I. J. Computer Network and Information Security, 2013, 3, 1-12
Published Online March 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2013.03.01

A Full-text Website Search Engine Powered by Lucene and The Depth First Search Algorithm

Modinat. A. Mabayoje (MSAN, MNCS) and O. S. Oni
Department of Computer Science, University of Ilorin, P.M.B 1515, Ilorin, Nigeria
mmabayoje@yahoo.com

Olawale S. Adebayo (MCPN, MNCS)
Cyber Security Science Department, Federal University of Technology PMB 65, Minna, Nigeria
waleadebayo@futminna.edu.ng, olawalebayo@yahoo.com

Abstract — With the amount of available text data on the web growing rapidly, the need for users to search such information is dramatically increasing. Full text search engines and relational databases each have unique strengths as development tools but also have overlapping capabilities. Both can provide for storage and update of data and both support search of the data. Full text systems are better for quickly searching high volumes of unstructured text for the presence of any word or combination of words. They provide rich text search capabilities and sophisticated relevancy ranking tools for ordering results based on how well they match a potentially fuzzy search request. Relational databases, on the other hand, excel at storing and manipulating structured data -- records of fields of specific types (text, integer, currency, etc.). They can do so with little or no redundancy. They support flexible search of multiple record types for specific values of fields, as well strong tools for quickly and securely updating individual records. The web being a collection of largely unstructured document which is ever growing in size, the appeal of using RDBMS for searching this collection of documents has become very costly.

This paper describes the architecture, design and implementation of a prototype website search engine powered by Lucene to search through any website. This approach involves the development of a small scale web crawler to gather information from the desired website. The gathered information are then converted to a Lucene document and stored in the index. The time taken to search the index is very short when compared with how long it takes for a relational database to process a query.

Index Terms —Full Text search engine, Relational Database, Information Retrieval, Lucene, Depth first search algorithm

I. INTRODUCTION

Many applications that handle information on the internet would be completely inadequate without the support of information retrieval technology. How would we find information on the World Wide Web if there were no web search engines? How would we manage our email without spam filtering? Information retrieval (IR) is the area of study in information system concerned with searching for documents, for information within documents, and for metadata about documents, as well as that of searching structured or unstructured storage, relational databases, and the World Wide Web [1].

The system assists users in locating the information they need. It does not explicitly return information or answer questions. Instead, it informs on the existence and location of documents that might contain the desired information. Some suggested documents will, hopefully, satisfy the user's information need. These documents are called relevant documents. A perfect retrieval system would retrieve only the relevant documents and no irrelevant documents. However, perfect retrieval systems do not exist and will not exist, because search statements are necessarily incomplete and relevance depends on the subjective opinion of the user. In practice, two users may pose the same query to an information retrieval system and judge the relevance of the retrieved documents differently: Some users would like the results, while others will not [2].

There are three basic processes an information retrieval system has to support: the representation of the content of the documents, the representation of the user's information need, and the comparison of the two representations. Representing the documents is usually called the indexing process. The process takes place offline, that is, the end user of the information retrieval system is not directly involved. The indexing process results in a representation of the document. Users do not search just for fun; they have a need for information. The process of representing their information need is often referred to as the query formulation process. The resulting representation is the query. In a broad sense, query formulation might denote the complete interactive dialogue between system and user, leading not only to a suitable query but possibly also to the user better understanding his/her information need. The comparison of the query against the document representations is called the matching process. The matching process usually results

in a ranked list of documents. Users will walk down this document list in search of the information they need.

II. RELATED WORKS

According to [10], the following sites offer free search capabilities for Web site developers who are willing to use them.

A. Fusionbot

It offers multiple levels of search, at the free level you get: 250 pages indexed, 1 automatic index per month, 1 manual index per month, basic reporting, sitemap, and more. It even supports searching across SSL domains.

B. Freefind

It is simple to sign up for this free service. It has additional features of a site map, and "what's new" pages that are automatically generated along with your search field. You control how often they spider your site, so you can be sure that new pages are added to the index. It also allows you to add additional sites to the spider to be included in the search.

C. Google custom search engine

The Google custom search engine allows you to search not only your own site, but also create collections to search within. This makes the search more interesting for your readers because you can specify multiple sites to include in the search results. You can also invite your community to contribute sites to the search engine.

The drawbacks to this method are that you are limited to the features that the search company provides. Also, they can only catalogue pages that are live on the Internet (Intranet and Extranet sites cannot be catalogued). Finally, they only catalogue a site periodically, so you don't have any guarantee that your newest pages will be added to the search database immediately.

III. INFORMATION RETRIEVAL MODELS

Much of the development of information retrieval technology, such as web search engines and spam filters, requires a combination of experimentation and theory. Experimentation and rigorous empirical testing are needed to keep up with increasing volumes of web pages and emails. Furthermore, experimentation and constant adaptation of technology is needed in practice to counteract the effects of people that deliberately try to manipulate the technology, such as email spammers. However, if experimentation is not guided by theory, engineering becomes trial and error. New problems and challenges for information retrieval come up constantly.

A. The boolean model

The Boolean model is the first model of information retrieval and probably also the most criticised model. The model can be explained by thinking of a query term as an unambiguous definition of a set of documents. Using the operators of George Boole's mathematical logic, query terms and their corresponding sets of documents can be combined to form new sets of documents. Boole defined three basic operators, the logical product called AND, the logical sum called OR and the logical difference called NOT.

B. The vector space model

Gerard Salton and his colleagues suggested a model based on Luhn's similarity criterion that has a stronger theoretical motivation. [6]

They considered the index representations and the query as vectors embedded in a high dimensional Euclidean space, where each term is assigned a separate dimension. The similarity measure is usually the cosine of the angle that separates the two vectors $\vec{d}$ and$\vec{q}$. The cosine of an angle is 0 if the vectors are orthogonal in the multidimensional space and 1 if the angle is 0 degrees.

