qc.ca/content/anti-consumerism
38.0    http://cufa.net
30.0    https://csu.qc.ca/bringbissanhome
28.0    https://thelinknewspaper.ca/article/csu-hosts-town-hall-to-denounce-international-tuition-hikes
22.0    https://csu.qc.ca/content/student-groups-associations
21.0    https://csu.qc.ca/content/international-tuition-hike
15.0    https://www.concordia.ca/web/feedback-forms.html
2.0     https://www.concordia.ca/web/accessibility.html
2.0     https://csu.qc.ca/fr/content/groupes-financ%C3%A9s-par-les-cotisations
0.0     https://csu.qc.ca/content/fee-levy-groups
0.0     https://csu.qc.ca/fr/node/610
-6.0    https://csu.qc.ca/content/austerity
-154.0  https://csu.qc.ca/fr/content/aust%C3%A9rit%C3%A9
-165.0  https://csu.qc.ca/content/tribunals
-165.0  https://www.concordia.ca/students/accessibility.html
```

Query: **finance** (neutral sentiment)



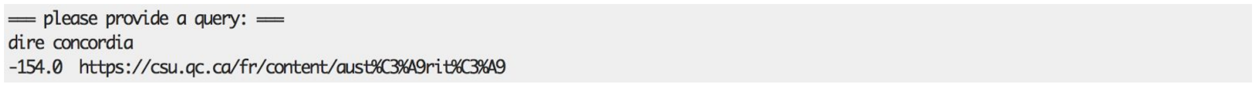
```
== please provide a query: ==
finance
418.0   https://csu.qc.ca/active-clubs
417.0   https://csu.qc.ca/fr/active-clubs
146.0   https://csu.qc.ca/content/food-system-special-project-funding-policy
28.0    https://thelinknewspaper.ca/article/csu-hosts-town-hall-to-denounce-international-tuition-hikes
```

Query: **science education** (neutral sentiment)



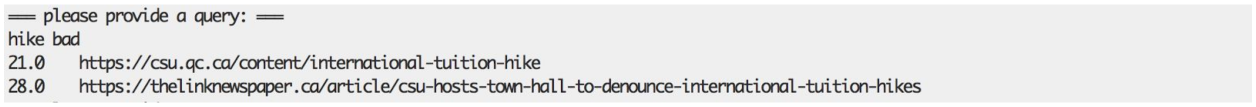
```
== please provide a query: ==
science education
28.0    https://thelinknewspaper.ca/article/csu-hosts-town-hall-to-denounce-international-tuition-hikes
```

Query: **dire concordia** (negative sentiment)



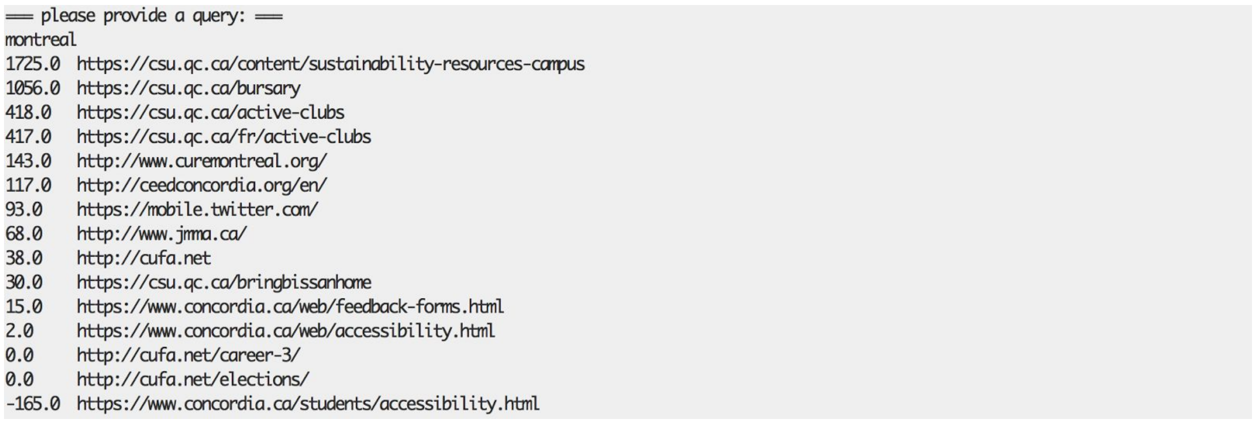
```
=== please provide a query: ===
dire concordia
-154.0  https://csu.qc.ca/fr/content/aust%C3%A9rit%C3%A9
```

Query: **hike bad** (negative sentiment)



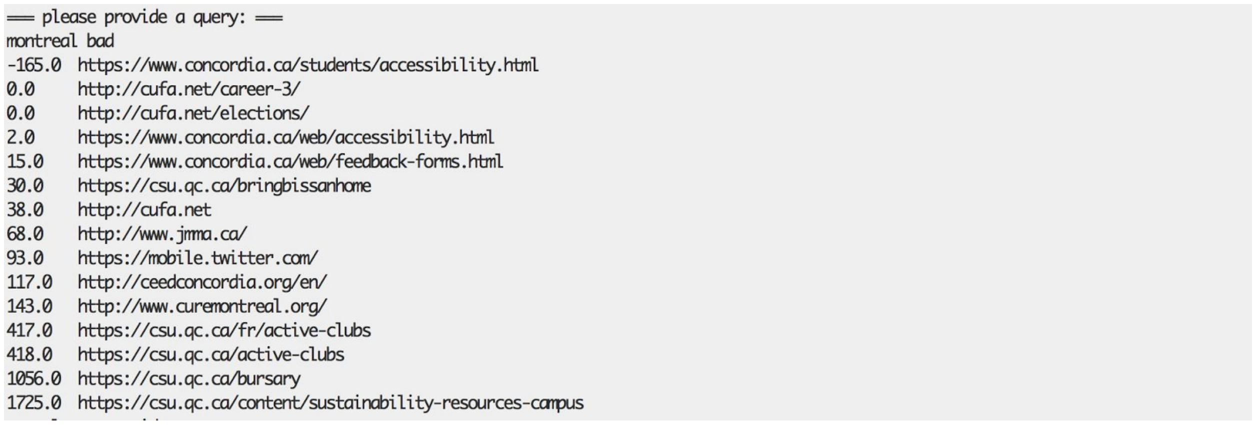
