CS242: Information Retrieval and Web Search

Project Report

1. Collaboration Details:
2. Abhishek Sharma: Lucene Indexing implementation
3. Gaurav Ratnakar: Lucene Indexing implementation
4. Neil Castellino: Web crawler implementation
5. Rajat M. Jain: Web crawler implementation
6. Overview of the crawling system:

Our crawler crawls website of [The New York Times](https://www.nytimes.com/) for news articles and stores them to our data set.

1. Architecture:
2. main.py:

This file is the top-level code or the entry point which acts as a driver and runs our crawler program. It imports our crawler module and initializes some important variables like:

first\_article\_link: The homepage of The New York Times

url\_per\_file: Which stores the URLs collected from the page

url\_limit = The stopping condition for our crawler

1. crawler.py:

Our architecture of the program can be divided into several sections, each handling one significant component of the program:

1. Data Writing: \_\_write\_\_json() and \_\_write\_\_urls() write the data object and URLs repectively.
2. Relevant URL extraction: \_\_remove\_excess() removes all the URLs that do not begin with <http://www.nytimes.com>, since they do not contribute to our aim of crawling NYTimes.
3. Visited URL tracking and continuation: \_\_gret\_visited\_urls() help the program to return to the point where the flow was terminated if it was.
4. Crawling: \_\_extract\_href(), \_\_parse\_data() and \_\_crawl\_website() are the functions that majorly handling the crawling segment of the code.
5. Duplicates removal:
6. The Crawling Strategy:

The implementation begins with calling the crawl() method which very firstly initializes a queue. This queue is for the program to keep track of the URLs we visit and begin back from the last URL in cases where the program crashes or runs into errors resulting the termination of the process. The program then checks for the start\_url which when none, triggers to begin the program with the homepage or if start\_url is not none we get the list of the href for that starting URL.

After completing checks for where the crawler is with crawling, the program calls the \_\_crawl\_website() function which uses BeautifulSoup to get important information like the title, article summary and article body. These attributes along with the article date, article ID and article URL is combined into a JSON object and appending to our collection.

Data articles from the files are stored every time we finish visiting 100 URLs and the upper bound for visited URLs is 150000. This approach helped us gather more than 1Gb of data.

1. Overview of the Lucene indexing strategy
2. Limitations:
3. Limited search results due to limited data.
4. Obstacles:
5. Duplicate articles: We soon noticed that some of the articles were duplicates and were not taken care of since they had the same content but different URLs. Hence we modified the code to remove those duplicates.
6. Crawling limitations: NYTimes did not let us crawl the websites initially for more than 181 times.
7. Instruction on how to deploy the crawler:
8. Instruction on how to build the Lucene Index: