Journal of Data Acquisition and Processing

ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.7752040

COMPREHENSIVE APPROACH FOR INDEXING AND SEARCHING USING APACHE LUCENE FRAMEWORK

Dipanshu Tomar, Kartik Tyagi, Divyansh Rana, Karan Kumar, Satish Babu
Department of Computer Science and Engineering
Meerut Institute of Engineering and Technology, Meerut, UP, India
{dipanshu.tomar.cs.2019@miet.ac.in, kartik.tyagi.cs.2019@miet.ac.in , divyansh.rana.cs.2019@miet.ac.in, karan.kumar.cs.2019@miet.ac.in , satish.babu@miet.ac.in}

Abstract

The amount of information is expanding exponentially therefore it is very important to develop an efficient and unique way to retrieve them in an effective manner. It is extremely challenging to locate and retrieve significant data from these files or documents due to their nature, rapid speed of expansion, and scattered character. This research speaks about the insertion of incoming data followed by indexing of its contents. Lucene is responsible for document indexing when the search engine is coupled using the specified database or file system.

Keywords: Apache Lucene, Indexing, Searching.

INTRODUCTION

Due to the explosion in information technology, the amount of information at our disposal has been growing tremendously over time. [1] The enormous amount of knowledge and information that is available to people has been greatly influenced by the wide availability of papers in many forms, especially digital form. The original documents were delivered in a variety of formats, including HTML, PDF, and WORD, with no conventions being observed. We refer to them as unstructured documents. With the development in technology, various software products are available in the market. However, employing commercial software solutions frequently guarantees a quick system deployment. I have employed Apache Lucene, an open-source indexing technology. Although implementing Lucene was simple in my experiments, merging the Lucene indexing and the database queries proved to be difficult. [2] My implementation has shown how to build a framework that combines database query methods with Lucene's comprehensive text indexing engine to create a generalized keyword search method. This framework has a lot of uses for small and medium-sized businesses and organizations.

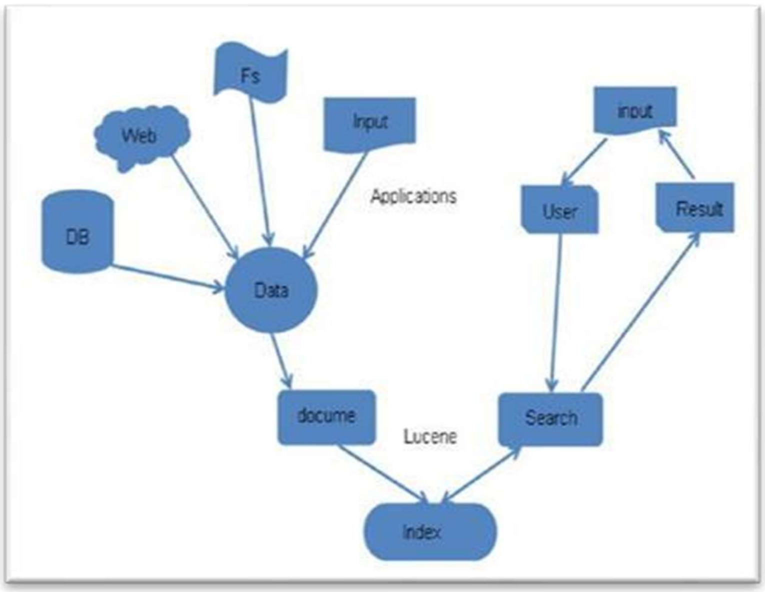


Figure 1 (Comprehensive Architecture)

LITERATURE REVIEW

Lucene Features
[3] Lucene include a set of attributes which are present to be grouped into following sections: entering data and query analysis, indexing, and keeping, search, and plugins (everything else). The first three components make up what is known as Lucene's "core," while the final component is a code library that has been successful in helping people solve search-related issues (such as result highlighting).

Analysis of Language
The analytics features of Lucene oversee converting content, such as documents to be indexed or searchable queries, into a suitable internal representation that may be used as necessary. Parsing creates tokens that facilitate accurate query expressions at query time as well as tokens that are subsequently inserted into Lucene's inverted index at index time. Three tasks are chained together as part of the parsing process to process the received content. (Stop word distance, generated n-grams.)

Indexing and Storage
[5] The following are the main components of Lucene's indexing and storing tier: Many of these features are detailed in the Architecture and Implementation 4302 section. Using the analysis function, each field may or may not be examined.

1. user-defined documents being saved.
2. Near real-time indexing to make documents searchable once they are indexed; lock-free indexing and accounting list data structures.
3. Append and rollback transaction support.
4. Terms, documents, and corpus levels to enable different scoring models.