Mathematically According to [7], Documents and queries are both vectors

$$\overrightarrow{d_i} = \left(\mathrm{w}_{\mathrm{i},1}, \mathrm{w}_{\mathrm{i},2}, \ldots \mathrm{w}_{\mathrm{i,t}-1}, \mathrm{w}_{\mathrm{i,t}}\right)$$

Where each $\mathrm{w}_{\mathrm{i},1}$ is a weight for term j in document i ,Similarity of a document vector to a query vector = cosine of the angle between them

The cosine similarity measure formula is given by:

$$sim(d_i, q) = \frac{\Sigma_{j=1} w_{i,j} \cdot w_{q,j}}{\sqrt{\Sigma_{j=1}(w_{i,j})^2} \cdot \sqrt{\Sigma_{j=1}(w_{q,j})^2}}$$

sim(d,q) = 1 when d = q
sim(d,q) = 0 when d and q share no terms

C. Fuzzy set theory model

The IR models discussed so far assumed that index terms are independent of each other. They all represented document as a collection of index terms and this way lost the semantics of the document. As a result the matching of the query and document is often a vague one.

In fuzzy set theory each query term qi defines a fuzzy set of documents. Each document dj in the collection has a degree of membership ($\mu_{i,j} < 1$) in this set. The term $\mu_{i,j}$ is defined as:

$$\mu_{i,j} = 1 - \prod_{k_l \in d_j} (1 - c_{i,l})$$

Where $c_{i,l}$ is the correlation of the index term i and index term l (a query term is also an index term). The correlation is calculated as the odds of the term appearing together and not appearing together; given by:

$$c_{i,l} = \frac{\#\ times\ terms\ i\ and\ l\ appear\ together}{\#\ times\ terms\ i\ and\ l\ does\ not\ appear\ together} = \frac{n_{i,l}}{n_i + n_l - n_{i,l}}$$

The equation actually calculates the algebraic sum of correlations of query term q_i with all the terms in the document. The sum is implemented as complemented of a negated algebraic product. Firstly, this formulation ensures that whenever there is one index term in the document which is strongly related to q_i (i.e. $c_{i,l} \approx 1$) then $\mu_{i,j}$ will also be ≈ 1. The degree of membership is calculated using an algebraic sum overall index terms instead of a usual max function to allow smooth transition for the values of the $\mu_{i,j}$ factor. [9]

D. Crawling

The web crawler automatically retrieves documents from the web as per some defined strategy. The crawler creates a copy of all the documents it crawls to be processed by the search engine. The crawler starts from a list of URLs (documents) called seed. The crawler visits the URLs, identifies the outgoing hyperlinks there and adds them to the list of URLs (documents) to be visited. This way the crawler traverses the web graph following hyperlinks. It saves a copy of each document it visits. According to [9], the following policies are used by crawlers:

E. Indexing

Once all the data is stored to a repository (e.g. Database, file system or internet), the fun can start. At this stage the files and data are not very searchable as the data is stored in a so called "heap". To search all of this unstructured data would be very inefficient and slow. In order to make the content more accessible the data need to be stored in a structured format called an index. Thus this is why this process is called indexing. In its simplest form an index is a sorted list of all of the words and phrases that are found in the content that has been retrieved. The words and phrases will be stored in alphabetical order along with their source and rank or popularity.

IV. LUCENE LIBRARY

Lucene is a high performance Information Retrieval (IR) library, also known as a search engine library. Lucene contains powerful APIs for creating full text indexes and implementing advanced and precise search technologies into your programs. Some people may confuse Lucene with a ready to use application like a web search/crawler, or a file search application, but Lucene is not such an application, it's a framework library. Lucene provides a framework for implementing these difficult technologies yourself. Lucene makes no discriminations on what you can index and search, which gives you a lot more power compared to other full text indexing/searching implications; you can index anything that can be represented as text. There are also ways to get Lucene to index HTML, Office documents, PDF files, and much more.

A number of products have used Lucene to build their searches; some well-known websites include Wikipedia, CNET, Monster.com, Mayo Clinic, FedEx, and many more. Lucene is currently undergoing incubation at the Apache Software Foundation. Its source code is held in a subversion repository and can be found on https://svn.apache.org/repos/asf/incubator/lucene/. If you need help downloading the source, you can use the free TortoiseSVN, or RapidSVN. The Lucene project always welcomes new contributors [16].

V. DEPTH-FIRST SEARCH ALGORITHM

Depth first search follow a path to its end to its end before starting to explore another path. Precisely, suppose that a search starts from vertex v of the graph G, then the depth first search algorithm proceeds as follows. (The vertices here represent a web page on the website).

The main steps in the algorithm are listed below:

Initialization: Mark all vertices on the graph as unvisited.

Visit v and mark it as visited.

Select a vertex, say w, yet unvisited, but adjacent to v and perform a depth first search with w taken as the starting point.

On reaching a vertex with no unvisited adjacent vertices backtrack to the most recently visited vertex w that has an unvisited adjacent vertex say u, and perform a depth first search on a sub-tree having u regarded as a start vertex. Where no vertex with yet unvisited adjacent vertex can be found, terminate the search.

In [12], this is put in a recursive procedure as follows,

Procedure DepthFirstSearch(v:vertex);
Var w: vertex;
Begin
Visit v and mark it visited;
For each vertex w adjacent to v but yet unvisited do DepthFirstSearch(w)
End;

Some constraints are added to the implementation of the algorithm such that any external link encountered during the traversal of the website is ignored so as to prevent the crawler from traversing the entire web.

