Inverted Indexing Mechanism for Search Engine

Priyanka S. Zaware
Department of Computer Engineering
JSPM's Imperial College of Engineering and Research, Wagholi, Pune
Savitribai Phule Pune University, India

Satish R. Todmal
Head of Department, Department of Computer Engineering
JSPM's Imperial College of Engineering and Research, Wagholi, Pune
Savitribai Phule Pune University, India

ABSTRACT
Now a day, the web search engine becomes a very important gateway for common people looking for useful information on the internet but due to the dynamic nature of the web, it is difficult to find the relevant documents to fulfill user requirements. For this purpose, search engine maintains the index of documents stored in the repository. When a user enters a query search engine searches the index in order to find the relevant documents to fulfill user requirements. The query topic relevance depends on the information stored in the index. The performance of search engine is depends on the powerful structure of the index. Generally, inverted index are based on the frequency of keywords present in number of documents. An improved indexing mechanism to index the web documents is being proposed to improve the efficiency of the search engine that keeps the topic id along with the document id and also inverted index for ancestors in the (term, d, t) form. The structure is implemented using Trie structure. The proposed method will efficiently store the documents and will make the search fast.

General Terms
Information retrieval

Keywords
Search engine, Indexing, Web document, Repository, Inverted index

1. INTRODUCTION
Internet is a physical network of large number of computers connected to each other, the World Wide Web, WWW is a logical collection of hyperlinked documents shared by the number of computers of this network. A hyperlinked document is a document with references to other documents. Search engines are used for finding specific information on the web. The major issue raised by the WWW is to find best suit of documents that fulfill user needs. With the technology growth, internet has become the most important source for information search and also one of the most important media of information exchange. Search engines contain different components for simplifying the task of searching. Otherwise, it is very hard to search such a large collection of web pages and get the result back in few seconds.

Generally search engine is made up of three parts: crawler, indexer and query processor. The Crawler also called the spider that parses the web collecting information and stores them into a huge repository. After collecting these documents are then processed to extract relevant information. An important subtask of Web crawling is the identification of duplicate pages in the Web. Indexing module prepares the index of the local database. Each Web document has its own ID number called document identifier, which is assigned to it when a new URL is traversed out of a web page. The indexer takes the web pages collected by the crawlers and stores them in the form of inverted index. Query processing module handles the user queries by traversing the index. The main part that is heart of search engine is an index module which increases the speed of searching the relevant information with help of an better indexing technique.

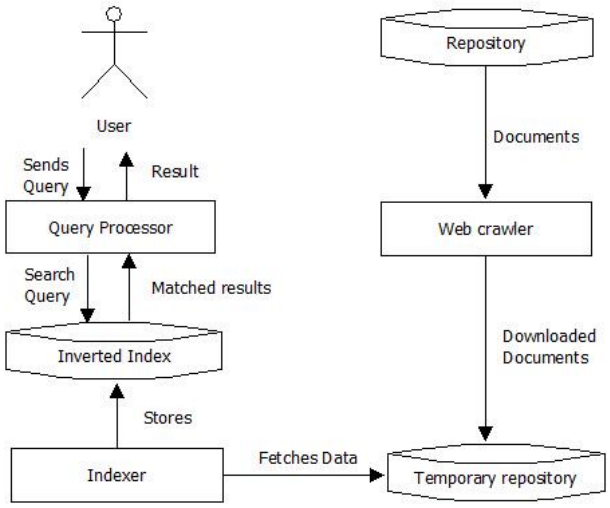

Figure 1 Search Engine Architecture

An architecture of search engine is shown in Fig 1. It has two layers: Font-end layer and Back-end layer. Front-end layer consist of a two main components: User Interface and Query Processor. User interacts with the system with the help of user interface. Then search engine takes query from the user which acts as an input to the query processor. The query processor searches the result related to the query to get the desired match results and rank them in specific order and then give it to user interface to display it to the user. Back-end layer consist of web crawler, repository and Indexer. Web crawler starts it working with list of stored web pages as an initial input. It traverses the whole repository to collect the corresponding web page and stores them into temporary repository. Indexers take the downloaded documents from temporary repository and extract all the keywords, stop words are removed and stem the root word and store it in an Inverted Index.

