International Journal of Computer Applications (0975 – 8887)
Volume 31– No.6, October 2011

Crawler Indexing using Tree Structure and its Implementation

Deepika Sharma
Department of Computer Engineering
Apeejay College of Engineering

Parul Gupta
Department of Computer Engineering
YM.CAUST

Dr. A.K. Sharma
Department of Computer
YM.CAUST

ABSTRACT

The plentiful content of the World-Wide Web is useful to millions. Information seekers use a search engine such as Google, Yahoo etc to begin their Web activity. Our aim is to make a search tool that is cost-effective, efficient, fast and user friendly. In response to a query, it should retrieve the most relevant information which has been stored into the database. It should also be portable, so that it can easily be deployed at any platform without any cost and inconvenience. Our goal is to make a Web Search Engine that will retrieve the best matched WebPages in the shortest possible time. This paper proposes an algorithm for crawler in which crawler crawls the WebPages recursively and stores the relevant data in the database. The algorithm uses the basic principles of tree structure while maintaining the crawled data by the crawler to be used by the search engine. The proposed work makes the searching on the web more efficient. It uses the tree/node structure in the database which filters the searched word more efficiently and gives faster results to the user. The paper has also implemented the crawler indexing with tree structure using HTML based Update File at Web Server' while making the crawling and searching more efficient.

1. INTRODUCTION

The World Wide Web consists of millions of web sites. Each of these websites themselves contains a number of web pages. It is important to have some sort of a mechanism that would allow an internet user to search through the internet for specific data. Imagine a student, who wants to search for information about his project, searching each and every website which he must know. Search engines come to our rescue in such cases. With a search engine, all the student has to do is type in "keywords" relating to the information that he needs. The search engine would then return a set of results that match best with the keywords entered. A Web Search Engine can therefore be defined as a software program that takes input from the user, searches its database and returns a set of results. It is important to note here that the search engine does not search the internet; rather it searches its database, which is populated with data from the internet by its crawler(s). Web search engines work by storing information about many web pages, which they retrieve from the WWW itself. These pages are retrieved by a Web crawler which follows every link it sees. Exclusions can be made by the use of robots.txt. The contents of each page are then analyzed to determine how it should be indexed (for example, words are extracted from the titles, headings, or special fields called meta tags). Data about web pages are stored in an index database for use in later queries. Therefore, we chose to develop a Web Search Engine to understand the complex mechanism that takes place when we give just a few keywords as input and receive a set of valid and well organized results. The algorithm proposed in this paper has used the basic principles of tree structure while maintaining the crawled data by the crawler to be used by the search engine. This makes the searching on the web more efficient. It uses the tree/node structure in the database which filters the searched word more efficiently and gives faster results to the user.

2. RELATED WORK

In this paper, a review of previous work on index organization is given. In this field of index organization and maintenance, many algorithms and techniques have already been proposed but they seem to be less efficient in efficiently accessing the index.
In [1], the authors introduce a double indexing mechanism for search engines based on campus Net. The Campus Net Search Engine (CNSE) is based on full-text search engine, but it is not general full-text search engine as it is basically a private net. The CNSE consists of crawl machine, Chinese automatic segmentation, index and search machine. They proposed double-indexing mechanism, which means, it has document index as well as word index. The so-called document index is based on the documents do the clustering, and ordered by the position in each document. In the retrieval, the search engine first gets the document id of the word in the word index, and then goes to the position of corresponding word in the document index. Because in the document index, the word in the same document is adjacent, the search engine directly compares the largest word matching assembly with the sentence that users submit. The mechanism proposed by them seems to be time consuming as the index exists at two levels.
Another work proposed was the reordering algorithm [2] which partitions the set of documents into k ordered clusters on the basis of similarity measure. According to this algorithm, the biggest document is selected as centroid of the first cluster and n/k1 most similar documents are assigned to this cluster. Then the biggest document is selected and the same process repeats. The process keeps on repeating until all the k clusters are formed and each cluster gets completed with n/k documents. This algorithm is not effective in clustering the most similar documents. The biggest document may not have similarity with any of the

documents but still it is taken as the representative of the cluster.
Another proposed work was the threshold based clustering algorithm [3] in which the number of clusters is unknown. However, two documents are classified to the same cluster if the similarity between them is below a specified threshold. This threshold is defined by the user before the algorithm starts. It is easy to see that if the threshold is small, all the elements will get assigned to different clusters. If the threshold is large, the elements may get assigned to just one cluster. Thus the algorithm is sensitive to specification of threshold.