Querying

Numerous query possibilities are supported by Lucene on the search side and can filter, paginate, sort, and 4200 pseudo-relevance feedback on results. [4] In order to assist developers, create their own query parsers, Lucene offers over 50 different types of query expressions, numerous query parsers, and a query parser framework. On query possibilities, more later. Additionally, pluggable scoring model system that may be modified by developers is now supported by Lucene 4.0.

Lucene Architecture and Lucene Applications Overview

Any application can be given indexing and searching capabilities by utilising Lucene. Any material that can be translated into a text format can be indexed and made searchable. As long as the data can be transformed to text, Lucene is agnostic of the source, format, and even language of the data. This means that Lucene can index and search text data that is stored in files, web pages, on distant web servers, documents saved locally on a computer's file system, basic text files, Microsoft Word documents, HTML or PDF files, or any other format from which textual data may be extracted.

PROCESS FLOW

The below flow chart depicts the basic flow of data throughout the application. The flow chart consists of various modules representing their specific functionality. It consists of the way through which data comes in various formats and gets stored followed by indexing. It also shows how the data gets searched.

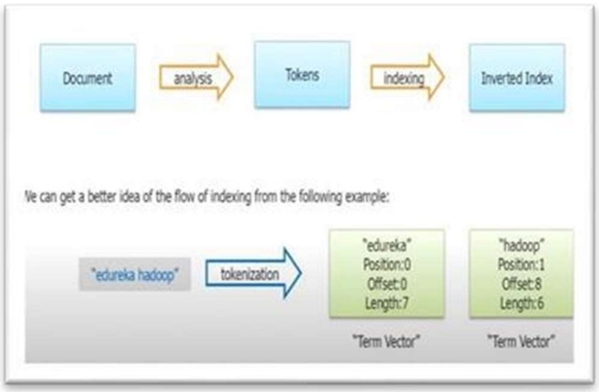

Document
analysis
Tokens
indexing
Inverted Index
We can get a better idea of the flow of indexing from the following example:
"edureka hadoop"
tokenization
"edureka" Position:0 Offset:0 Length:7
"hadoop" Position:1 Offset:8 Length:6
"Term Vector"
"Term Vector"

Figure 2 (Tokenization and Indexing)

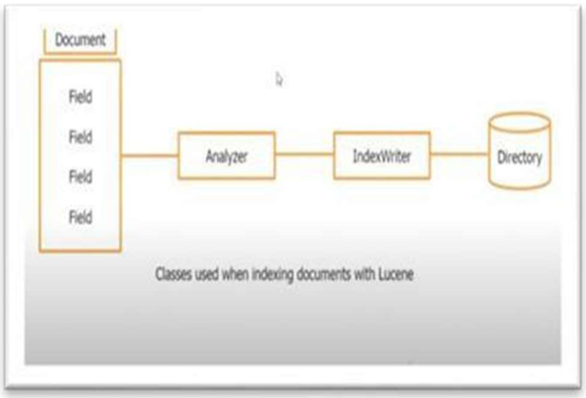


Figure 3 (Writing to index)

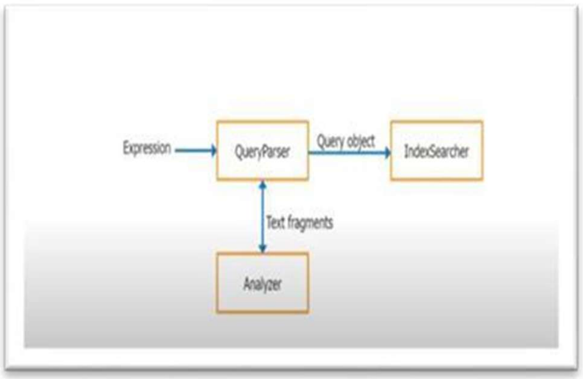


Figure 4 (Searching of Index)

DATA DESCRIPTION

In our project, we are including the unformatted, raw materials that were provided in many file types, including HTML, PDF, and WORD. We refer to them as unstructured documents. Apache Lucene is a Java based library used for the entire text search of documents, and is at the core of many searching based servers. It can also be embedded or indulged into Java webbed backends. While Lucene's configuration options are quite fascinating, they are provided to be used by core databases developers on a generic ocean of text. If the provided data files or documents follow a specific structure or a specific type of content, you can take benefit of either to improve search efficiency and query elasticity. Below is the pictorial representation of how indexing of the given document is done.