Indexing is the process of arranging the records in systematic order. An index enables user to search records faster. Without the index, researchers have to look through hundreds or thousands of records to locate an individual record. The existing Indexing technique stores the web pages only in the form of keywords present in them. However, ontology based

Indexing using contextual senses [7] has tried to store some related information but they have not considered the importance of the keyword. Only Keyword is not sufficient for getting information about the web page. The presence of the keyword in various HTML tags of web documents should also be considered for indexing the web pages.

The proposed indexing has considered the presence of keywords in various HTML tags of web documents. The weight is assigned to each of these tags and stores it using trie structure. This will help to optimize the speed and performance in finding relevant documents for a search query.

2. RELATED WORK

The proposed contextual based indexing [15] has considered the presence of keywords in various HTML tags of web documents such as head, title, keyword, description, body and link. The weight is assigned to each of these tags and stores it using trie structure. This will help to optimize the speed and performance in finding relevant documents for a search query.

L. Huilin [2] has proposed an effective crawling method for focused search engine consisting of two components; filtering and link forecasting. Filtering component filters out the irrelevant documents from previously fetched list of documents and the link forecasting searches the best matched links form related documents.

Rada Mihalcea and Dan Moldovan proposed technique for the semantic indexing [5], which shows improved effectiveness over the classic word based indexing techniques. There are two main tasks of the system, first is the indexing and second is retrieval of components, which uses a combined word-based and sense-based approach.

Focused search engine [3] gives the topic specific information by assembling the web pages for different topics for post-processing. Because of the explicit topic specific documents the performance of a focused crawler is enhanced.

A research proposed by X. Chen, X. Zhang is HAWK crawler [8] and is based on the content of the document and on link analysis. 'HAWK: A Focused Crawler with Content and Link Analysis' is implemented using user-defined relevant formula, shark-search and Page-Rank.

Soumyadeb Mitra [6] proposed a method to update an inverted index based on merging of the posting lists of terms. Jump index is invented by him, which is an efficient index for join queries. Jump indexes execute insert and lookup operations on number of indexed documents.

Context Based Indexing of Web Document using Ontology [7] is the proposed technique of an indexing structure in which index is built on the basis of context of the document rather than on the terms basis using ontology. This ontology-based collection selection method uses context to draw collections and search engines. The crawler collects the context of the documents in the repository is being extracted by the indexer using the context repository and ontology repository and then documents are indexed according to their respective context.

Observations from the existing techniques that show technical gaps are there for accessing the index: The context of the keywords of the user query does not considered by the Focused search engine. Whereas it is essential to focus on the context of the keywords present in the query as well as in the web documents resulting in providing more specific web page.

3. PROPOSED APPROACH

There are so many publicly available repositories which allow for manual tagging and editing on each topic (e.g., DMOZ). There are many documents in the document repository. The textual data associated with the topics comprise a document repository D(R), so that each leaf topic $t \in R$ finds its associated document set $D(t) \subset D(R)$, which describes t itself. For simplicity, here assume that $d(t1) \cap d(t2) = \emptyset$ if $t1 \neq t2$. This shows that, there is only one topic associated with each document in D(R). Thus, for each leaf topic $t \in R$, it is possible to generate an inverted-index, denoted by I[t], containing entries like {term; doc_id; topic_id} for all documents in D(t).

In the end, a hierarchy of inverted indices is obtaining, where each index file I[t] contains all the documents within the taxonomy of t. This inverted index structure enables us to efficiently search keyword and retrieval of document. The entire document set is maintained by the root index file I[Top], which can support term-based topic searching in R.

The existing architecture [11] of search engine shows that the index is built on the basis of the terms of the document and consists of an array of the posting lists where each posting list is associated with a term and contains the term as well as the identifiers of the documents containing the term. Terms are used in the current information retrieval systems to describe documents and search engines.

The proposed indexing considers the senses of the keyword. Contextual senses are the different meanings of the same word. Different users type the same query to get the different results according to their interest. For Example, the keyword bank has different contextual meanings like depository financial institution, a long ridge or pile, a stock held in reserve for future use etc. So the proposed contextual based indexing will provide the different senses to the user which will help in generating the more relevant results to the user.

The proposed architecture for indexer consists of following components: Keyword Extractor, Filterer, Stemmer, Weight Adder, Trie Generator and Inverted Index. Keyword Extractor will extract the keywords from web pages stored in temporary repository i.e csv(comma separated values) file. This file is sent to the filterer. Keyword Filterer matches the list of keywords from csv file with list of stop words and matched words will be eliminated from the list. Keyword Stemmer stems list of keywords to its root from like playing to play etc. Then duplicate keywords will be merged and their weights will be added. Then Trie Structure will be generated for the keywords, documents and senses from contextual sense generator. Inverted Index will be created which consists of keyword, Contextual Sense and Document Ids where the keywords are present within. The proposed indexing architecture is shown in Fig 2.

