CS242 Project-PartA: Web crawling and Lucene indexing & search on Wikipedia

JANNAT ARA MEEM, SAKIB FUAD, SAKSHAR CHAKRAVARTY, and TOMAL MAJUMDER

1 INTRODUCTION

This is a general knowledge related search engine based on Wikipedia data that searches through indexes generated by Lucene java library. The datasets are collected by manually crawling through Wikipedia pages using JSoup. The crawled data are related to world war, politics, novels, football and sushi.

2 WEB CRAWLER

2.1 Crawling Architecture & Strategy

We have implemented a web crawling program to find and download Wikipedia pages from given seed URLs. The crawler has been written in Java. JSoup library has been used to fetch document from URL and parse the document for getting different components like title, body, and images. The pseudocode of our crawling strategy has been outlined in Algorithm 1. First, we have stored a global HashSet, *wikiPageTitleList* to store the title of the wikipedia pages (Line 1) and declared a global variable *maxDepth*(integer) to limit the processing of URLs to specific depth. We have stored the titles of the Wikipedia pages to make sure that there will not be more than one downloaded Wikipedia pages having the same title. Assuming that the title of the page can identify a Wikipedia page uniquely, this hashing strategy have ensured an efficient crawling by eliminating duplicate pages.

Algorithm 1 Crawler

```
1:  Global State:
2:   HashSet<String> wikiPageTitleList
3:   Integer maxDepth
4:  procedure CrawlWikiPage(url, depth):
5:  document ← retrieveDocumentFromURL(url)
6:  title ← retrieveTitleFromDocument(document)
7:  if not wikiPageTitleList.contains(title) and depth ≤ maxDepth then
8:      text ← retrieveTextFromDocument(document)
9:      storeDocumentToDisk(text)
10:     images ← retrieveImagesFromDocument(document)
11:     storeImagesToDisk(images)
12:     for each url in parse(text) do
13:         CrawlWikiPage(url, depth + 1)
14:     end for
15: end if
16: end procedure
```

The procedure of the crawling starts with taking an URL and current depth value as arguments (Line 4). First we have retrieved the document and title of the document from the given URL (Lines 5-6). If the HashSet does not contain the tile and current depth value does not exceed the maxDepth value (Line 7), the text and images from the document

Authors' address: Jannat Ara Meem, jmeem001@ucr.edu; Sakib Fuad, sfuad001@ucr.edu; Sakshar Chakravarty, schak026@ucr.edu; Tomal Majumder, tmaju002@ucr.edu.

are retrieved and stored them on our disk (Lines 8-11). After that, we have parsed the text of the document to find the link tags that might contain other useful URLs to fetch. In short, we fetched all the links of other wikipedia pages contained in the document's text. Next for each retrieved URL, we called our crawling procedure recursively setting crawling depth of that URL as depth+1.

2.2 Crawling Statistics

Figure 1 shows statistics of our crawled pages. Due to our time and resource constraints, all of our group members have run the crawling program separately in our personal computers with different seed URLs. We have downloaded the Wikipedia pages (texts and images) on five different topics: Novel, Politics, Sushi, Football and World War. In total, 13,806 text files and 274,106 image files having total size nearly 3GB data has been stored in our Dataset.

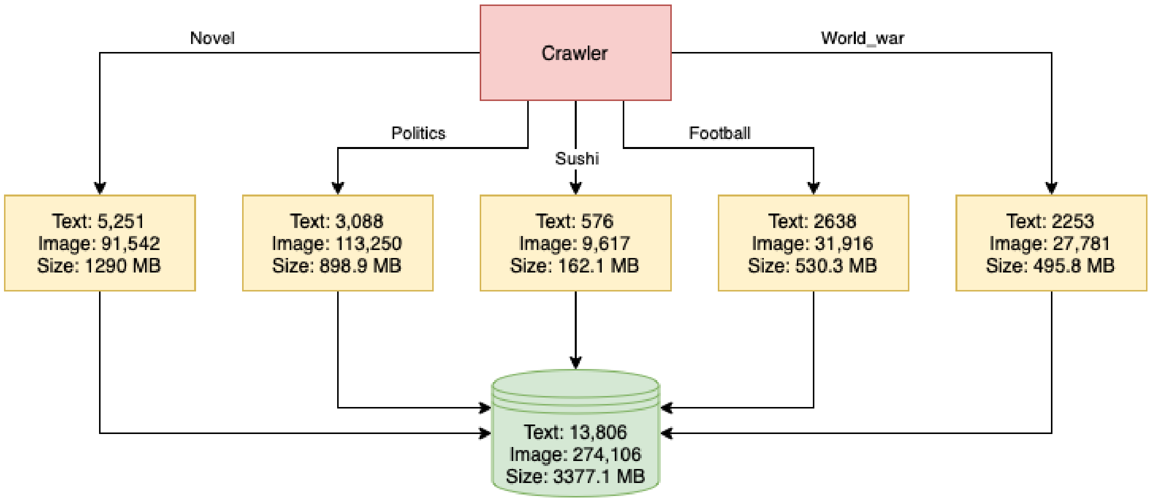

Fig. 1. Crawler statistics

3 LUCENE INDEXING & SEARCH

Lucene is a java library that offers powerful indexing and search features for text data. Following the guideline for this project, we have used this library on our crawled text data to generate index files and then to retrieve documents based on the queries.

