Search Engine using Apache Lucene

Mamatha Balipa
Department of MCA
NMAM.I.T
Nitte

Balasubramani R., PhD
Professor and Head
Department of IS and E
NMAM.I.T, Nitte

ABSTRACT

The World-Wide Web is a huge network of billions of workstations and this network contains billions of web pages containing information on a wide variety of topics. There are a lot of topics discussed by people, opinions and suggestions shared on various social networking sites that the users are interested in. Low precision and low recall still exists in the current search engines. So a search engine that is effective and one that applies Web mining technology has become very important. A discussion on the various technologies used to implement a search engine and its techniques like indexing and searching on the world wide web is done in this paper. The authors propose to describe the method to create a search engine by using JSoup and Apache Lucene API in the paper.

General Terms

Search engine, web mining, text mining.

Keywords

Web, crawler, searching, indexing, JSoup, Apache Lucene.

1. INTRODUCTION

The World Wide Web is ever expanding. There are a lot of forums and social media sites like Twitter, e-commerce sites like Amazon, health related discussion forums where a lot of opinions and knowledge is shared between users.

Users or the public have increasingly searched for and used the information available on online health forums. According to a 2013 Pew Research Center survey, 72\% of the U.S. adult internet users have searched for health information online (V. V. Vydiswaran, Q. Mei, D. A. Hanauer, and K. Zheng, , 2014).

The number of social networking sites and healthcare forums have increased and patients voluntarily share their health information as well as the treatments they have undergone and the solutions they have found for their problems as well as the drugs they have used.

Medications.com, SteadyHealth.com, MedHelp.org and HealthBoards.com are examples of such online forums (H. Sampathkumar, X.-w. Chen, and B. Luo, 2014). This information is also used by researchers who help the health authorities in their post-marketing drug surveillance as well as to predict the outbreak of diseases.

Existing search engines like Google and Yahoo do a good job of searching and indexing information available on the web. But finding information is still difficult.

So to search and mine the information available on the web, the development of effective search engines become important.

Most search engines search based on the keywords. Information on the web is usually unstructured. So approaches like data mining, web mining, text mining and Natural Language processing are increasingly being used in Search engines and Web mining has become a topic research.

2. PREVIOUS WORK ON SEARCH ENGINES

2.1 Google Search Engine

Google search engine instantiates many crawlers that visit links that appear in web pages and downloads the pages represented by those links. Each page is given a unique docId. The pages are compressed and stored. An indexer parses the contents of the pages and converts them into word occurrences called hits, sorts them, distributes them and creates forward indexes. It creates an anchor file where the information about the links are stored (S. Brin and L. Page,, 2012).

2.2 Page Rank

Google search engine analyses the links in the web pages available on the internet to rank the pages which helps in providing more efficient results. This ranking is called page ranking (S. Brin and L. Page,, 2012).

2.3 Anchor Text

Anchors provide information about web pages they point to. Anchor can be used to retrieve pages that have not been crawled. These pages may contain some images or other links (S. Brin and L. Page,, 2012).

2.4 Google's Data Structures

The data structures used by Google search engine are designed in such a way that large amount of documents can be quickly and efficiently indexed and searched. But the disk seek still takes 10 ms (S. Brin and L. Page,, 2012).

2.5 Big Files

Google's Big files are virtual files distributed across many file systems. Big files are addressed using 64 bit integers (S. Brin and L. Page,, 2012).

2.6 Repository

Google's repository consists of downloaded and compressed web pages (S. Brin and L. Page,, 2012).

2.7 Document Index

Each downloaded document is given a unique docid and is stored with details of the document like, status, pointer to the file in the repository and other details (S. Brin and L. Page,, 2012).

2.8 Some of the other search engines

Alcides Calsavora and Schmidt describe a semantic search engine that can be used by users to get information about sellers and service providers and their products that can be hierarchically organized.

Sara Cohen Mamon et al developed a semantic search engine that uses XML (S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv, "Xsearch: 2003). The search engine retrieves parts of the document related to the users' query. The information retrieved is ranked using extended information retrieval methods and are presented in the order of their ranking.

Bhagwat and Polyzotis presented a file system search engine – Eureka (D. Bhagwat and N. Polyzotis, 2003). It is a semantic based search engine. It creates links between files and a file ranking system to order the files according to their importance.

Wang et al, presented a semantic search technique to get information from regular tables (H.-L. Wang, S.-H. Wu, I. Wang, C.-L. Sung, W.-L. Hsu, and W.-K. Shih, 2000). The technique recognizes the relationship between table cells and stores the data in the database. It uses query language to get information from the database.

Kandogan et al, presented a semantic search engine Avatar, that uses text search along with ontology annotations.