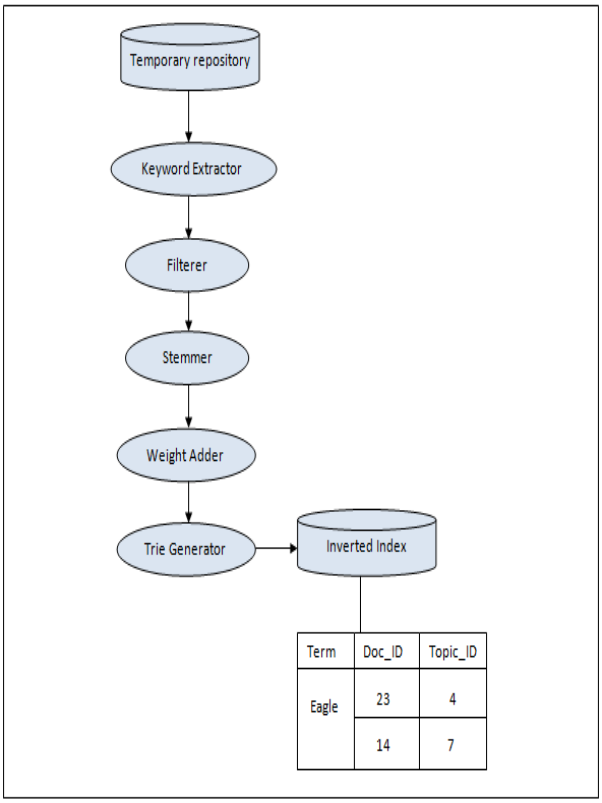

Figure 2: Proposed Architecture for Indexer

Working of each component is shown below:

1) Keyword Extractor*:* Keyword is the smallest unit of a text document. It creates meaning to the document. Keyword extraction means the tokenization of the document. Keyword Extractor parses the web documents which are retrieved from temporary repository. It extracts all the keywords present in web document from different HTML tags and their frequencies in respective tags. The keywords extracted are called as tokens. The specific weights are assigned to the different HTML tags. Then keyword extractor creates a list of all tokens present in that web document with their corresponding Weight, Percent and Total Frequency. This list is then stored in comma delimited report format.

2) Filterer: Keyword filtering means removing stop words from the list. Stop words are the keywords those occur frequently in the web page but do not take part into the context of web document. The list of all the stop words is stored in a file, for example, a, an, the, and, it, to, in, of etc. This list is then matched with the list read from the keyword list of document and the words which are common in both the lists are removed from the csv file.

3) Stemmer: Stemming is a process which reduces a word to its stem or root form. It is performed generally by removing the prefix of the words. For example, the word banking is reduced to bank by splitting from the suffix. Thus, the key terms of a query or document are represented by stems rather than by the original words. For this purpose, Porter Stemming Algorithm [14] is used. After stemming has been done, the new result is stored in the file for the next module processing.

4) Weight Adder: It searches the file after stemming to find out the occurrence of a keyword at multiple places in different form of text fonts. For example, the occurrence of keyword "Eagle" in the web document can be in different forms such as Eagle, eagle, EAGLE. It converts all these occurrences of word to lowercase (eagle). It updates the file by the occurrence of that keyword by one and the corresponding weight by the sum of weights of all the earlier occurrences.

5) Trie Generator: It generates the data structure in the form of trie which actually produce the index of all these keywords or terms of a web document. Trie is used to store pieces of data that have a key and possibly a value which is any additional data associated with the key. Trie has its origin from the core section of the word "**retrieval",** and this origin hints on its usage: information retrieval systems.

The hierarchical depth of the trie data structure depends on the amount of data stored in it. At the highest level of the hierarchy each element of data is stored that still allows a unique retrieval. The data structure has been maintained in search-insert fashion, i.e., when a new word is encountered, it get inserted at proper places and accordingly the trie is updated.

Trie structure appear as a tree type structure which will store the keyword, for example, eagle in tree form as shown in Fig 3.