3.1 Indexing

We have used both the default analyzer in Lucene that is “StandardAnalyzer" and “EnglishAnalyzer". “StandardAnalyzer" builds an analyzer with the default set of stop words provided in Lucene library. In addition to it, “EnglishAnalyzer" also performs stemming. The reason behind analyzing with “EnglishAnalyzer" is that performing some stemming on documents containing plain English texts can enhance the performance of the document retrieval process. Each of the documents in our dataset has two fields; title and content. We pass the fields and our crawled data to “IndexWriter" that generates index files using inverted index technique and store them in the disk.

Wikipedia

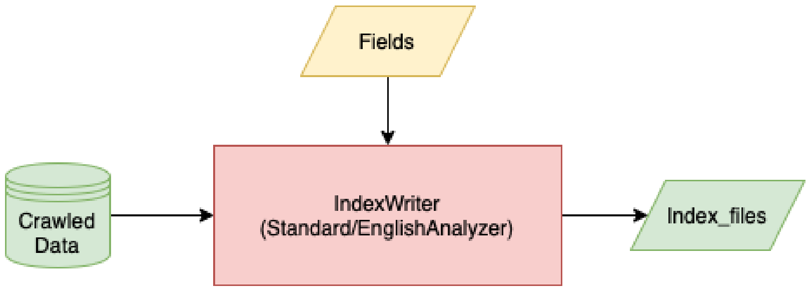

Fig. 2. Flow-chart for Lucene indexing

3.2 Search

The search and document retrieval is handled by “QueryParser". It takes the query, q and the number of top ranked pages to retrieve, k as inputs and also loads the previously generated index files from disk. Then it applies a relevance model to rank documents and shows top k of them in the output. Here, we have used five different relevance models namely tf-idf (default), boolean similarity, BM25, language model with Jelinek-Mercer and Dirichlet smoothing. We provide a comparison among these models in section 4.

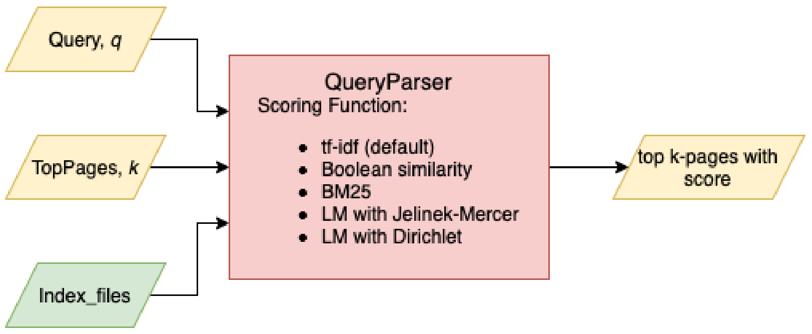

Fig. 3. Flow-chart for Query search

4 PERFORMANCE ANALYSIS & COMPARISON

4.1 Performance analysis of lucene indexing

In figure 4, we plot the time required for indexing our crawled data using two different analyzers by gradually increasing the number of documents to analyze by a thousand at each step. The plot shows that “EnglishAnalyzer" spends a little more time than “StandardAnalyzer" to perform the same task. This is expected since “EnglishAnalyzer" does both stop word removal and stemming whereas “StandardAnalyzer" does stop word removal only.

4.2 Comparison among relevance models

We have shown the top results and their comparisons for two different queries in this section. We have shown the titles of the top 5 web pages ranked on 5 different relevance models for the query: "UC Education System" in Table 1. We have observed that all the models except the Boolean Similarity model returns relevant web pages and their