The implementation of the crawler using this algorithm will be explained in the next chapter.

VI. DATABASE SEARCH ENGINE ARCHITECTURE OF MOST WEBSITES

Databases are built for searching. One of the primary benefits of a database driven approach to web search development is advanced searching. It is significantly faster to search through a thousand database records

than a thousand HTML pages. Additionally, since content is broken up into logical data fields within the database, users can search for very specific content. Advanced queries such as one that would locate, say, all of the articles in a database that have an author named "John," a title containing the words "buy" and "sell," and were published in 1997, are fast and manageable with a database approach. These types of queries would be virtually impossible to facilitate with a static site.

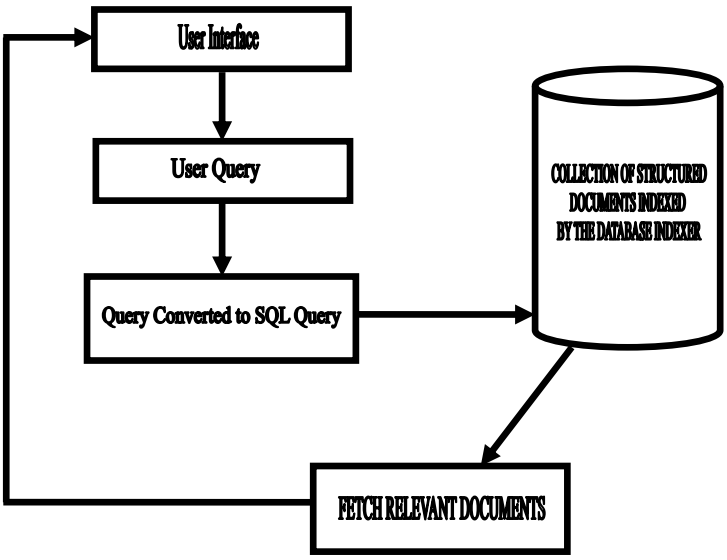

Figure 1. A database driven information retrieval on a website. [4]

VII. CHALLENGES OF THE DATABASE SEARCH MODEL

1) For every user query, the query has to be reformulated to SQL query; the entire tables of the database has to be searched in other to get all relevant documents; this increases the time it takes for the result to be returned to the user.
2) Another challenge with the database approach is that it takes into consideration only the information in the database. From research, it is discover that most contents on websites are in the form of text which are literally stored on web pages and not stored in the database. Therefore, any search by the user will not search through the pages on the site but only through the database.
3) In case of big amounts of data, SQL makes an inner join between result set returned by Full-Text search and the rest of the query which might be slow if database is running on the low powered machine (2GB ram for 20 GB of data). Switching the same query to Lucene will improve speed considerably, i.e. as the size of the database grows, more memory is needed.
4) Relational Databases had shortcomings in handling unstructured data. They are designed to provide search results that satisfy the user information need 100% because queries are built on structural field constraints, with the increase in unstructured text, developments of information retrieval systems have been gaining momentum. The aim of this IR project is to perform fast full text search specifically on free form text data.
5) Lacks a ranking mechanism for the results; when searching over unstructured data, ranking mechanism is very important. Most users examine top 10 or 20 results and ignore the rest; therefore results must be sorted by relevance in order to satisfy user's information need. The relevancy ranking of results for unstructured text search for most relational databases is not on par with that of the best full text search systems.

VIII. PROPOSED DESIGN

Based on the shortcomings of the database system analysed above, the design for the proposed system is presented here. The proposed system makes use of a crawler to gather information from every document on the website and store this information in the index. The index is a structured system of storing the unstructured data returned by the crawler. The sections below contain the design architecture of the proposed system and explained how each part of the architecture is designed.

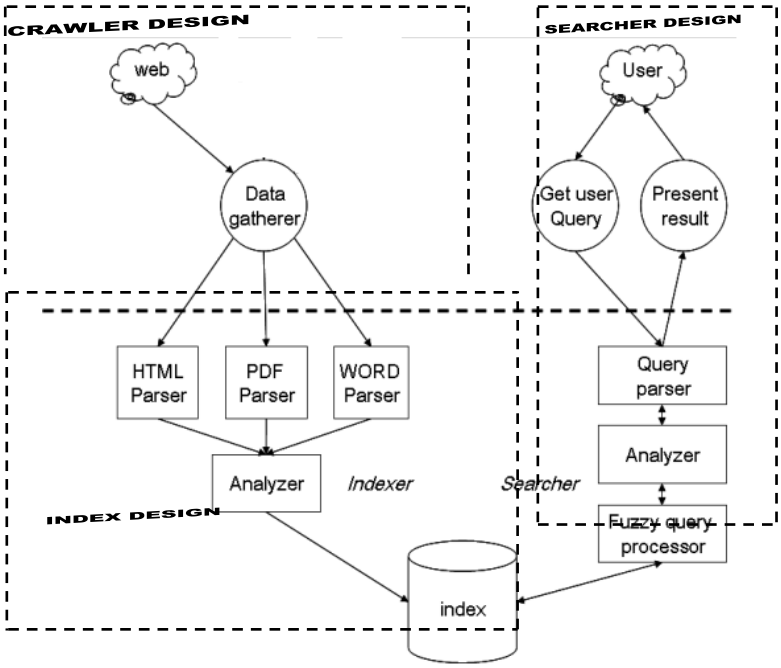

Figure 2. Architecture for the proposed design

A. Crawler design

Figure 3.3 below shows the flow of the basic sequential crawler. The crawler maintains a list of unvisited URLs called the *frontier*. The list is initialized with seed URLs which may be provided by a user or another program. Each *crawling loop* involves picking the next URL to crawl from the frontier, fetching the page corresponding to the URL through HTTP, parsing the retrieved page to extract the URLs and application specific information, and finally adding the unvisited URLs to the frontier. The crawling process may be terminated when a certain number of pages have been crawled. If the crawler is ready to crawl another page and the frontier is empty, the situation signals a dead-end for the crawler. The crawler has no new page to fetch and hence it stops.

