ISSN: 2321-7782 (Online)
Impact Factor: 6.047

Volume 4, Issue 5, May 2016

International Journal of Advance Research in Computer Science and Management Studies

Research Article / Survey Paper / Case Study

Available online at: www.ijarcsms.com

A Semantic Search Engine for Indexing and Retrieval of Relevant Text Documents

Jayalakshmi T S[1]
M.Tech in Computer Science and Engg
PES College of Engg
Mandya, Karnataka – India

C Chethana[2]
Asst. Professor, Dept.of CSE
PES College of Engg
Mandya, Karnataka – India

Abstract: ***In Existing System the Search Engine searches the file based on the name of the file or folder. And also search based on the modified time, size of the files, file format. In the proposed system search engine searches based on the contents of the files. Based on the contents in the file the engine creates the indexing and retrieves the relevant document from the computer. It consumes less time for searching the files in the PC. This design based on the Inverse Document Frequency [IDF] Text Mining Technology. The proposed system consist A Document Processor, The query processor, and Search and Matching Function modules. It handles the Synonyms and wild characters (?,+).***

Keywords: Text Mining, Search Engine, Document Processor, Query processor, IDF, Preprocessing.

I. INTRODUCTION

Search engines match queries against an index that they create. The index consists of the words in each document, plus pointers to their locations within the documents. This is called an inverted file. In this work search engine or IR system comprises three essential modules:

- A document processor
- A query processor
- A search and matching function

In Existing System the Search Engine searches the file based on the name of the file or folder. And also search based on the modified time, size of the files, file format. In the proposed system search engine searches based on the contents of the files. Based on the contents in the file the engine creates the indexing and retrieves the relevant document from the computer. It consumes less time for searching the files in the PC.

A. Full Text Search Engine

Full text searching is the process of searching for words in a block of text. There are 2 aspects to full text indexers / searchers:

Existence: means finding words that exist in the many blocks of texts stored (e.g. 'bob' is in the PDF documents stored)

Relevance: meaning the text blocks returned are delivered by a ranking system which sorts the most relevant first.

The first part is easy, the second part is difficult and there is much contention as to the ranking formula to use. This application only implements the first part in this version, as most of use and need the existence more in our applications than relevance.

B. Features:

This application has been built with the following features in mind:

- Fast operating speed.
- Incredibly small code size.
- Uses WAH compressed BitArray to store information.
- Multi-threaded implementation meaning you can query while indexing.
- Tiny size only 38kb DLL.
- Highly optimized storage, typically ~60%.
- Query strings are parsed on spaces with the AND operator (e.g. all words must exist).
- Wildcard characters are supported (*,?) in queries.
- OR operations are done by default.
- AND operations require a (+) prefix.

II. Literature Survey

Text Mining has become an important research area. Text Mining is the discovery by computer of new, previously unknown information, by automatically extracting information from different written resources. A key element is the linking together of the extracted information together to form new facts or new hypotheses to be explored further by more conventional means of experimentation. Text mining is different from what are familiar with in web search. In search, the user is typically looking for something that is already known and has been written by someone else. In text mining, the goal is to discover unknown information, something that no one yet knows and so could not have yet written down.

Search engine is the popular term for an information retrieval (IR) system. While researchers and developers take a broader view of IR systems, consumers think of them more in terms of what they want the systems to do — namely search the Web, or an intranet, or a database. Actually consumers would really prefer a finding engine, rather than a search engine.

Sergey brin et al., presents work of search engine. The engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the web, very little academic research has been done on them. in this addresses this question of how to build a practical large-scale system which can exploit the additional information present in hypertext. also we look at the problem of how to effectively deal with uncontrolled hypertext collections where anyone can publish anything they want.

K. satya sai prakash et al., proposed the architecture, design, implementation and performance evaluation of new generation search engine. He give a generic architecture for new generation search engine and discuss how a prototype search engine sarvagna1, based on that architecture is developed. we also give a set of design standards for search engines and adherence to those standards while designing is highlighted. a benchmark metric is introduced with the help of the design guidelines to rank any search engine.

