

CPS842

SEARCH ENGINE

0.1 Implementation Details

In the following we will discuss different steps of our implementation details of the search engine.

0.1.1 Crawler

We use an existing crawler from github, <https://github.com/mirkomantovani/web-search-engine-UIC/blob/master/crawler.py>, to collect some web data which is a multithreaded crawler with 20 worker nodes. We choose <https://www.ryerson.ca> as our *seed url* and restrict the data collection to only *ryerson* domain. To comply with politeness of crawlers, each request to server has a *10 sec* timeout. We crawled over 12,000 pages; however, we only processed 10,000 pages due to the hardware limitation.

0.1.2 Ranking algorithm

For ranking algorithm, we implemented **vector space model**. To calculate document weights, we implemented only *IDF*. The final score is calculated based on the following score (d, q) calculation:

$$\text{score } (d, q) = w_1 * \text{cosine-similarity}(d, q) + w_2 * \text{pagerank } (d) \quad (1)$$

We find that $w_1 = 0.3$ and $w_2 = 0.7$ give us better results. In order to perform the page ranking we had to create a graph of the network

We apply the page ranking algorithm in our *parser* class which does the parsing of *html* files and pre-process the result. We run the page ranking algorithm 200 times; however, it converged after *150 iteration*. Every intermediate result which we need for searching e.g., Dictionary, posting list, web graph are stored in pickle files and are loaded during the searching process.

0.1.3 Inverted index

After pre-processing the web data, we created tuples of (*word, docID*) and from that we created dictionary and posting list. We further calculated the length of each document in our data set and stored in a dictionary (*doc-length-dict*) in order to expedite the searching process.

0.1.4 Challenges

The most difficult part was to find a crawler that was easier to use and was appropriate for our machine. We spent days working with *Apache nutch* to use it as a crawler but we could not figure it out. We also tried working with other open source crawlers and the only crawler that worked for us was <https://github.com/mirkomantovani/web-search-engine-UIC/blob/master/crawler.py>. So we decided to use the same page ranking as well from this directory. We originally wanted to crawl more web pages and process more data; however, our machines hardware limitation did not allow us. We put the crawler on to work overnight however every time it crashed after 1 or 2 hours of working.

0.1.5 Running the program

To run the program, the following steps are followed:

1. To start searching :
 - Run search.py
2. To create crawl :
 - Run run_multithreaded.py
3. To Preprocess
 - Run m_invert.py
 - Run run_page_rank.py

Once the program is executed, a simple UI will appear which will take in a query and display the first 100 results related to ryerson. If the user double clicks the link, the program should take the user to the website,as shown below with screenshots.We have also added screenshot (Figure 10, and Figure 11) for the query which has no results such as when we search for "Saghar Jiantavana", the program will print "No result", since there is no such name in <https://www.ryerson.ca>.

0.1.6 Screenshots and Results from the Program

```
Running page rank
itr num 0 running page rank
itr num 1 running page rank
itr num 2 running page rank
itr num 3 running page rank
itr num 4 running page rank
itr num 5 running page rank
itr num 6 running page rank
itr num 7 running page rank
itr num 8 running page rank
itr num 9 running page rank
itr num 10 running page rank
itr num 11 running page rank
itr num 12 running page rank
itr num 13 running page rank
itr num 14 running page rank
itr num 15 running page rank
itr num 16 running page rank
itr num 17 running page rank
itr num 18 running page rank
itr num 19 running page rank
itr num 20 running page rank
itr num 21 running page rank
itr num 22 running page rank
itr num 23 running page rank
itr num 24 running page rank
itr num 25 running page rank
itr num 26 running page rank
itr num 27 running page rank
itr num 28 running page rank
itr num 29 running page rank
itr num 30 running page rank
itr num 31 running page rank
itr num 32 running page rank
itr num 33 running page rank
itr num 34 running page rank
itr num 35 running page rank
itr num 36 running page rank
itr num 37 running page rank
itr num 38 running page rank
```