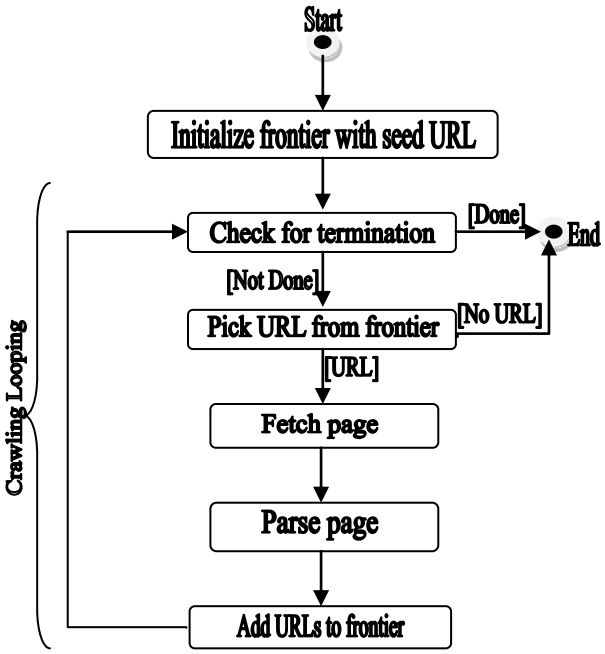

Figure 3. Architecture design of a Crawler

Crawling can be viewed as a graph search problem. The Web is seen as a large graph with pages at its nodes and hyperlinks as its edges. A crawler starts at a few of the nodes (seeds) and then follows the edges to reach other nodes. The process of fetching a page and extracting the links within it is analogous to expanding

a node in graph search. The relation among web pages on internet can be defined as a directed graph G= (W, V), W is a set of web pages in internet, V is a set of url_{ij} that the linkage exists between web $page_i$ and web $page_i$.

An information retriever is necessary to search web pages in internet. The implement method of information retriever depends mainly on the linkage structure among web pages. The linkage structure based on two Assumptions:

1) *Assumption 1: A hyperlink from web page A to web page B is a recommendation of page B by the author of page A.*
2) *Assumption 2: If web page A and web page B are connected by a hyperlink, then they might be on the same topic.*

Since the internet is a graph and crawling is carried out by traversal of the graph. Therefore, the Depth First Algorithm was used for the design of the crawler.

B. Indexer design

The crawling and the indexing process are carried out together i.e. as the crawler fetches the document, it is analysed and indexed. The heart of a search engine resides in the *index*. An index is highly efficient cross-reference lookup data structure. In most search engines, a variation of the well-known *inverted index* structure is used. An inverted index is an inside-out arrangement of documents such that terms take centre stage. Each term refers to a set of documents.

Index construction algorithm [2]

```
Algorithm CreateIndex(collection, stemmer) {
    For Each Document doc in collection {
        doc_entry = index.addDocEntry(doc.id);
        For Each Token tok in doc.fullText {
            If tok is compound word tok_ont in onto {
                tok = stemmer.stem(tok_ont);
            } Else {
                tok = stemmer.stem(tok);
            }
            doc_entry.countOccurence(doc, tok);
            If doc_entry is not in index.termEntries {
                Index.addTermEntry(tok);
            }
            term_entry = index.getTermEntry(tok);
            term_entry.countOccurence(doc, tok);
        }
    }
    Index.computeWeightTerms();
    Returns index;
}
```

The indexing process begins with collecting the available set of documents by the data gatherer (crawler).

The parser converts them to a stream of plain text. For each document format, a parser has to be implemented.

In the analysis phase, the stream of data is tokenized according to predefined delimiters and a number of operations are performed on the tokens. Each of the tokenized word is added to the index. The search process begins with parsing the user query. The tokens and the Boolean operators are extracted. The tokens have to be analysed by the same analyser used for indexing. Then, the index is traversed for possible matches in order to return an ordered collection of hits. The fuzzy query processor is responsible for defining the match criteria during the traversal and the score of the hit.

Complete index creation operation occurs usually once. The whole set of documents is parsed and analyzed in order to create the index from scratch. This operation can take several hours to complete.

The operation of updating index is called *incremental indexing*. It is not supported by all search engines. Typically, a worker thread of the application monitors the actual inventory of documents. In case of document insertion, update, or deletion, the index is changed on the spot and its content is immediately made searchable. Lucene supports this operation.

Lucene divides its index into several *segments*. The data in each segment is spread across several files. Each index file carries a certain type of information.

The exact number of files that constitute a Lucene index and the exact number of segments vary from one index to another and depend on the number of fields the index contains. The internal structure of the index file is public and is platform independent.

IX. IMPLEMENTATION PHASE

A. Software requirement

The following software should be installed on the computer to be able to implement the system design earlier specified.

1) Windows vista OS or windows 7 must be installed on the system
2) Java development kit (JDK 1.5 or later version)
3) Netbeans Integrated development environment with glassfish version 3
4) Lucene and HTML parser library

B. Crawler and Index Implementation

The frontier is implemented as an array of type URL in which case the depth-first crawler can be used to blindly crawl the Web. Iteration was done over the array to get the URL to crawl next and the new URLs are added to the tail of the queue. Due to the limited size of the frontier, precaution must be taken to make sure that no duplicate URLs exist in the frontier. A

linear search to find out if a newly extracted URL is already in the frontier is costly.