Maedch et al, developed an ontology search engine that uses a unified method for ontology searching A. M¨adche, B. Motik, L. Stojanovic, R. Studer, and R. Volz, 2003. They use an ontology registry to store the ontology metadata and a server to store the ontologies.

George Gardarin et al presented SEWISE that maps text data present in the web pages and creates an XML structure. It also makes the hidden semantic in the text available to program.

2.9 Keyword and sentence extraction

Machine learning methods use keywords that are extracted to create a model which is then used to retrieve keywords from new pages.

Kea is an algorithm to extract keywords using Naïve Bayes classifier. It computes the normalized the term frequency(TF), the inverted document frequency(IDF), and the position of the word in the document. The extracted words from every page are used to create a Naïve Bayes classifier. The resultant classifier is used to analyse if the given phrase is a keyword or not (H. Kian and M. Zahedi, 2011).

An ontology can contain vocabulary for representing terms belonging to a subject area. It also contains logical statements that describe the relationships between the terms.

A content repository, thesaurus and ontology repository are used to extract the content of the documents gathered by the crawler. The pages are then indexed according to their context (P. Gupta and D. A. Sharma, 2010).

2.10 Meta search engine

Meta search engine combines the results returned by multiple search engines and presented if required in a sorted order (P. Gupta and D. A. Sharma, 2010).

3. SEARCH ENGINE DEVELOPMENT

3.1 Web Document Parser using JSoup API

JSoup is an HTML parser. The various elements of an HTML page can be searched and their contents retrieved using JSoup. A crawler for crawling the web and downloading web pages has been developed using JSoup parser.

The search engine developed parses the HTML, follows all the links and downloads each page pointed to by the link. The search engine is developed using Java and threads are created to visit each link in different pages and download the links. The memory is efficiently managed by using thread pools (P. Houston, 2013).

3.2 Indexing and searching using Apache Lucene

3.2.1 Apache Lucene

Apache lucene provides API to index documents as well to query the index and fetch documents that match the query.

Apache Lucene is a open source library developed by Apache Software Foundation. It is usually used to implement text based search engines. It has API to efficiently index text documents and search for text in them.

It can be used to search the web, databases and has been used by sites like Wikipedia, Linkdin,etc., It is Java based but can also be used in programming languages like Perl, Python and .Net. (A. Sonawane, 2009)

Lucene has efficient and precise search algorithms. It retrieves the documents queried based on their ranking. It provides different types of queries like PhraseQuery, WildcardQuery, RangeQuery, FuzzyQuery, BooleanQuery and more.

In the search engine devneloped, an indexer has been built to index the web pages downloaded, by using Apache Lucene API (A. Sonawane, 2009).

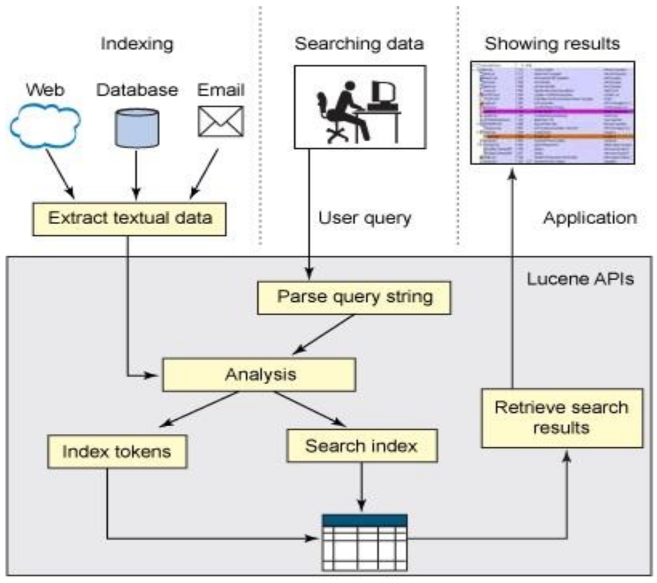


Fig 1: Building application using Lucene

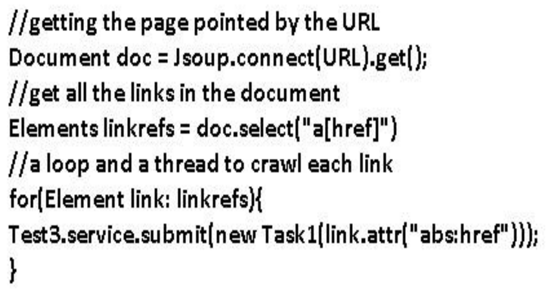
3.2.2 The indexing process

Indexing is the method of storing text data in index files, in a format that helps in fast and efficient text searching. Lucene stores the text data in a data structure called inverted index which can be stored in the file system or memory. The text data is analysed prior to storing it in the index files (A. Sonawane, 2009).