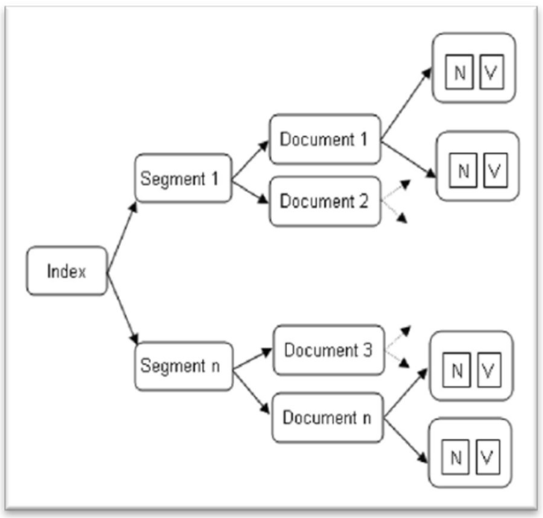
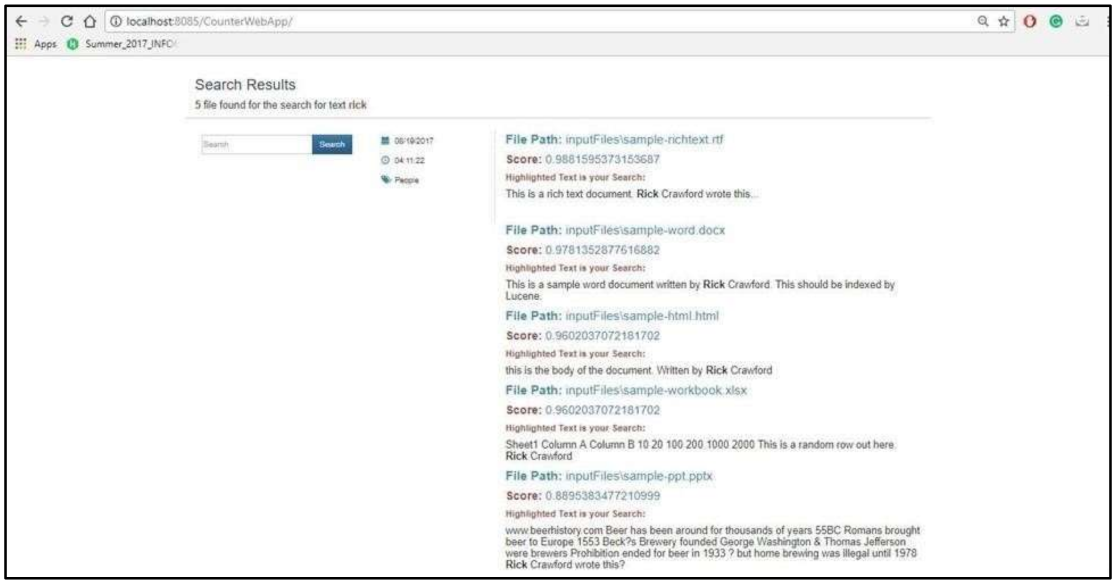


Figure 5 (Indexing of data)

RESULT

Search Results:

The results of a search run through Lucene are presented in a very user-friendly format. All the search results are produced on an HTML page. The search term is highlighted and the best fragments are extracted on the search results page. All of this is accomplished by utilising the class that comes with Lucene as a highlighter. [6] The highlighter class marks up highlighted terms found in the most valuable sections of a document using customizable Fragmenter, Scorer, Formatter, Encoder, and Tokenizers. A brief excerpt from the text in the search string containing document makes up the search results. By utilising the getBestFragment method, the greatest number of 4 instances of the hunt phrase are found in the extracted fragment. The method getBestFragment retrieves by far pertinent bits, and the paper content is then broken down and analysed to track hits throughout the document. Formatting for results: The search queries in have results available. There are two ways that the results of the searches can be displayed, making it very user-friendly.



localhost:8085/CounterWebApp/

Search Results

5 file found for the search for text rick

08/19/2017

04:11:22

People

File Path: inputFiles\sample-richtext.rtf
Score: 0.9881595373153687
Highlighted Text is your Search:
This is a rich text document. **Rick** Crawford wrote this...

File Path: inputFiles\sample-word.docx
Score: 0.9781352877616882
Highlighted Text is your Search:
This is a sample word document written by **Rick** Crawford. This should be indexed by Lucene.