In order to fetch a Web page, an HTTP client which sends an HTTP request for a page and reads the response is needed. The client needs to have timeouts to make sure that an unnecessary amount of time is not spent on slow servers or in reading large pages. Modern programming languages such as Java and Perl provide very simple and often multiple programmatic interfaces for fetching pages from the Web. However, one must be careful in using high level interfaces where it may be harder to find lower level problems. For example, with Java one may want to use the java.net.Socket class to send HTTP requests instead of using the more ready-made java.net.HttpURLConnection class.

Once a page has been fetched, its content is parsed in order to extract information that will feed and possibly guide the future path of the crawler. Parsing may imply simple hyperlink/URL extraction or it may involve the more complex process of tidying up the HTML content in order to analyse the HTML tag tree. Parsing might also involve steps to convert the extracted URL to a canonical form, remove stop words from the page's content and stem the remaining words. In this part, the concentration is on crawling a website and then adding the crawled site to a lucene index. In chapter three, the notion of recursively crawling a web page to eventually find all of the pages/links in a web site using the depth first search algorithm was discussed. This can be achieved by implementing a recursive indexing/crawling function. The recursive crawler/indexer makes use of an object called *Link Parser;* this is an object that makes use of the HTML parser library to extract all of the links form a particular web page. The link extracted is then parsed into a function that fetches the page and then converts it into a lucene document because only lucene documents can be stored in the index.

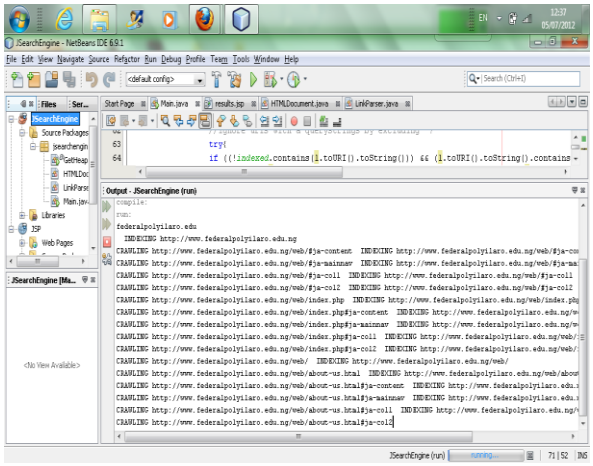

Figure 4. Process page

In order to write a document to the index, lucene requires the use of an Analyser, an analyser processes the content before it is added to the index.

Figure 5. Search Page (index.jsp)

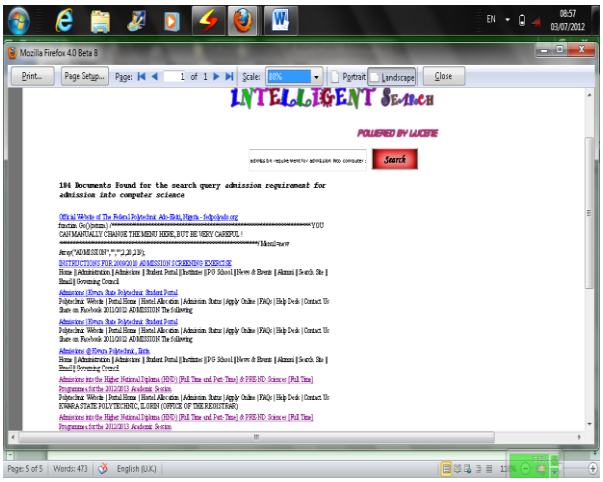

Figure 6. Result page (result.jsp)

X. SYSTEM PERFORMANCE EVALUATION

In this section, the evaluation of the effectiveness of the website search developed using lucene is carried out. Due to the absence of standard corpora with suitable characteristics, we use three locally compiled corpora. Evaluation based on the precision and recall metrics as seen in chapter two requires labour-intensive screening of the complete corpora, as well as the collaboration of several experts in the domain of the corpora. In comparison, a known-item retrieval setting reduces the amount of manual labour required and allows a semi-automatic selection of items, as described in the following sections.

A. Evaluation benchmark

1) Start with a corpus of documents.
2) Collect a set of queries for this corpus.
3) Have one or more human experts exhaustively label the relevant documents for each query.
4) Typically assumes binary relevance judgments, that is, relevant or not relevant.
5) Requires considerable human effort for large document/query corpora.

B. Corpora

The domain model for the information retrieval as in figure 3 requires that a website is present in the evaluation corpus. For the evaluation of this work, the data gatherer (crawler) in conjunction with the indexer was used to index three polytechnic websites. The reason for choosing these tertiary institutions is for us

to have access to a lot of students who can help us in determining whether a set of retrieved document is relevant to a query or not. The corpora consist of web information from Kwara State Polytechnic Website (www.kwarapolytechnic.com), FEDERAL polytechnic, Offa Website (www.fedpoffa.edu.ng), and The Federal Polytechnic, Ado-Ekiti Website (www.fedpolyado.org).

Table 1. Corpus and the number of documents

CORPUS	NUMBER OF DOCUMENTS
www.kwarapolytechnic.com	102
www.fedpoffa.edu.ng	50
www.fedpolyado.org	131

C. Methodology for choosing search queries

Choosing representative search queries and relevant documents is a central part of the known-item retrieval scenario; it is usually performed by experts in the subject matter with a reasonably complete knowledge of the documents in the corpus. Known items and search queries in a semi-automatic manner were extracted due to a limited amount of manpower available for the evaluation. Since objective criteria are used for choosing search queries, personal bias was prevented from affecting the evaluation results. Where a human judgement is necessary, two different judges choose relevant documents independent from each other. Here, the 10 queries used are the ones obtained from FAQs (Frequently Asked Questions) of most higher institutions which are the ones based on admission, registrations and requirements for a field of study.