3. PROPOSED WORK

For finding the pages the crawler visits the web server for the first time looking at the Robot.txt as the reference. This file contains URLs of updated pages. This file informs updates to the crawler. File is placed at the root of website. When pages in website are updated, web manager puts the URLs of updated pages on this file. Crawler will only visit this file and updated pages for updates, instead of visiting the full website. New updates are at the top and old updates at the bottom. The results are then stored in the local database of the search engine with the details of the URL, metadata/keywords.
The algorithm proposed in this paper crawls the web pages recursively and stores the relevant data in the database. This data includes title, meta keywords, meta title, meta description, body etc of the Webpage. When a query is submitted to the search engine, it searches its own database in response to it. In universal search, it lists all the URLs that match the query. The proposed algorithm has used the basic principles of tree structure while maintaining the crawled data by the crawler to be used by the search engine. This makes the searching on the web more efficient. It uses the tree/node structure in the database which filters the searched word more efficiently and gives faster results to the user. The storage of this data is done by storing each keyword in the webpage in a separate row with the first character in the root column. As a result of this when a user searches for a word, instead of searching the complete data set, it will look for the root word and will search selectively which will significantly reduce the search time. The paper has also implemented smart approach to reduce the web crawling traffic of existing system using HTML based Update File at Web Server while making the crawling and searching more efficient.

3.1 Proposed algorithm for Crawling

1. Crawler visits all web pages of website for first time.
2. It uses Robot.txt for reference.
3. Crawler checks the updates and compares with its own last visit.
4. If updates in file are new for crawler, crawler visits the updated pages and download pages for indexing.
5. Crawler splits the keywords of each webpage.
6. For all separated keywords, Crawler takes the first alphabet of keyword.
7. Crawler Initialize first alphabet of keyword as root of the keyword.
8. Crawler Stores the keywords in database with their roots.

3.2 Pseudo code for Crawling

TreeNodeslist [] – This array contains the keywords from the document.

```
    1.  For (each document in local database) do
                  Extract all different keywords from the
        document
                  TreeNodeslist [] = all the keywords
        from a document.

    2.  For (i=0;i<=length of TreeNodeslist [];i++)
    3.  If (TreeNodeslist [] ==’A’ to ‘Z’)

               Root [BST] = Keyword
       Else
           If TreeNodeslist [] ==value(node[i]) then
                  Add the pointer to this document in the
list of pointers to documents
  End of for in step 2
End of for in step 1
```