Figure 1: Page Rank

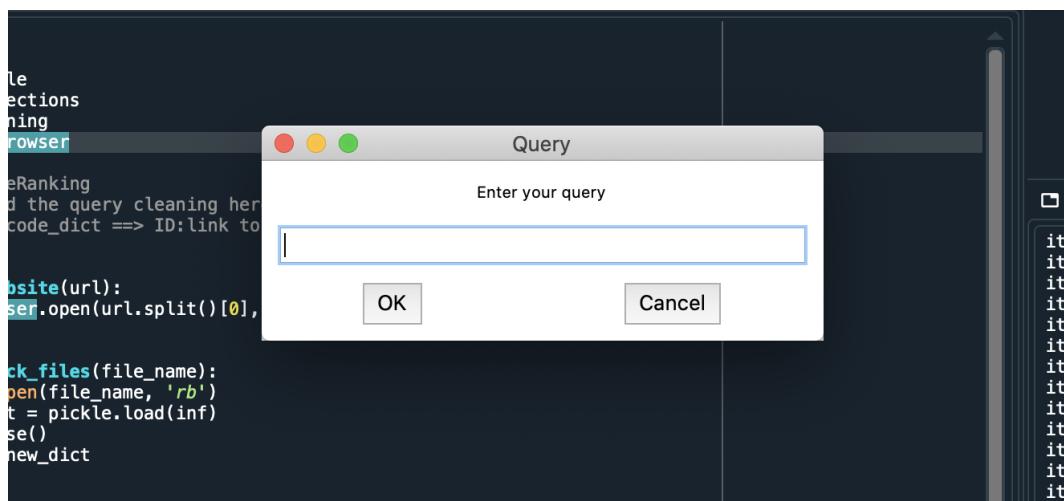


Figure 2: Query UI

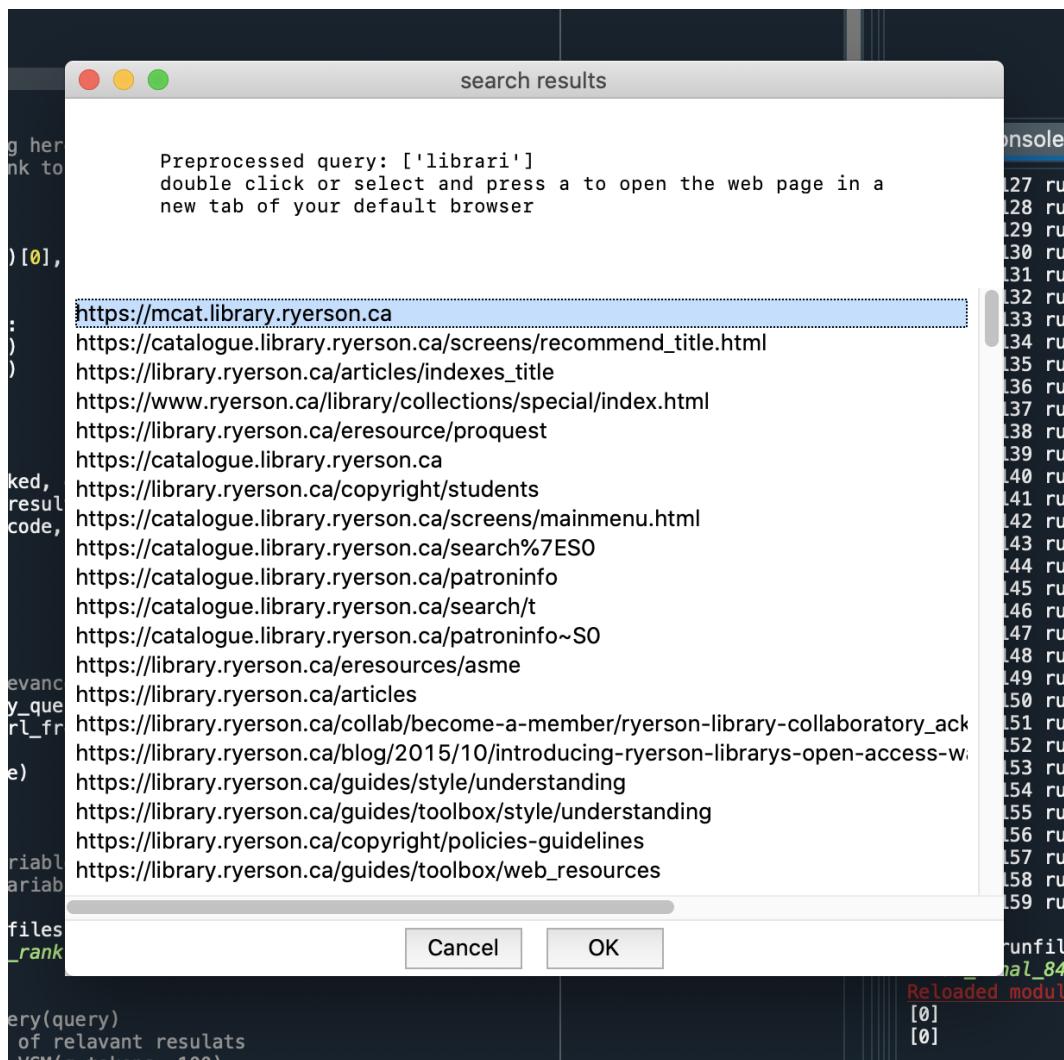


Figure 3: Search Results



Figure 4: Web Results

```

fix make it variable
fix make it variable
cs = list()
e = read_pick_files
ck_files("page_rank"
.ask_query()
ng.Parser()
p.tokenize_query(q)
nge the number of r
core = VSM.run_VSM(
0] for e in vsm_doc
= [pg[e] for e in d
range(len(docs)):
ore = pg_scores[i]*w1+vsm_docs_score[i][1]*w2
(docs[i], hyb_score)
ore_docs.append(tmp)

new = sorted(
ore_docs, key=lambda tup: tup[1], reverse=True)
[e[0] for e in doc_score_new]
w_query(docs_new, q_tokens, url_from_code)
y_query_results(docs_new, url_from_code, q_tokens)

e_docs = list()
UI.ask_query()
== None:
()
ning.Parser()
= p.tokenize_query(query)

```

```

Usage
Here you can see the front end of the system

Console 5/A
itr num 127 running page rank
itr num 128 running page rank
itr num 129 running page rank
itr num 130 running page rank
itr num 131 running page rank
itr num 132 running page rank
itr num 133 running page rank
itr num 134 running page rank
itr num 135 running page rank
itr num 136 running page rank
itr num 137 running page rank
itr num 138 running page rank
itr num 139 running page rank
itr num 140 running page rank
itr num 141 running page rank
itr num 142 running page rank
itr num 143 running page rank
itr num 144 running page rank