File Path: inputFiles\sample-html.html
Score: 0.9602037072181702
Highlighted Text is your Search:
this is the body of the document. Written by **Rick** Crawford

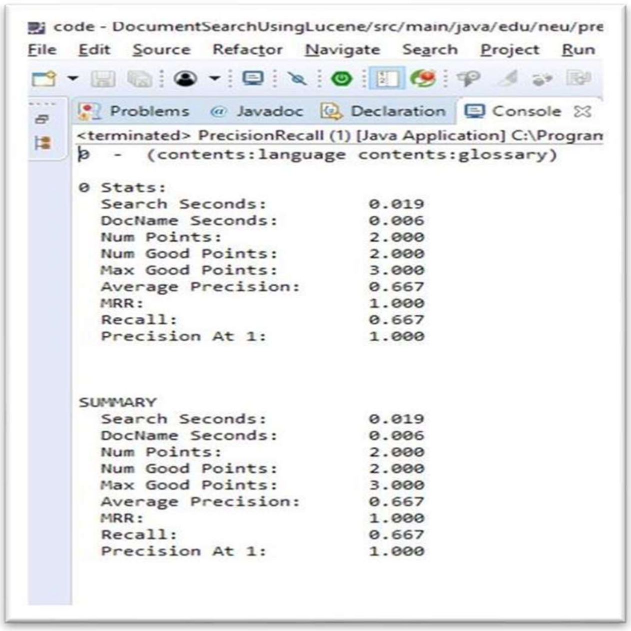
File Path: inputFiles\sample-workbook.xlsx
Score: 0.9602037072181702
Highlighted Text is your Search:
Sheet1 Column A Column B 10 20 100 200 1000 2000 This is a random row out here **Rick** Crawford

File Path: inputFiles\sample-ppt.pptx
Score: 0.8895383477210999
Highlighted Text is your Search:
www.beerhistory.com Beer has been around for thousands of years 55BC Romans brought beer to Europe 1553 Beck?s Brewery founded George Washington & Thomas Jefferson were brewers Prohibition ended for beer in 1933 ? but home brewing was illegal until 1978 **Rick** Crawford wrote this?

Figure 6 (Search Results)

SCORE AND RESUME:

I prefered Trec evaluation for precision, recall calculation. Trec evaluation inputs Relevance judgments and generate result file as an argument to calculate the precision, recall.



code - DocumentSearchUsingLucene/src/main/java/edu/neu/pre

```
<terminated> PrecisionRecall (1) [Java Application] C:\Progran
0  -  (contents:language contents:glossary)

0 Stats:
  Search Seconds:          0.019
  DocName Seconds:         0.006
  Num Points:              2.000
  Num Good Points:         2.000
  Max Good Points:         3.000
  Average Precision:       0.667
  MRR:                     1.000
  Recall:                  0.667
  Precision At 1:          1.000



SUMMARY
  Search Seconds:          0.019
  DocName Seconds:         0.006
  Num Points:              2.000
  Num Good Points:         2.000
  Max Good Points:         3.000
  Average Precision:       0.667
  MRR:                     1.000
  Recall:                  0.667
  Precision At 1:          1.000
```

Figure 7 (Statistics of Result)

CONCLUSION

This research paper describes the different methods used by search engines and what has already been done in the field of search engines. This research also describes the use of various efficient parsers and their use in search engine development. This white paper also describes the Apache Lucene API, which provides the use of indexers and search APIs that can be used to index downloaded pages and perform text searches on indexed documents, essential for search engine development. Although implementing Lucene was simple in my experiments, merging the Lucene indexing and the database queries proved to be difficult. My implementation has demonstrated a framework for combining database query techniques with the full-scale text indexing engine Lucene for a generic approach to keyword search. It comes out to be very competitive in the field of searching and indexing rather than the traditional method because of the following characteristics:

1. High-Performance Indexing, Scalable.
2. Powerful, Efficient Search Algorithms and Accurate.
3. Cross-Platform Solution and high availability.
4. Wide set of functionalities.

REFERENCES

[1] M. Chau and C. H. Wong. Designing the user interface and functions of a search engine development tool. Decision Support Systems, 48(2):369 – 382, 2010.

[2] A. Paepcke. Digital libraries: Searching is not enough–what we learned on-site. D-Lib Magazine, 2(5), May 1996. http://www.dlib.org/dlib/may96/stanford/05paepcke. html.

[3] The Apache Software Foundation. 2012. Accessed 1/11/2022.

[4] The Lemur Project. CIIR. Accessed 7/2/2012. http://www.lemurproject.org

[5] Y. C. Li, "Research and Application of Full-Text Search Engine Based on Lucene,".

[6] C. J. van Rijsbergen. Information Retrieval, 2nd edition. 1979, Butterworths.