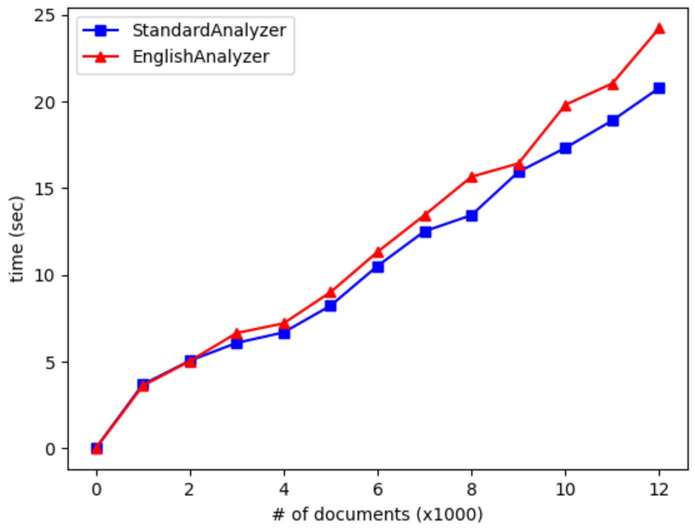

Fig. 4. A plot showing the time required for indexing vs. the number of documents used

performances are comparable. The Boolean Similarity model gives all the web pages the same score and returns mostly irrelevant web pages.

Next, we have shown the comparison among the results for different search indexing fields for the query: "Sushi Japan" in Table 2, 3 and 4. We have observed that we get the highest scores while searching on both the index i.e. "title" and "content". For example, the highest ranked document for tf-idf model gets scores 4.64, 6.91 and 11.55 when indexed on "title", "content" and both "title", "content" respectively. From our observation, we can conclude that the most relevant results are found when the documents are indexed on both the fields.

5 OBSTACLES & DISCUSSION

We had to overcome some obstacles in the crawling phase of our project. We have discussed those in this section.

5.1 Duplicate Elimination

We used several approaches for eliminating duplicate page crawling in our search engine. The approaches and their corresponding limitations are as follows:

(1) **URL-based:** First, we used a URL-based hashset for duplicate elimination. So, the crawler never crawled the web page with an URL that has been found before. However, we found that different URLs can refer to the same page. For example, the crawler treated the following two pages:
https://en.wikipedia.org/wiki/Civil_War and
https://en.wikipedia.org/wiki/Civil_War#disambiguation
as different URLs although both of these URLs refer to the same page. So, the crawler kept crawling the same page over and over again in different levels of the depth. This may have resulted in a never-ending loop.

Wikipedia

Relevance Model	Score	Web page Title
tf-idf (default)	7.0196576	University of California, Riverside - Wikipedia
	6.565757	University of California, Los Angeles - Wikipedia
	6.2392716	University of California, Berkeley - Wikipedia
	5.798729	Troubadour - Wikipedia
	5.730804	Medical anthropology - Wikipedia
LM with Jelinek-Mercer	19.898636	Higher Education Statistics Agency - Wikipedia
	19.326626	Education in Albania - Wikipedia
	19.30302	Compulsory education - Wikipedia
	19.25279	EFMD Quality Improvement System - Wikipedia
	19.145214	Education - Wikipedia
LM with Dirichlet	8.965622	University of California, Riverside - Wikipedia
	7.798186	University of California, Los Angeles - Wikipedia
	7.556681	University of California, Berkeley - Wikipedia
	7.356885	Public university - Wikipedia
	7.131752	Los Alamos National Laboratory - Wikipedia
BM25 Similarity	7.0196576	University of California, Riverside - Wikipedia
	6.565757	University of California, Los Angeles - Wikipedia
	6.2392716	University of California, Berkeley - Wikipedia
	5.798729	Troubadour - Wikipedia
	5.730804	Medical anthropology - Wikipedia
Boolean Similarity	3.0	University of California, Berkeley - Wikipedia
	3.0	Higher education - Wikipedia
	3.0	School shooting - Wikipedia
	3.0	Pakistan - Wikipedia
	3.0	Texas Education Agency - Wikipedia

Table 1. Comparison among relevance models for query: "UC Education System" with top 5 hits, search index : "title", "content"

Relevance Model	Score	Web page Title
tf-idf(default)	4.641589	Sushi - Wikipedia
	4.050813	YO! Sushi - Wikipedia
	2.792059	Japan - Wikipedia
LM with Jelinek-Mercer	11.053236	Sushi - Wikipedia
	10.647779	YO! Sushi - Wikipedia
	7.8347406	Japan - Wikipedia
LM with Dirichlet	2.0804815	Sushi - Wikipedia
	2.0799823	YO! Sushi - Wikipedia
	0.24637087	Japan - Wikipedia
BM25 Similarity	4.641589	Sushi - Wikipedia
	4.050813	YO! Sushi - Wikipedia
	2.792059	Japan - Wikipedia
Boolean Similarity	1.0	Video games in Japan - Wikipedia
	1.0	Japan during World War I - Wikipedia
	1.0	Japan Maritime Self-Defense Force - Wikipedia