D. 11-Point Average Precision

The 11-point average precision is a measure for representing performance with a single value. A threshold is repeatedly tuned such that allow the recall to take the values 0.0, 0.1, 0.2 ….., 0.9, 1.0. At every point the precision is calculated and at the end the average over these eleven values is returned. The retrieval system must support ranking policy. [1]

Precision: This is the ability to retrieve top-ranked documents that are mostly relevant.

Recall: This is the ability of the search to find *all* of the relevant items in the corpus.

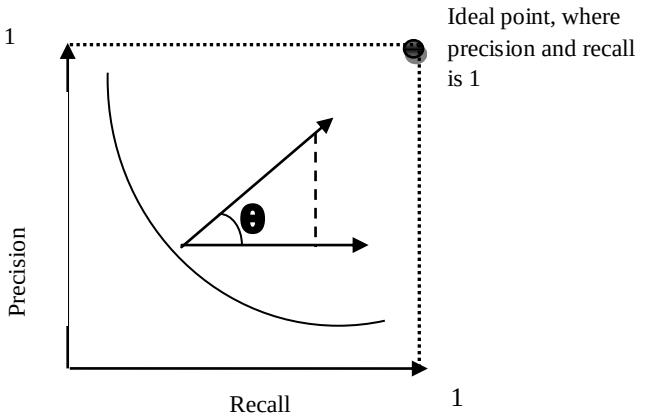

Figure 7. A Typical Recall/Precision Curve

TABLE 2. 11 Point Recall / Precision Table for Kwara State Polytechnic, Ilorin, Nigeria Search Module over 10 Queries on KWARAPOLY Website

RECALL	AVERAGE PRECISION
0.0	1
0.1	1
0.2	0.889
0.3	0.889
0.4	0.516667
0.5	0.516667
0.6	0.166667
0.7	0
0.8	0
0.9	0
1.0	0

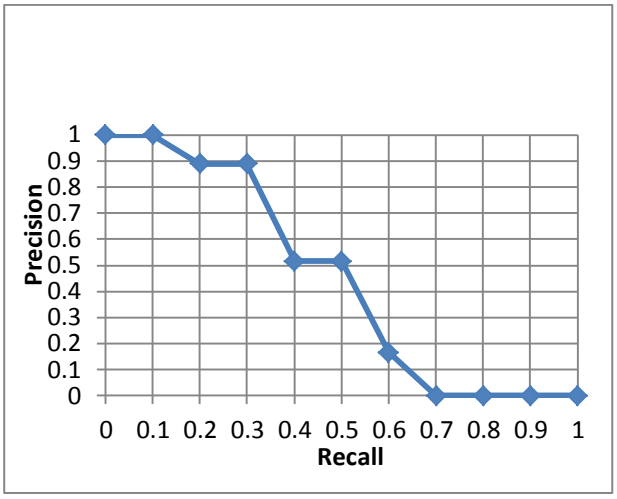

Figure 8. Average 11-point r-p curve across 10 queries on kwarapoly website using kwarapoly search module 11 point. Average precision = 0.4525

TABLE.3. 11 Point recall/precision table for website search engine using lucene on kwara polytechnic website

RECALL	AVERAGE PRECISION
0.0	1
0.1	1
0.2	1
0.3	1
0.4	0.833333
0.5	0.833333
0.6	0.857
0.7	0.486
0.8	0.486
0.9	0.285667
1.0	0.285667

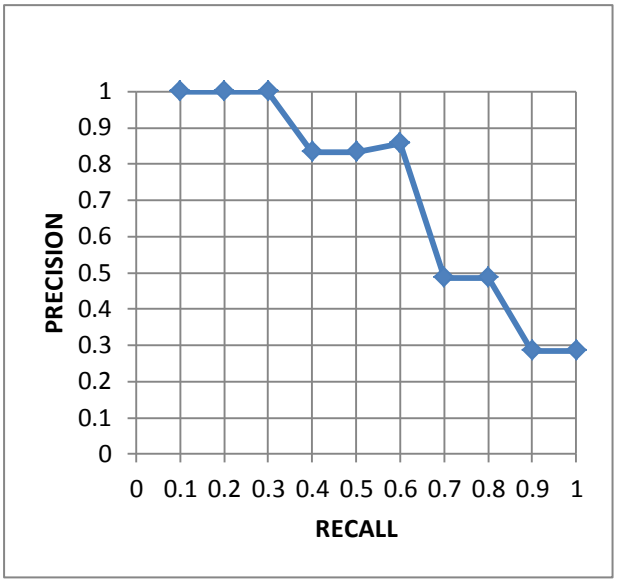

Figure 9. Average 11-point r-p curve across 10 queries on kp website using lucene search. Mean average precision= 0.7334

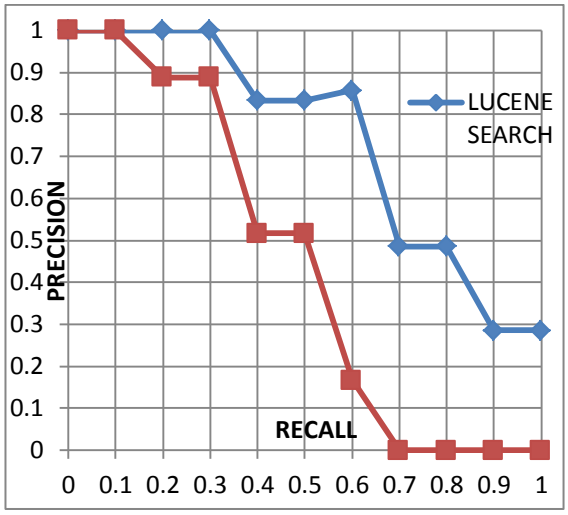

Figure 10. Comparison of 11-point average of both search engine

REMARK: The curve closest to the upper right-hand corner of the graph indicates the best performance.

