**CS 242: Information Retrieval & Web Search Winter 2023**

Course Project: Build a Search Engine (Part A)

**‘PatentSearch’**

*a Patent Search Engine*

Project team members**:** Team 16

* Kaushik Daiv [netID - kdaiv001, SID - 862393181]
* Mrunali Lachake [netID - mlach008, SID - 862394872]
* Amrutha Alewoor [netID - aalew002, SID- 862395063]
* Akash Bilgi [netID - abilg003, SID - 862395080]
* Anish More [SID - 862324523 ]

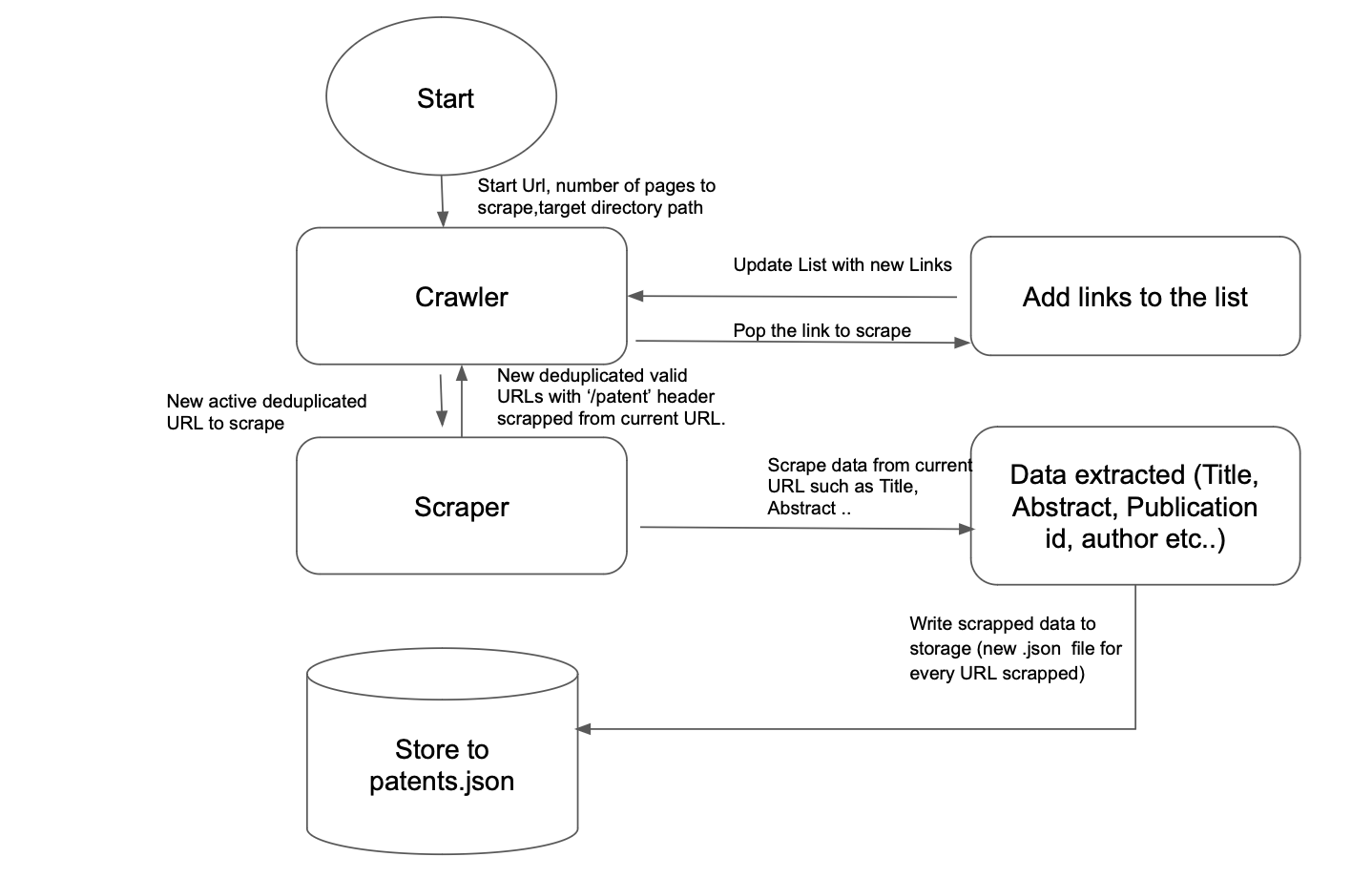
**Collaboration Details**

The tasks were distributed and undertaken as follows:

* **Kaushik Daiv:** Designed and created a web crawler as depicted in the architecture in the report. This crawler crawls the links added in the spider queue. Contributed in collection of 1 GB data in patents.json file.
* **Mrunali Lachake:** Implemented the scraper to extract data from page url using Scrapy. The data extracted are title, url, author name, abstract, date. Contributed in collection of 1 GB data in patents.json file.
* **Anish More:** Contributed in trying out different strategies to build out the web crawler.
* **Amrutha Alewoor**: Implementing lucene query, tokenizer, trying out analyzers, content retrieval by parsing the indexed database, lucene Indexing
* **Akash Bilgi:** Implementing lucene indexing, designing input json, parsing and query integration, handling meta fields and, error handling, time elapsed, graphical representation.
* **Common task:** part-A report

**Overview of the crawling system**

1. Architecture



1. Objective:

Our objective is to scrape patents from Google Patents - [**https://patents.google.com**](https://patents.google.com/patent/) on a variety of fields under the domain of Information Retrieval.

1. The crawling strategy

To begin with, we are using Scrapy for crawling the web and scraping the desired data. Scrapy is a Python-based open-source web crawling framework that is capable of extracting data from a wide range of websites, including HTML pages, XML, and JSON. It is specifically designed to handle large-scale web crawling tasks efficiently, making it a popular choice for web scraping and data extraction. In our project, the spider crawler that we have programmed using Scrapy visits the initial seed URL and extracts various pieces of data such as the patent’s title, publication ID, abstract, inventor names, self URL, publication date. In addition to this, crawler further scrapes all the URLs on the current webpage starting with the domain [**https://patents.google.com/patent/**](https://patents.google.com/patent/) for later processing. This list is passed to the follow\_all() method provided by Scrapy. follow\_all() is a method that can be used in a spider to follow links on a page and send requests for each of those links. By default, Scrapy's follow\_all method automatically filters out duplicate URLs, so a link that has already been visited by the spider will not be followed again. This is done to avoid duplication of patents and prevent the spider from getting stuck in a loop by endlessly following the same links.

The scraped data is yielded in dictionary format and added to a JSON file, which is then used as input to the Lucene application.

1. Basic paradigms
   1. Multi-threading:

Scrapy is a single-threaded crawler and does not support multi-threading.

* 1. Handling duplicate URLs:

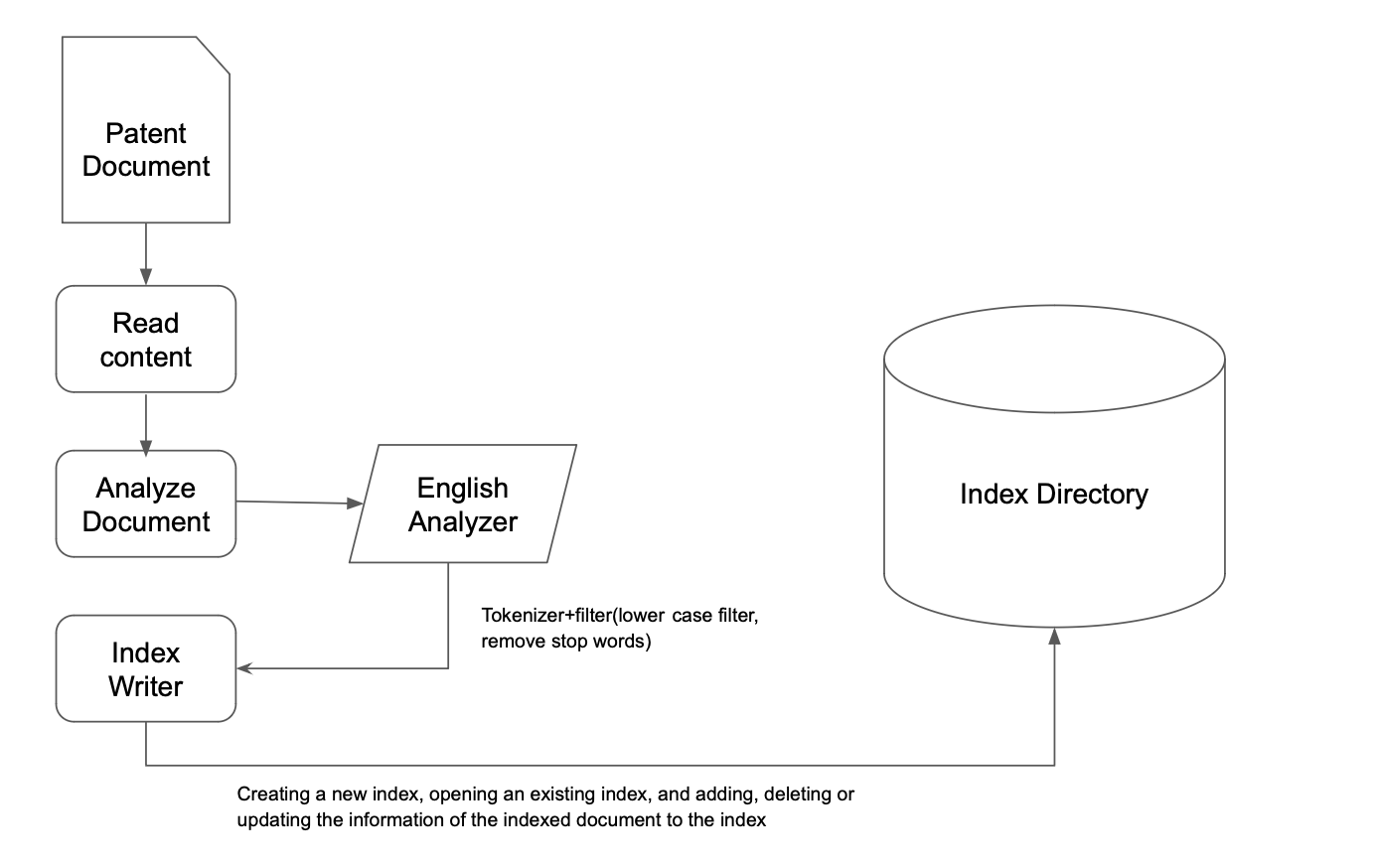
We are using the inbuilt “follow\_all” method of the Scrapy to crawl the next page. This method of Scrapy handles the duplicate URLs and does not allow crawling the same URL again.

* 1. Handling the dirty URLs:

Scrapy by default handles the dirty URLs. In Scrapy HttpErrorMiddleware is default enabled, which filters out unsuccessful http responses so the spiders don’t have to deal with them.

**Overview of the Lucene indexing system:**

1. Architecture



1. The indexing strategy

Using Pylucene to index Google Patents involves several steps, including document parsing, tokenization, and indexing. Here is an overview of the process:

**Document Parsing:** The first step is to parse the patent documents in order to extract the relevant content. The content may include metadata, such as patent number[unique ID], title, abstract, and inventor information, as well as the actual text of the patent.

**Tokenization:** Once the documents have been parsed, the next step is to tokenize the text into individual terms. Tokenization involves breaking the text into words or phrases that can be indexed and searched later. Pylucene provides a variety of built-in tokenizers, such as the StandardTokenizer, which splits the text into words based on whitespace and punctuation.

**Analysis:** After the text has been tokenized, it is passed through an analyzer to further process the terms. An analyzer can perform a variety of operations on the terms, such as lowercasing, stemming, and removing stop words. In the case of indexing Google Patents, we are using the EnglishAnalyzer, which is a built-in analyzer that applies a set of English-specific processing rules to the text.

**Indexing:** After the text has been tokenized, it can be indexed using Pylucene. The indexing process involves creating an inverted index, which is a data structure that maps terms to the documents that contain them. This allows for quick and efficient searching of the patent documents. Pylucene provides a variety of tools for customizing the indexing process, such as defining stop words (common words that are excluded from the index), setting up field-based indexing, and specifying weighting factors for different terms.