```
=== please provide a query: ===
hike bad
21.0    https://csu.qc.ca/content/international-tuition-hike
28.0    https://thelinknewspaper.ca/article/csu-hosts-town-hall-to-denounce-international-tuition-hikes
```

Query: **montreal** (neutral sentiment)



```
=== please provide a query: ===
montreal
1725.0  https://csu.qc.ca/content/sustainability-resources-campus
1056.0  https://csu.qc.ca/bursary
418.0   https://csu.qc.ca/active-clubs
417.0   https://csu.qc.ca/fr/active-clubs
143.0   http://www.curemontreal.org/
117.0   http://ceedconcordia.org/en/
93.0    https://mobile.twitter.com/
68.0    http://www.jmma.ca/
38.0    http://cufa.net
30.0    https://csu.qc.ca/bringbissanhome
15.0    https://www.concordia.ca/web/feedback-forms.html
2.0     https://www.concordia.ca/web/accessibility.html
0.0     http://cufa.net/career-3/
0.0     http://cufa.net/elections/
-165.0  https://www.concordia.ca/students/accessibility.html
```

Query: **montreal bad** (negative sentiment)



```
=== please provide a query: ===
montreal bad
-165.0  https://www.concordia.ca/students/accessibility.html
0.0     http://cufa.net/career-3/
0.0     http://cufa.net/elections/
2.0     https://www.concordia.ca/web/accessibility.html
15.0    https://www.concordia.ca/web/feedback-forms.html
30.0    https://csu.qc.ca/bringbissanhome
38.0    http://cufa.net
68.0    http://www.jmma.ca/
93.0    https://mobile.twitter.com/
117.0   http://ceedconcordia.org/en/
143.0   http://www.curemontreal.org/
417.0   https://csu.qc.ca/fr/active-clubs
418.0   https://csu.qc.ca/active-clubs
1056.0  https://csu.qc.ca/bursary
1725.0  https://csu.qc.ca/content/sustainability-resources-campus
```

Additional Questions

Question : what was the hardest step?