3.2.3 Environment Used

Windows 7 operating system was the platform used for the work. A machine with 4 GB RAM and a intel core duo microprocessor with a speed of 2.93GHz HGz was used for running the search engine. With a download speed of 1 Mbps, the search engine took 3 hours was taken to download 5000 pages. Java programming language was used to implement the search engine. The out of memory error on running multiple threads was handled by creating thread pools.

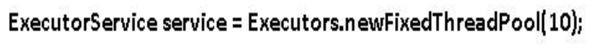
3.2.4 Code snippet using JSoup API to visit all links in a Page



```
//getting the page pointed by the URL
Document doc = Jsoup.connect(URL).get();
//get all the links in the document
Elements linkrefs = doc.select("a[href]")
//a loop and a thread to crawl each link
for(Element link: linkrefs){
Test3.service.submit(new Task1(link.attr("abs:href")));
}
```

Fig 2: Code using JSoup API to crawl links in a Page

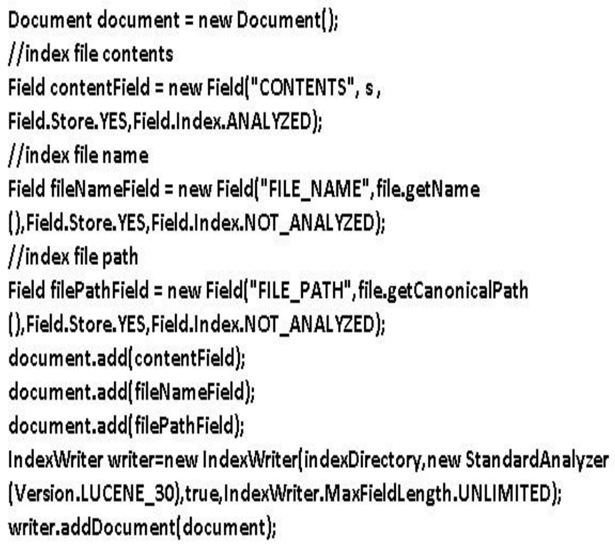
3.2.5 Code Snippet for creating Thread Pool



```
ExecutorService service = Executors.newFixedThreadPool(10);
```

Fig 3: Code to create threadpool

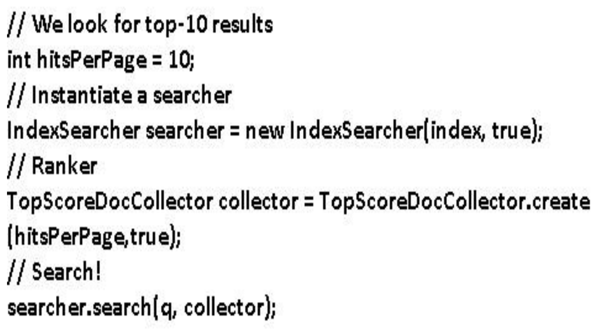
3.2.6 Code Snippet for Indexing using Lucene API



```
Document document = new Document();
//index file contents
Field contentField = new Field("CONTENTS", s ,
Field.Store.YES,Field.Index.ANALYZED);
//index file name
Field fileNameField = new Field("FILE_NAME",file.getName
(),Field.Store.YES,Field.Index.NOT_ANALYZED);
//index file path
Field filePathField = new Field("FILE_PATH",file.getCanonicalPath
(),Field.Store.YES,Field.Index.NOT_ANALYZED);
document.add(contentField);
document.add(fileNameField);
document.add(filePathField);
IndexWriter writer=new IndexWriter(indexDirectory,new StandardAnalyzer
(Version.LUCENE_30),true,IndexWriter.MaxFieldLength.UNLIMITED);
writer.addDocument(document);
```