III. Architecture of Proposed System

Search engines match queries against an index that they create. The index consists of the words in each document, plus pointers to their locations within the documents. This is called an inverted file.

The users specify the source and take the input query from the user then the all documents are collected from specified source. Then Builds the list of words and notes where they were found then builds the Index. This index is creating based on its own system of weighting. Encodes the data to save space. Encoding data consumes less memory than the normal data and stores the data for user to access. The query processor take the input query from the user and it process and optimized the query.

The query processor gives the output to the search and matching. The search and matching module matches the query in the index. If the search query is matched then it shows the full path of the file which include search term otherwise no search files are found.

In this work search engine or IR system comprises three essential modules:\

- A document processor
- A query processor
- A search and matching function

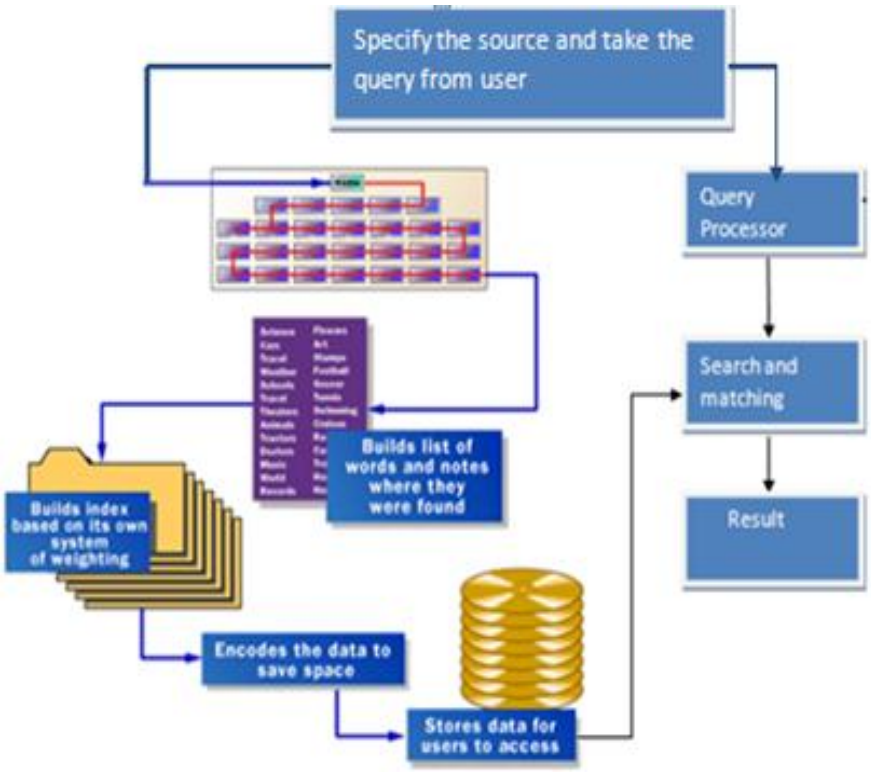

Fig 1: Architecture of Search engine/proposed system

IV. MODULES AND TECHNIQUES OF PROPOSED SYSTEM

A. *Document Processor*

The document processor prepares, processes, and inputs the documents that users search against. The document processor performs some or all of the following steps:

- Normalizes the document stream to a predefined format.
- Breaks the document stream into desired retrievable units.
- Removes special characters.
- Identifies potential indexable elements in documents.
- Deletes stop words.

- Stems terms.
- Extracts index entries.

Creates and updates the main inverted file against which the search engine searches in order to match queries to documents.

B. Query Processor

Query processing has seven possible steps, though a system can cut these steps short and proceed to match the query to the inverted file at any of a number of places during the processing. Document processing shares many steps with query processing. More steps and more documents make the process more expensive for processing in terms of computational resources and responsiveness. However, the longer the wait for results, the higher the quality of results. Thus, search system designers must choose what is most important to their users — time or quality. Publicly available search engines usually choose time over very high quality, having too many documents to search against.

The steps in query processing are as follows (with the option to stop processing and start matching indicated as "Matcher"):

- Tokenize query terms.
- Recognize query terms vs. special operators.
- Delete stop words.
- Stem words.
- Create query representation.
- Expand query terms.
- Compute weights.