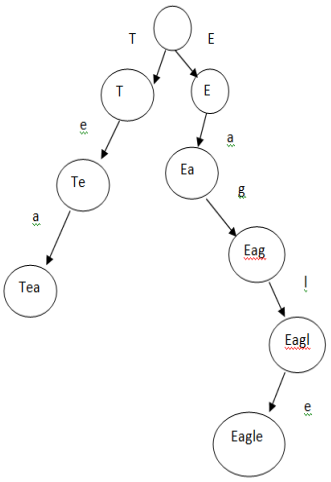

Figure 3 Trie Structure

Table 1 Trie Structure

Term	Doc_ID	Topic_ID
Eagle	23	4
	14	7
	7	13

6) Inverted Index: It is created by trie generator. Each entry of inverted index is a data structure which stores the keyword, topic ids and corresponding document ids. By a tree searching algorithm, this inverted index can be explored to acquire a match for a particular query keyword in search-insert fashion.. Generated inverted index is in the following form { keyword , document id, topic id, frequency, contextual senses}

4. RESULT

Relevant Mathematics

The database contains the number of keywords and their respective links. It may happen, same keywords can extract from the number of links. So for calculating the frequency of those keywords, need to find the rank of each keyword from the database. It depends on the incoming links and outgoing links. To minimize the calculation of ranking value, dumpling factor can use which is 0.8 here.

$$R = \frac{IC}{IC+OG} + \frac{Freq.of\ keywords * 0.8}{No.of\ keywords}$$

Here, R is the value of rank. IC and OG is an incoming link and outgoing links respectively. 0.8 is dumpling factor.

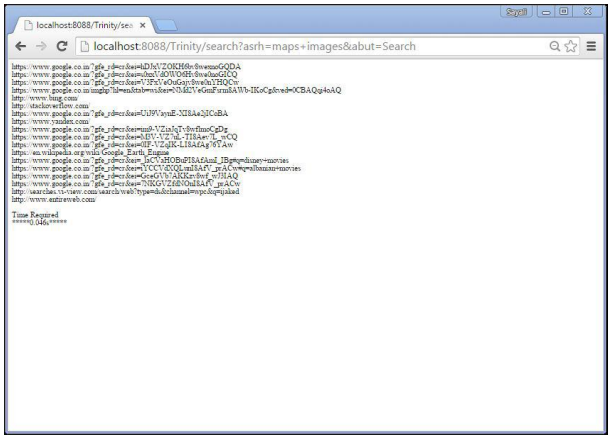

Figure 4 Result as links with ranking

There is execution time in CPU milliseconds (ET). Figure 5 shows the result for indexing mechanism execution time. There are total three web sites from which we can calculate executing time. This graph is for comparisons of execution time values of different web sites.

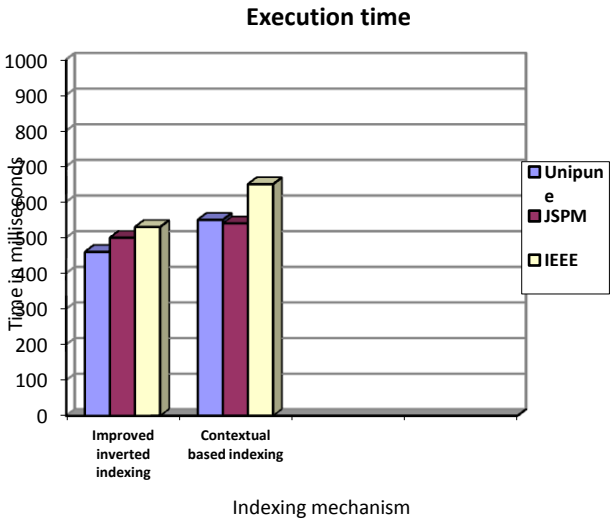

Figure 5 Comparison between execution time

Table 2. Comparison between proposed and existing indexing Mechanism

Mechanism	Document Approach	Document Search	System Performance
Existing	Consider the frequency of keywords in various tags	World Wide Web	Fast access to documents
Proposed	Consider the frequency of keywords from documents in repository	Publicly available repository	Faster than existing

In the existing system because of the network delays are typically much higher than the time needed to process a Web page, it is crucial to send in parallel a large number of requests to different hosts and crawler can only download a limited number of web pages within a given time but here system uses publicly available repository. In proposed system the aim is to remove drawback from the existing system.