itr num 145 running page rank
itr num 146 running page rank
itr num 147 running page rank
itr num 148 running page rank

```

Figure 5: Query word 'food'

The screenshot shows an IPython notebook interface. A modal window titled "search results" displays a list of URLs related to food security at Ryerson University. The list includes various course pages like FNY 405, FNY 403, and FNY 402, along with news articles and general food-related information. The background shows the code used to generate the search results.

```

# fix make it variable
docs = list()
_code = read_pickles('page_rank')
d_pickles('page_rank')
UI.ask_query()
cleaning.Parser()
ns = p.tokenize_query(q)
change the number of n
cs_score = VSM.run_VSM([e[0] for e in vsm_docs])
cores = [pg[e] for e in d in range(len(docs))]:
p_score = pg_scores[i]
p = (docs[i]), hyb_score
w_score_docs.append(tmp)

ore_new = sorted(
w_score_docs, key=lambda
ew = [e[0] for e in doc
show_query(docs_new, d
play_query_results(doc
ue):
score_docs = list()
y = UI.ask_query()
query == None:
exit()
cleaning.Parser()
kens = p.tokenize_query()
x: change the number of
docs_score = VSM.run_VS
= [e[0] for e in vsm_d
cores = [pg[e] for e in i
in range(len(docs))]:
hyb_score = pg_scores[i]
tmp = (docs[i]), hyb_sco
new_score_docs.append(t

score_new = sorted(
new_score_docs, key=lam
new = [e[0] for e in d
handle_show_query(docs_
== False:
exit()

== "__main__":