TABLE 4. 11Point recall/precision table for website search engine using lucene on fedpolyado website

RECALL	AVERAGE PRECISION
0.0	0.547667
0.1	0.547667
0.2	0.583333
0.3	0.5
0.4	0.428667
0.5	0.451
0.6	0.095333
0.7	0
0.8	0
0.9	0
1.0	0

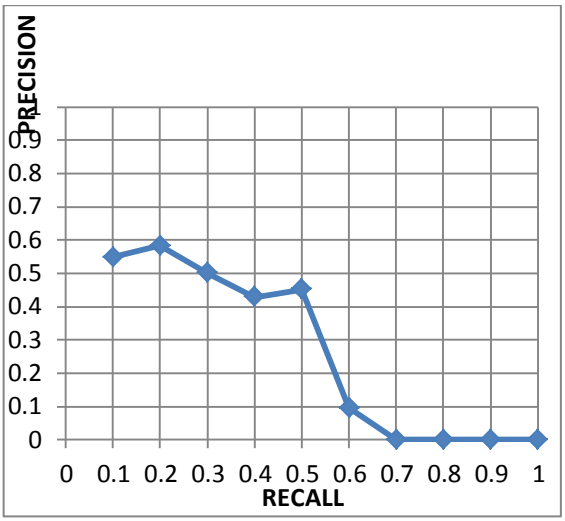

Figure 11. Average 11-point r-p curve across 10 queries on fedpoado website run from the new system Mean average precision = 0.2606

Table 5. average 11-point r-p curve across 10 queries on offa poly website using lucene search engine

RECALL	AVERAGE PRECISION
0.0	0.527667
0.1	0.527667
0.2	0.577667
0.3	0.633333
0.4	0.666667
0.5	0.576333
0.6	0.611
0.7	0.277667
0.8	0.222333
0.9	0
1.0	0

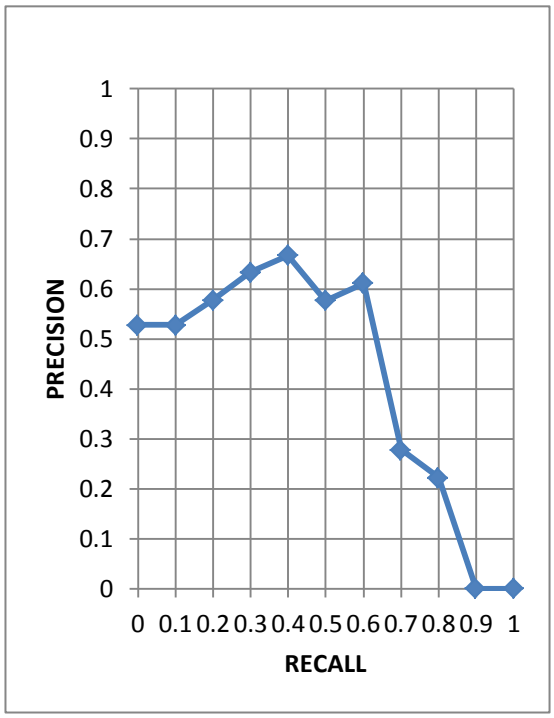

Figure 12. AVerage 11-point r-p curve across 10 queries on offa poly website from the new system Mean average precision = 0.42003

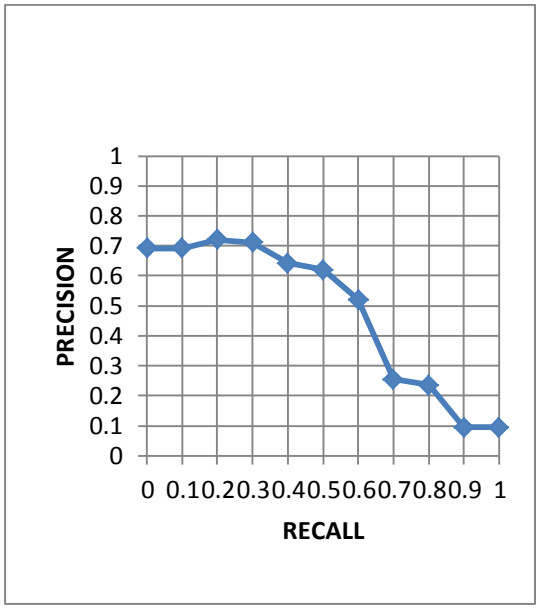

Figure 13. Overall r-p curve of the new search engine Mean average precision = 0.48003

XI. DISCUSSION

Table 2 and Figure 8 show the recall-precision table and curve for results of the 10 queries run from the Google powered search module on the kwara polytechnic website respectively. Table 3and figure 9 shows the recall-precision table and curve on the same query set on kwara polytechnic website but using the newly developed. On comparison of the two curves in figure 9, we observed that the blue curve (obtained from the newly developed system) is closer to the upper right of the curve than the red curve (obtained from Google powered search module on kwara polytechnic website) signifying its proximity to the ideal point of an information retrieval system as explained in figure 8.