Table 2. Comparison among relevance models for query: "Sushi Japan" with top 3 hits, search index : "title"

Relevance Model	Score	Web page Title
tf-idf(default)	6.911422	Sushi - Wikipedia
	6.667172	Japanese cuisine - Wikipedia
	6.2095485	California roll - Wikipedia
LM with Jelinek-Mercer	15.024651	Sushi - Wikipedia
	15.002071	Odori ebi - Wikipedia
	13.847268	Wasabi - Wikipedia
LM with Dirichlet	9.94124	Sushi - Wikipedia
	8.860995	Japanese cuisine - Wikipedia
	7.6018033	Japanese rice - Wikipedia
BM25 Similarity	6.911422	Sushi - Wikipedia
	6.667172	Japanese cuisine - Wikipedia
	6.2095485	California roll - Wikipedia
Boolean Similarity	2.0	Japanese diaspora - Wikipedia
	2.0	Shiso - Wikipedia
	2.0	Globalization - Wikipedia

Table 3. Comparison among relevance models for query: "Sushi Japan" with top 3 hits, search index : "content"

Relevance Model	Score	Web page Title
tf-idf(default)	11.553011	Sushi - Wikipedia
	9.170557	YO! Sushi - Wikipedia
	7.1262007	Culture of Japan - Wikipedia
LM with Jelinek-Mercer	26.077887	Sushi - Wikipedia
	21.177158	YO! Sushi - Wikipedia
	19.714666	Culture of Japan - Wikipedia
LM with Dirichlet	12.021723	Sushi - Wikipedia
	8.946575	YO! Sushi - Wikipedia
	8.860995	Japanese cuisine - Wikipedia
BM25 Similarity	11.553011	Sushi - Wikipedia
	9.170557	YO! Sushi - Wikipedia
	7.1262007	Culture of Japan - Wikipedia
Boolean Similarity	3.0	Sushi - Wikipedia
	3.0	Culture of Japan - Wikipedia
	3.0	Japan - Wikipedia

Table 4. Comparison among relevance models for query: "Sushi Japan" with top 3 hits, search index : "title","content"

(2) **Wikidata-based:** For overcoming the issue with the first approach, we used one of the content properties of the web page for eliminating duplicates. We found out that each wikipedia web page has a unique wikidata indentifier URL. The pattern of such unique URLs are :
"https://www.wikidata.org/wiki/Special:EntityPage/**unique_id**". The **unique_id** in the URL is unique for each web page. So, we used a wikidata-based hashset where we stored the unique URL of the wikidata. In this approach, we had to match the aforementioned pattern with all the hyperlinks of the entire DOM tree of each page in order to find the wikidata URL. So, crawling with this approach was too time consuming.

(3) **Title-based:** We have come up with our final approach in order to crawl more efficiently and at the same time maintain the correctness in eliminating duplicates. We manually checked several wikipedia web pages and

found that two different web pages do not have the same title. Hence, we have used the title-based hashset for duplicate elimination. This approach is way faster than the previous one and never crawls a duplicate page.

5.2 Out-of-domain Pages removal

We have only crawled the pages that belong to the **wikipedia.org** domain. In order to do so, all the pages out of the domain had to be filtered out. We used pattern matching for this purpose. So, the crawler only crawls the pages whose URL match with the pattern *r/ ^ https://en.wikipedia.org/*.

6 LIMITATION

Our crawler depends on the assumption that each web page has a unique title. We found no evidence that disproves our assumption. So, we can arguably state that our assumption is true. However, there may have been some web pages that did not come to our attention. In that case, our crawler may have treated two different web pages as duplicates and the second web page would never have been crawled.

7 EXECUTION LOGISTICS

We have three bash scripts named crawler.sh, lucene_indexing.sh, and lucene_search.sh. Their functionalities are discussed below:

crawler.sh This script starts crawling the web pages. It takes four command line arguments. The arguments are seed_path, max_depth, max_page_count, and, output_dir_path. For example, the command to run this script is:

```
sh crawler.sh https://en.wikipedia.org/wiki/Novel 4 100000 output
```

lucene_indexing.sh It generates the index files from the given documents. It asks the user input for analyzer options. We are working with two types of analyzers, one is standard analyzer and another one is english analyzer. The command to run this script is:

```
sh lucene_indexing.sh
```

lucene_search.sh This script outputs top k results of the given query using the indexed files. It takes the query, analyzer_option and totalHitCount (k) as arguments. One example to run this script is given below:

```
sh lucene_search.sh "UC Education System" 2 5
```

8 CONTRIBUTION

- Crawler: Jannat Ara Meem and Sakib Fuad
- Lucene: Sakshar Chakravarty and Tomal Majumder
- Report: All group members contributed equally