5. CONCLUSION

Here, this approach proposed an inverted indexing mechanism for keywords present in the document with their document as well as topic id. And will generate inverted index for the ancestors of the topic in the (term, document, topic) form. The mechanism removes the stop words, stems the keyword and after that creates the index. The search for matched results from the Inverted Index is fastening by the data structure trie. It also helps the user to process the user query with fast and more relevant results.

The main objective of this was to create an indexing method that facilitates finding valuable and relevant information in Web collections. This system is meant to index the entire content of a web collection, and consider whole document and other meta-data for capturing the topics of contents. System uses different indexing algorithms that exploit the query and document structures. In this work, it builds models that can identify relevant documents from Web collections. To do this, system used various representation techniques based on document content, query structures (the variation of query terms occurring in the document content based on query structures), and the applicable knowledge available in Web.

In future we try to find more relevancies between query and documents. Another direction of future work is to connecting our framework to distributed semantics models in order to achieve better empirical coverage of the natural language semantics phenomena.

6. ACKNOWLEDGMENTS

I take this opportunity to thank all in individuals for their guidance, help and timely support .It gives me great pleasure and immense satisfaction to present this paper. Which result of unwavering support, expert guidance and focused direction of my guide Prof.Satish R Todmal to whom I express my deep sense of gratitude and humble thanks, for valuable guidance throughout the work.

7. REFERENCES

[1] Changshang Zhou, Wei Ding, NaYang, "Double Indexing Mechanism of Search Engine based on Campus Net", *Proceedings 2006 IEEE Asia-Pacific Conference on Services Computing (APSCC'06)*.

[2] L. Huilin, K. Chunhua and W. Guangxing, "Efficiently Crawling Strategy for Focused Searching Engine", *Advances in Web and Network Technologies and*

Information Management, Lecture Notes in Computer Science, 2007, Vol. 4537/2007, 25-36.

[3] S. Chakrabarti, M. Berg, B. Dom,"Focused crawling: a new approach to topic-specific Web resource discovery", *The International Journal of Computer and Telecommunications Networking*, Volume 31 Issue 11-16, 1999.

[4] G.Salton and M.J.McGill , "An Introduction to Modern Information Retrieval" ,*McGaw-Hill*,1983.

[5] Rada Mihalcea and Dan Moldovan, "Semantic Indexing using WordNet senses", *in proceedings of ACL workshop on IR and NLP*, Hong Kong, 2000.

[6] S. Mitra, M. Winslett, Windsor W. Hsu and K. Chen-Chuan, "Trustworthy keyword search for compliance storage", *VLDB*, 2008, J.17(2), pp 225-242.

[7] Parul Gupta, Dr. A.K. Sharma, "Context based Indexing in Search Engines using Ontology", *International Journal of Computer Applications*(0975-8887), Vol.1-14,2010.

[8] X. Chen, X. Zhang, " HAWK: A Focused Crawler with Content and Link Analysis", *IEEE International Conference on e-Business Engineering*. pp- 677-680, 2008.

[9] Zhiqiang Wang and Ruifan Li , ''An Index Design in Topicfocused Search Engine", Centre for Intelligent Science and Technology Beijing University of Posts and Telecommunications.

[10] S Büttcher, CLA Clarke , "Indexing time vs. query time: trade-offs in dynamic information retrieval systems", *Proceedings of the 14th ACM international in 2005.*

[11] S.Brin and L.Page, '' The Anatomy of a Large-Scale Hypertextual Web Search Engine*". In: Seventh International World-Wide Web Conference (WWW 1998)*, April 14-18, 1998, Brisbane, Australia.

[12] S.Chakrabarti, B.Com, P.Raghvan, S.Rajagopalana,D.Gibson, and J.Kleinberg, "Automatic Resource compilation by analyzing hyperlink structure associate text," *in Proc 7th World Wide Web Conference* , Brisbane, Australia,1998.

[13] Elizabeth Shanthi and R. Nadarajan, "An Index Structure for Fast Query Retrieval in Object Oriented Data Bases Using Signature Weight Declustering", *Information technology Journal*, vol-8, issue-3, pp.275-283, 2009.

[14] M.F., Porter, "An algorithm for suffix stripping", Program, vol. 14, pp. 130-137. 1980.

[15] Pooja Mudgil, A. K. Sharma, Pooja Gupta, "An Improved Indexing Mechanism to Index Web Documents", *2013 5th International Conference on Computational Intelligence and Communication Networks*, © 2013 IEEE DOI 10.1109/CICN.2013.101