One of the hardest steps was building an application which would be compatible with the school system and to successfully understand how to use Scrapy to crawl and scrape web pages for content. Initially, Scrapy would return unwanted terms such as inner tags, javascript content and inline CSS. This, of course, was not needed and was detrimental to the performance of the application. A lot of the work was used to search for various solutions on how to only obtain textual content which would add value to the application. Since Scrapy is an elegant solution to both crawling and scraping, we decided that it could also be used to extract valid words, tokenize and normalize content before the inverted index was created. Since Scrapy uses concurrent processing, it is highly beneficial for it to perform string operations instead of when building the index.

Question : how big is your index?

Our index size can be dynamically set by the user when he first runs the spider when specifying an optional bound parameter. He can choose the max bound of URLs to visit and thus JSON objects to write to the results file. The inverted index size is based on the number of terms extracted by our spider and thus if a larger bound is used when executing the spider, the larger the inverted index will be. On average, there are around 1,000 tokens per page.

Question : how did you define what constitutes an document in your index?

In our implementation, each URL (web page) is a document. Since our search system returns URLs which are associated with the terms in the queries, we have defined URLs to be our documents.

Question : what observations did you make during your experiments?

During experimenting with the code and results we observed that as intended, default sentiments provided no score to the overall score of the document sentiment. Stopwords for example could have been removed to increase processing speed since they did not contribute to the sentiment score, but we chose not to remove them since we wanted a more realistic representation for the user when performing a search query with stopwords present. Since we had the processing capabilities, we decided in keeping stopwords present in the solution. We also noticed how sentiment scoring worked and in which cases they would be useful such as for blogs, reviews etc.

Question : what did you learn from your experience?

Throughout the project, we learned of course how to effectively work as a team and delegate roles and user stories on how to complete the code and to the best of our abilities. We learned how sentiment analysis works functionally and in which contexts they are most appropriate, such as for blogs, reviews, customer service, social media contexts. The project also permitted us to learn more about web crawling and how it can be used successfully to crawl and scrape web pages for content in a complete solution. The project allowed us to see the implementation of a complete solution for defining and obtaining a corpus to create an inverted index and creating a search system.

Individual Contributions

The implementation of the application was divided into two main modules, the web crawler to extract data and create a corpus, and the inverted index module to build an index based on sentiment scoring. Since Michel and Vincent both had extensive previous knowledge of Python programming, and the code was difficult to intuitively divide into three developers without merge issues, they lead the implementation effort in developing a module each while Stephen wrote an initial draft of the project report. Michel was assigned the task of implementing the index module with aFinn sentiment scoring and searching, while Vincent was tasked with implementing the web crawler and modularizing the code with build parameters. Since both the index and web crawler were directly related, it was important that both Vincent and Michel worked closely together. In order to ensure no issues or bugs were present in the final solution, unit testing of both the crawler and index modules were performed, along with functional testing of the entire application, and peer testing to fully test the application. In the end, each team member also performed end-to-end testing on the various operating system as a means of user-acceptance testing.

The initial draft of the entire report was created by Stephen and he ensured that each team member aided in providing content for their respective implementation efforts since coding and design details were required. In the end, each team member attributed with the completion of the project report with coding details, implementation design, decisions, and with answers to the various required questions.