```

Figure 6: Search Result for 'food'

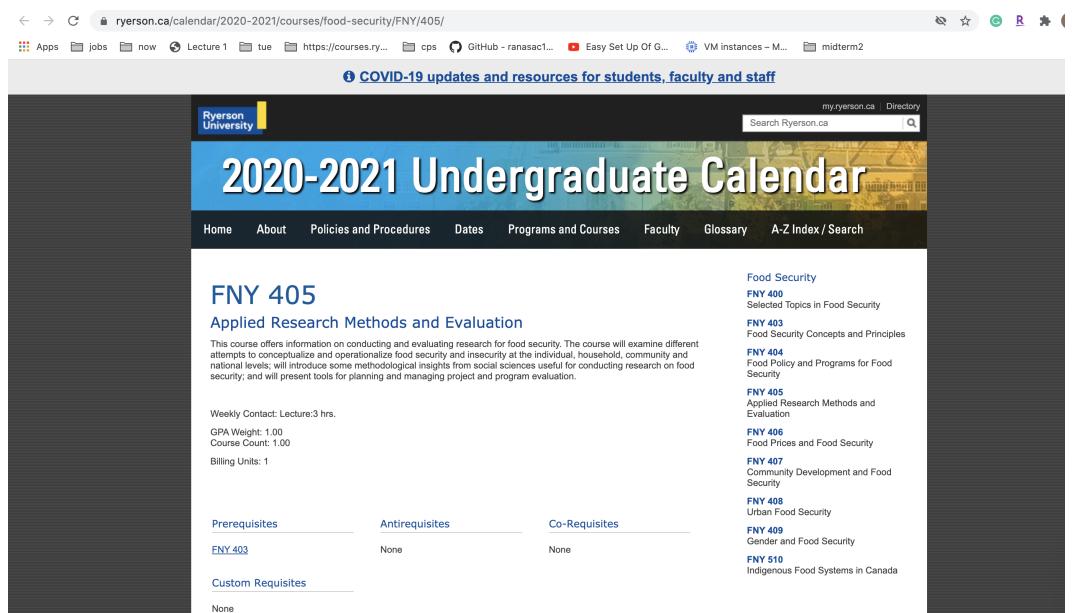


Figure 7: Web Result for 'food'

```

page_rank.py  search.py  page_rank.py  UI.py  m_invert.py  graph.py
  fix make it variable
# fix make it variable
docs = list()
ode = read_pick_files()
ick_files("page_rank")
I.ask_query()
ing.Parser()
= p.tokenize_query(q)
ange the number of r
score = VSM.run_VSM(
:[0] for e in vsm_doc
; = [pg[e] for e in d
range(len(docs)):
core = pg_scores[i]*w1+vsm_docs_score[i][1]*w2
(docs[i], hyb_score)
core_docs.append(tmp)

core_docs = sorted(
core_docs, key=lambda tup: tup[1], reverse=True)
= [e[0] for e in doc_score_new]
ow_query(docs_new, q_tokens, url_from_code)
ay_query_results(docs_new, url_from_code, q_tokens)
:
ore_docs = list()
I.ask_query()
ry = None:
t()

```

```

itr num 128 running page rank
itr num 129 running page rank
itr num 130 running page rank
itr num 131 running page rank
itr num 132 running page rank
itr num 133 running page rank
itr num 134 running page rank
itr num 135 running page rank
itr num 136 running page rank
itr num 137 running page rank
itr num 138 running page rank
itr num 139 running page rank
itr num 140 running page rank
itr num 141 running page rank
itr num 142 running page rank
itr num 143 running page rank
itr num 144 running page rank
itr num 145 running page rank
itr num 146 running page rank
itr num 147 running page rank

```

Figure 8: Searching 'course calender'