C. Search And Matching Function

How systems carry out their search and matching functions differs according to which theoretical model of information retrieval underlies the system's design philosophy. Since making the distinctions between these models goes far beyond the goals of this article, we will only make some broad generalization in the following description of the search and matching function.

Searching the inverted file for documents meeting the query requirements, referred to simply as "matching" is typically a standard binary search, no matter whether the search ends after the first two, five or all seven steps of query processing. while the computational processing required for simple, unweighted, non-Boolean query matching is far simpler than when the model is an NLP-based query within a weighted, Boolean model, it also follows that the simpler the document representation, the query representation, and the matching algorithm, the less relevant the results, except for very simple queries, such as one-word, non-ambiguous queries seeking the most generally known information.

Having determined which subset of documents or pages matches the query requirements to some degree, a similarity score is computed between the query and each document/page based on the scoring algorithm used by the system. scoring algorithms ranking are based on the presence/absence of query term(s), term frequency, tf/idf, Boolean logic fulfillment, or query term weights.

D. Preprocessing Technique

How do we count words? First we need to define what a word is. This is highly non-trivial for languages without space like Chinese. Even for seemingly simple English, getting text into a form where you can count words is quite involved, as we see below. Preprocessing text is called tokenization or text normalization. Things to consider include

- Throw away unwanted stuff(e.g., HTML tags – but sometimes they are valuable, UUencoding, etc.)
- Word boundaries: white space and punctuations – but words like Ph.D., isn't, e-mail, C/net or $19.99 are problematic. If you like, we can spend a whole semester on this... This is fairly domain dependent, and people typically use manually created regular expression rules.
- Stemming (Lemmatization): This is optional. English words like 'look' can be inflected with a morphological suffix to produce 'looks, looking, looked'. They share the same stem 'look'. Often (but not always) it is beneficial to map all inflected forms into the stem. This is a complex process, since there can be many exceptional cases (e.g., department vs. depart, be vs. were). The most commonly used stemmer is the Porter Stemmer.
- Stopword removal: the most frequent words often do not carry much meaning. Examples: "the, a, of, for, in, ..." You can also create your own stop word list for your application domain.
- Capitalization, case folding: often it is convenient to lower case every character. Counterexamples include 'US' vs. 'us'. Use with care.

*Inverse Document Frequency:*Inverse Document Frequency Estimate the rarity of a term in the whole document collection. (If a term occurs in all the documents of the collection, its IDF is zero.)

V. Conclusion

This paper has presented an overview of search engine and retrieval of relevant document. Search engines match queries against an index that they create. The index consists of the words in each document, plus pointers to their locations within the documents. Then it matches the word with index and display the full path of the file which contains search keyword. This paper is implemented based on the contents of the file. It consumes less memory to store the index and it searches the keyword in less amount of time. It handles the Synonyms and wild character.

VI. Acknowledgement

I would like to express my special thanks of gratitude to my Guide C Chethana as well as our principal Dr. V Sridhar who gave me the golden opportunity to do this wonderful project on the topic **A Semantic Search Engine for Indexing and Retrieval of Relevant Text Documents**, which also helped me in doing a lot of Research and I came to know about so many new things I am really thankful to them. Secondly i would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

References

1. Vishal Gupta, "A survey of Text Mining Techniques and Applications", journal of Emerging Technologies in web Intelligence,vol.1,NO. 1,August 2009
2. Lee Underwood, "A Brief History of Search Engines";www.webreference.com/authoring/search_history/.
3. Monica Peshave," HOW SEARCH ENGINES WORK AND A WEB CRAWLER APPLICATION"
4. Grossan, B. "Search Engines: What they are, how they work, and practical suggestions for getting the most out of them," February1997.
5. Salton, G., Buckley, C., 1988. Term weighting approaches in automatic text retrieval. Information Processing and Management, 24(5):513-523.
6. M. Bishop, Pattern Recognition and Machine Learning. Springer, 2006.
7. A Brief Survey of Text Mining. (Andreas Hotho KDE Group University of Kassel Andreas N¨urnberger Information Retrieval Group School of Computer Science May 13, 2005).