Table 4 and figure 9 show the recall-precision table and curve for results of running searches using the newly designed search engine on the same query sets. It can be deduced from the graph that the height of the curve is not as close to the upper right as the curve in figure 8 and figure 9. This doesn't happen as a result of flaws in the new design but largely has to do with how well the website is structured. A similar scenario is observed from figure 12. Figure 13 shows the overall average recall-precision curve for searches run across the entire corpora. It is observed that the effect of the lowly structured websites reduces the height of the graph. This led to the conclusion that how structured well the website is affects the precision and recall of the result even though the effect might be minimal on some websites.

From the figures above, it is also observed that there is usually a trade-off between recall and precision i.e. at a high recall value, more documents containing a lot of junks is retrieved by the system and hereby reducing precision while at a high precision value, less but the

most relevant documents are retrieved and thereby providing a low recall value. Another observation is that the system seems to perform well on one query than it does to another. This has to do with the query formulation skills of the user and how much knowledge a user had about the website content.

XII. FUTURE WORKS

The field of information retrieval is a very fascinating research area where improvements can always be made no matter how sophisticated your retrieval application looks. Based on the limitations outlined above and the challenges encountered during the development and testing stages, the following areas of improvements have been identified and they are highlighted below:

1) Apache Nutch should be used to develop a better crawler: Nutch is a full-text crawler library designed by apache using Lucene technology. The crawler designed in this work as stated above has a lot of flaws and we've identified Nutch as a solution.
2) Semantic annotation should be introduced to this work using ontology model: The use of ontologies to overcome the limitations of keyword-based search has been put forward as one of the motivations of the Semantic Web since its emergence in the late 90's. Ontology is a collection of concepts and their interrelationships which can collectively provide an abstract view of an application domain. This involves the application of sense or semantics to the knowledge base and user queries. Hence, result is not just returned based on the keywords but also on the meaning of the query.
3) The keywords should be highlighted: Google and other major search engine make use of this technique to make the result page more readable. We intend to integrate this in later works.

REFERENCES

[1] Wikipedia, the Encyclopaedia: *The vector Space Model* [Online], July, 2012. Available: http://en.wikipedia.org/wiki/Information_retrieval.

[2] H. S. Al-Obaidy, *Building Ontology Web Retrieval System Using Data Mining*, Unpublished PhD thesis, Dept. of Computer Science, Ahlia University, Bahrain, 2009.

[3] D. M. Christopher, R. Prabhakar and S. Hinrich, *An Introduction to Information Retrieval, (online edition)*. Cambridge University Press, 2009.

[4] H. DJOERD, *Information Retrieval Models (Author's Version)*. Twente: University of Twente, 2005.

[5] Jarkata Lucene Javadoc: *Lucene 3.6.0 Documentation [Online], May*, 2010. Available: http://lucene.apache.org/core/3_6_0/api/all/index.html.

[6] G. Salton & M. McGill, *Introduction to Modern Information Retrieval*, London: McGraw-Hill, 1983.

[7] Lecture Note, *The Vector Space Model* [Online], May, 2012.Available: http://www.csee.umbc.edu/~ian/irF02/lectures/07Models- VSM.pdf.

[8] S. E. Robertson, C. J. Van Rijsbergen and M. F. Porter. *Probabilistic models of indexing and searching. In R. Oddy et al. (Ed.), Information Retrieval Research, (pp. 35-56)*, Butterworths, 1981.

[9] D. Joydip and B. Pushpak, *Seminar Report on Ranking in Information Retrieval. Mumbai*: Indian Institute of Technology, Bombay, 2010.

[10] K. Jennifer, *Adding Search Functionality to Your Web Site* [Online], April, 2010. Available: http://webdesign.about.com/od/administration/a/aa091399.htm.

[11] P. WILSON, *Information Storage and Retrieval*, vol. 9(8), 457-471, 1973.

[12] P. B. SHOLA (2003), *Data Structure with implementation in C and Pascal*. Feyisetan Press, Ibadan, 2003, *pp. 119-120*.

[13] F. Burkowski, *Retrieval activities in a database consisting of heterogeneous collections of structured texts, in the 15th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'92), 1992, pp. 125*.

[14] Wikipedia, the Encyclopedia: *Java Programming Language* [Online]. July, 2012. Available: http://en.wikipedia.org/wiki/Java_programming.

[15] Oracle Corporation, Java Server Pages: A developer's perspective [Online], July, 2012. Available: http://java.sun.com/developer/technicalArticles/Programming/jsp/ Java Server Pages A developer's perspective.htm.

[16] Smith. Introducing Lucene.Net. [Online], May, 2012.Available: http://www.codeproject.com/Articles/29755/Introducing-Lucene-Net

[17] Source Fourge, HTML Link Parser Documentation [Online], April, 2012. Available: http://htmlparser.sourceforge.net/HTML Parser.htm

[18] M. ZHU. Recall Precision and Average Precision, 2004.

Authors

M. A. Mabayoje is a lecturer in the department of Computer Science, University of Ilorin, Nigeria. She bagged Bachelor of Science and Master of Science in Computer Science in the University of Ilorin, Nigeria. She is currently a PhD student in the same university. She is a member of Nigeria Computer, Society, Science Association of Nigeria among others. Her

research interests include ontology, Artificial Intelligence, Software Engineering. She is married with children.

Olawale Surajudeen Adebayo (MCPN, MNCS) is a lecturer in the department of Cyber security science department, Federal University of Technology, Minna, Niger State Nigeria. He bagged B.Tech. in, Mathematics and Computer science from Federal University of Technology, Minna and MSc. in Computer science from University of Ilorin, Kwara state, Nigeria. He is presently a PhD student in the department of cyber Security science, Federal University of Technology, Minna. His current research interests include: Information security, Cryptology, Machine learning, Data mining and Computational intelligent. He is a reviewer to more than five international and local Journals.