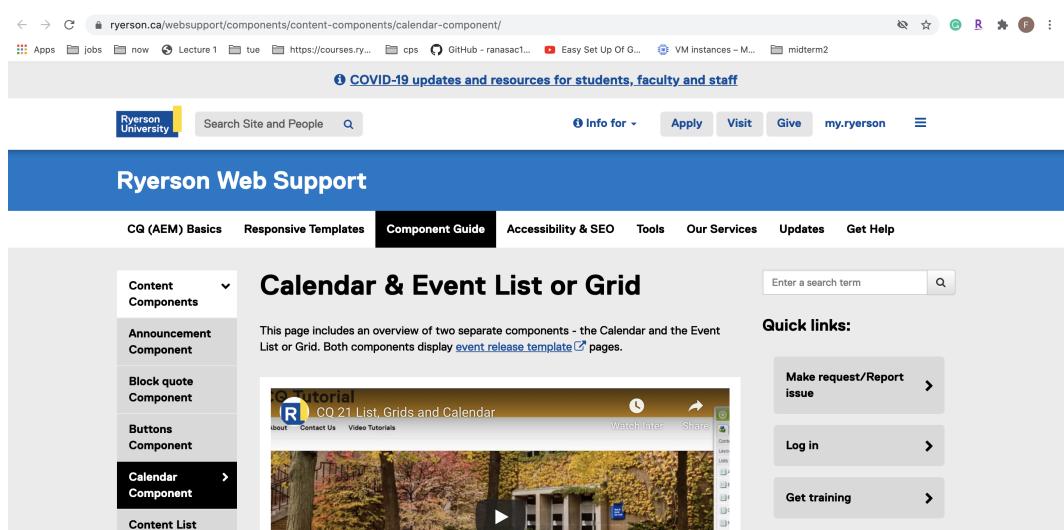


Figure 9: Web Result for 'course calender'

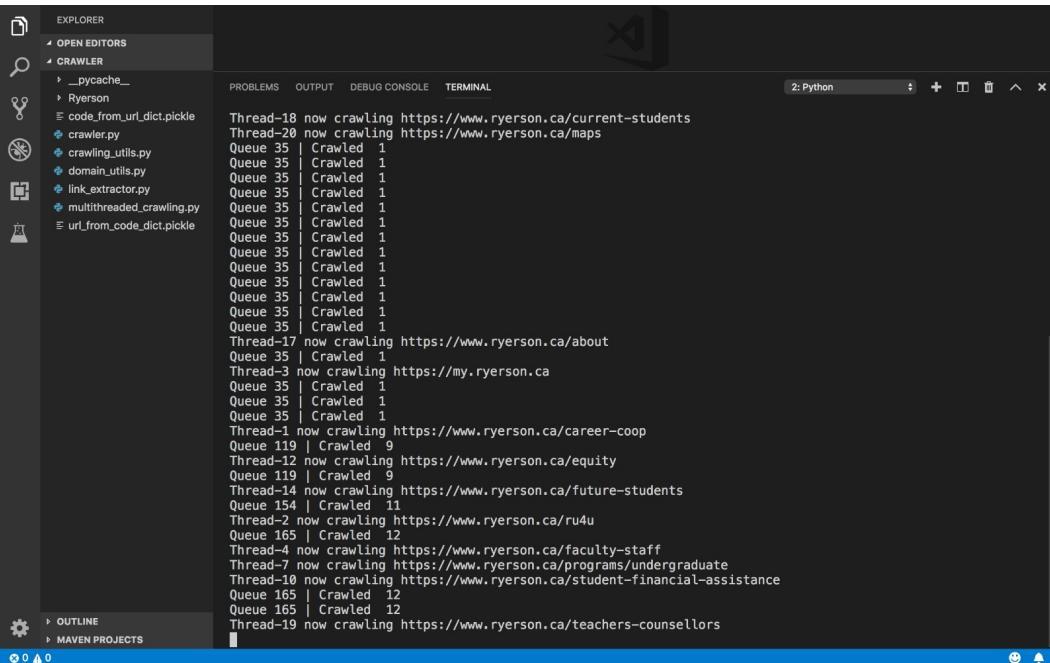


Figure 10: Crawler