1. Fields in the Lucene index for Google patent

* Title - title of the patent, which can provide valuable information about the patent's content
* Abstract - abstract or summary of the patent, which can help users quickly understand the key aspects of the patent
* Publication number - unique identifier for the patent, such as the USPTO Patent Number, the European Patent Office (EPO) number, or the World
* Date - Publication date of the patent
* Inventors - List of people who contributed to the invention

1. Text analyzer choices:

WhitespaceAnalyzer

* divides text at whitespace

SimpleAnalyzer

* divides text at non-letters
* convert to lower case

StandardAnalyzer

* good for most European languages
* removes stop words
* convert to lower case

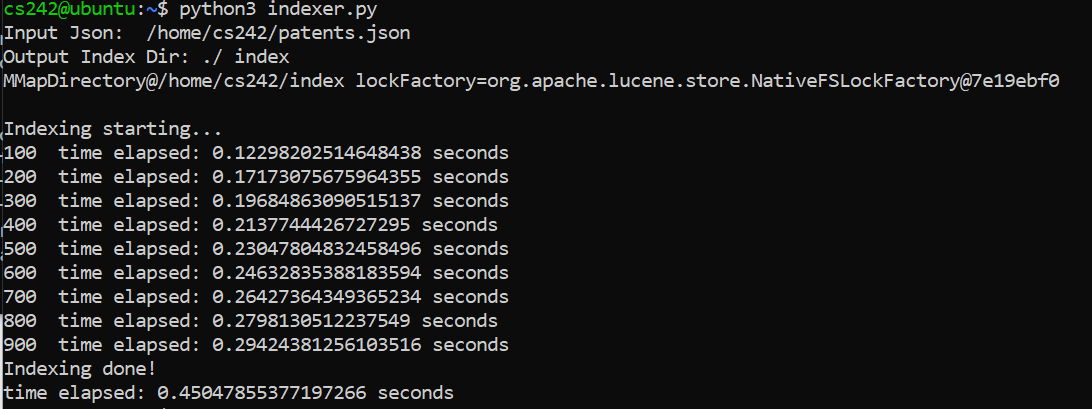
EnglishAnalyzer

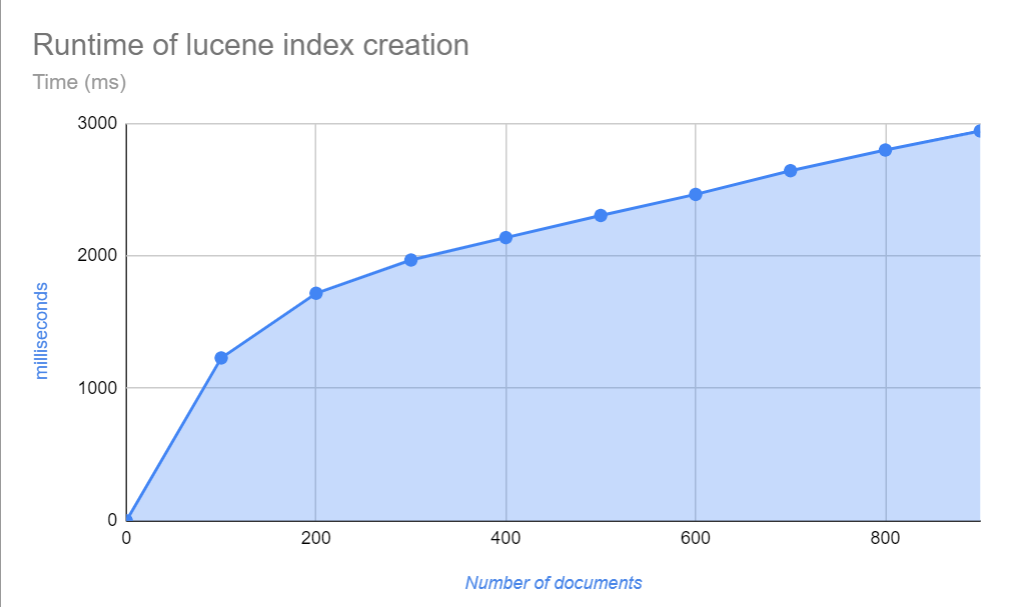
* Good for advanced processing of English data
* Works well with predominantly English data
* Stemming and synonym expansion

Our choice of analyzer is the **English analyzer** as our data is predominantly English, which offers all the functionalities of the Standard Analyzer, but also integrates stemming and synonym expansion. Stemming is the procedure of reducing words to their root form, e.g., converting "running" to "run," which enhances search accuracy by enabling the matching of variations of words.