Fig 4: Code for Indexing using Lucene

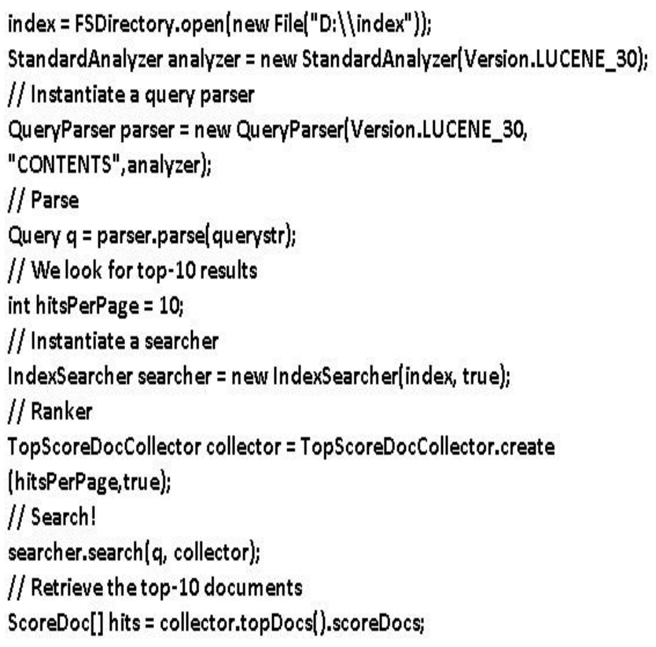
3.2.7 Code Snippet for calculating ranking/scores



```
// We look for top-10 results
int hitsPerPage = 10;
// Instantiate a searcher
IndexSearcher searcher = new IndexSearcher(index, true);
// Ranker
TopScoreDocCollector collector = TopScoreDocCollector.create
(hitsPerPage,true);
// Search!
searcher.search(q, collector);
```

Fig 5: Code for calculating rank/scores

3.2.8 Code Snippet for Searching



```
index = FSDirectory.open(new File("D:\\index"));
StandardAnalyzer analyzer = new StandardAnalyzer(Version.LUCENE_30);
// Instantiate a query parser
QueryParser parser = new QueryParser(Version.LUCENE_30,
"CONTENTS",analyzer);
// Parse
Query q = parser.parse(querystr);
// We look for top-10 results
int hitsPerPage = 10;
// Instantiate a searcher
IndexSearcher searcher = new IndexSearcher(index, true);
// Ranker
TopScoreDocCollector collector = TopScoreDocCollector.create
(hitsPerPage,true);
// Search!
searcher.search(q, collector);
// Retrieve the top-10 documents
ScoreDoc[] hits = collector.topDocs().scoreDocs;
```

Fig 6: Code for searching text

4. CONCLUSION

In this paper, the various techniques used in search engines and work that has already been done in the area of search engines are discussed. The paper also describes the use of JSoup parser and its use in developing a search engine. The paper also discusses the Apache Lucene API that provides the use of its indexer and searching API that can be used to index the downloaded pages and perform text based search in the indexed documents which is vital to the development of a search engine. In their future work the authors propose to use Natural Language processing to mine information available in the web pages and optimize the search engine.

5. REFERENCES

[1] V. V. Vydiswaran, Q. Mei, D. A. Hanauer, and K. Zheng, , 2014, “Mining consumer health vocabulary from community-generated text,” in AMIA Annual Symposium Proceedings, vol. 2014, p. 1150, American Medical Informatics Association. .

[2] H. Sampathkumar, X.-w. Chen, and B. Luo, 2014. “Mining adverse drug reactions from online healthcare forums using hidden markov model,” BMC medical informatics and decision making, vol. 14, no. 1, p. 91.

[3] S. Brin and L. Page,, 2012, “Reprint of: The anatomy of a large-scale hypertextual web search engine,” Computer networks, vol. 56, no. 18, pp. 3825–3833.

[4] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv, “Xsearch: 2003, A semantic search engine for xml,” in Proceedings of the 29th international conference on Very large data bases-Volume 29, pp. 45–56, VLDB Endowment.

[5] D. Bhagwat and N. Polyzotis, 2003 , “Searching a file system using inferred semantic links,” in Proceedings of the sixteenth ACM conference on Hypertext and hypermedia, pp. 85–87, ACM, 2005.Forman, G.

[6] H.-L. Wang, S.-H. Wu, I. Wang, C.-L. Sung, W.-L. Hsu, and W.-K. Shih, 2000, “Semantic search on internet tabular information extraction for answering queries,” in Proceedings of the ninth international conference on Information and knowledge management, pp. 243–249, ACM.

[7] A. M¨adche, B. Motik, L. Stojanovic, R. Studer, and R. Volz, 2003, “An infrastructure for searching, reusing and evolving distributed ontologies,” in Proceedings of the 12th international conference on World Wide Web, pp. 439–448, ACM.

[8] H. Kian and M. Zahedi, 2011, “An efficient approach for keyword selection; improving accessibility of web”.

[9] P. Gupta and D. A. Sharma, 2010, “Context based indexing in search engines using ontology,” International Journal of Computer Applications (0975–8887), vol. 1, no. 14.

[10] P. Houston, 2013, Instant jsoup How-to. Packt Publishing Ltd,.

[11] A. Sonawane, 2009, “Using apache lucene to search text,” Online At http://www. ibm. com/developerworks/opensource/library/os-apachelucenesearch/(as of 11 December 2013).