1. Run time of the Lucene index creation process

Program output of time elapsed per multiple of 100 docs:





**Limitations of the system:**

* Our crawler is single threaded as Scrapy does not support multi-threading
* Google patents contain duplicate patents with different publication numbers as patents have different publication numbers in different countries. This causes different URLs to have the same content.
* Lucene Analyzer: Inventors names are sometimes in different languages(<1%) (ex : Japanese, Chinese, German etc)

**Obstacles and solutions**

* Some of the HTML/CSS tags of the web pages are dynamically created in Javascript which made it difficult to search for the CSS selectors as some tags were different on the web page compared to the scraped web page tags. To solve this, we first saved response.txt on our machine and then searched for the tags to scrap.

**Instruction on how to deploy the crawler**

* Download the zip file patentSpider.zip and extract the content of the zip
* Go to the “patentSpider” folder using the command - “cd patentSpider”
* Give permission to the crawler.sh file to run it using the command - chmod +x crawler.sh
* Run the crawler.sh file using the command - ./crawler.sh. This command will start the crawler and scrap the data from Google Patents
* After successful completion of the script you will be able to see the patents.json file which will contain all the data of patents which the crawler has collected.
* Follow below steps for the indexing part

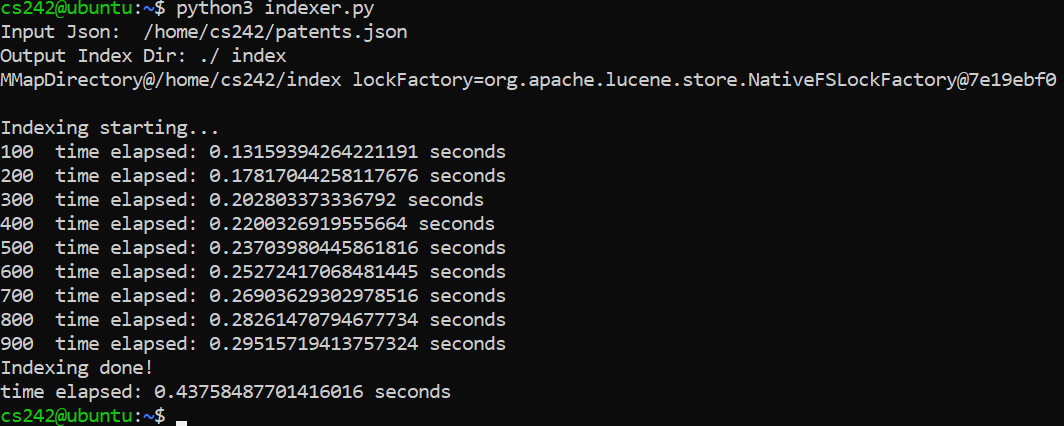
**Instruction on how to build the Lucene index**

* Run the indexer.py script in the patentScraper directory
* python3 indexer.py <op arg1 - query string>

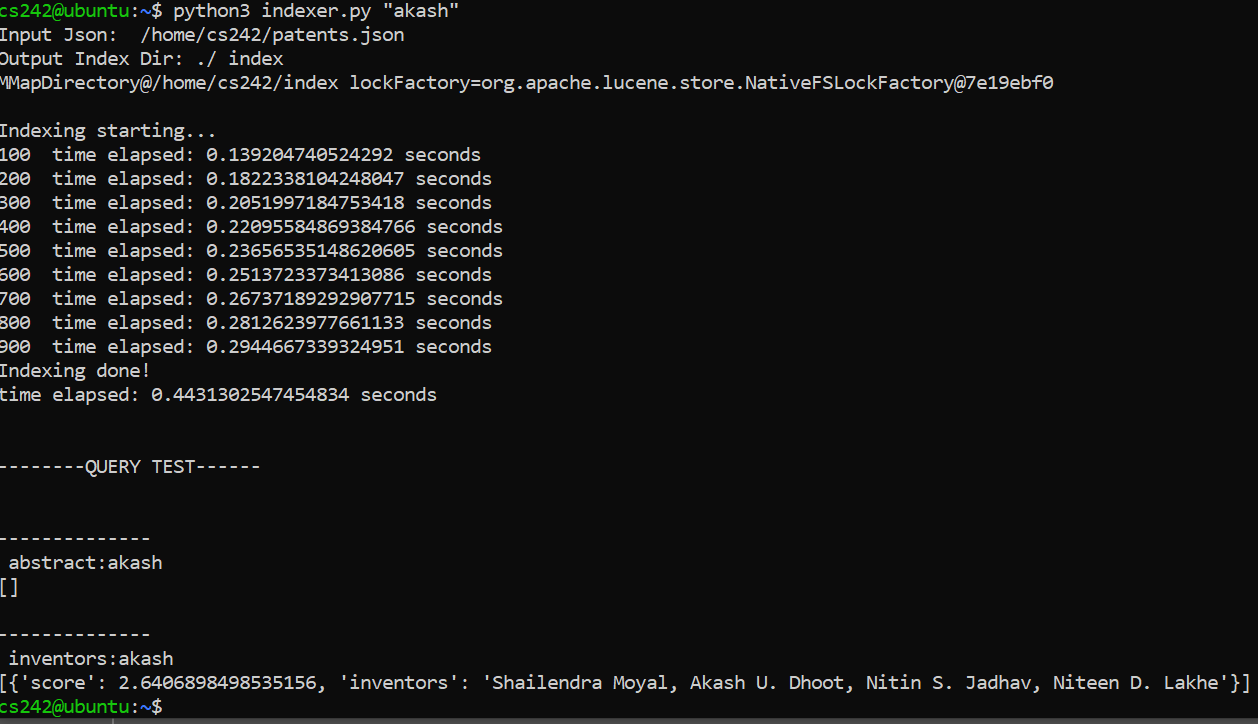
<op arg 1> = optional query string

Examples:

1. python3 indexer.py



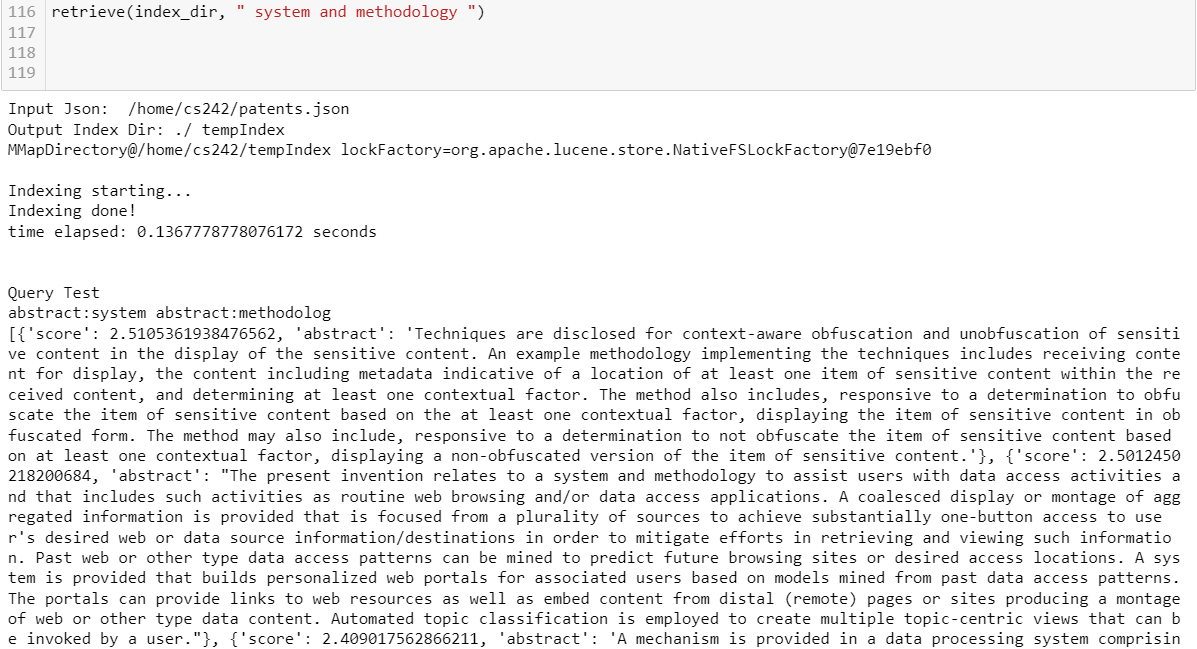
1. python3 indexer.py “akash”



**Sample query screenshot:**

Sample query : “system and methodology” and testing with matching abstract (will further improvise with links and patent numbers)

Or use $python3 indexer.py “QUERY STRING”

****