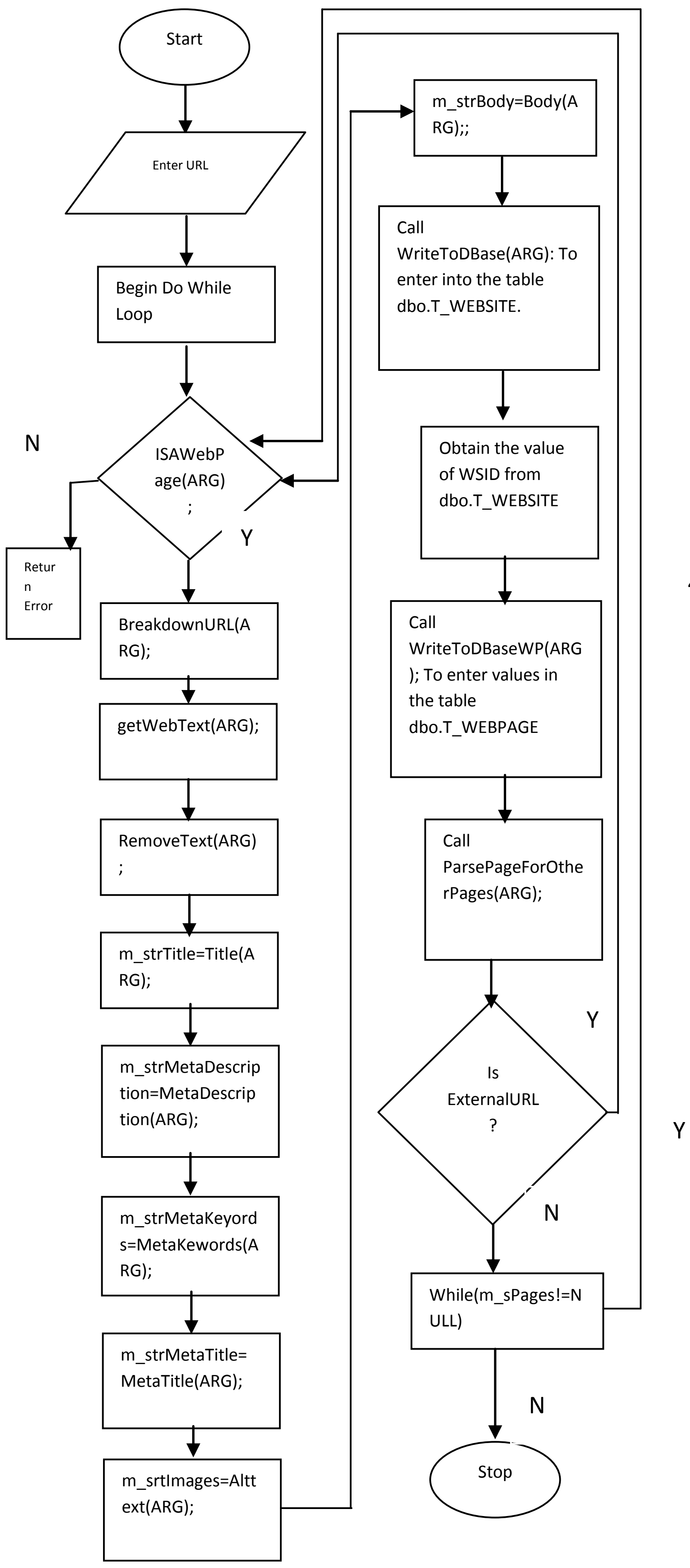
4. IMPLEMENTATION DETAILS

The proposed crawler algorithm implemented for the search engine suggests that the time taken to search the keywords on the web will be reduced significantly. Search engine with **‘Crawler indexing with tree structure’** will help in segregating the meta data and will help in reducing the search time. However the test collection was too small to allow the effectiveness of the ‘Crawler indexing with tree structure’ to be assessed.
Furthermore it covers:

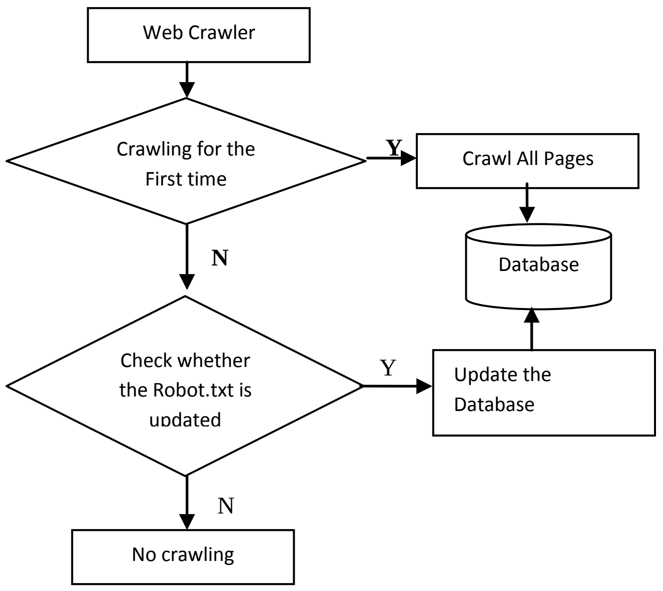
- Demonstrate the use old approach of crawling with Robot.txt.
- Demonstrate the search results with the Old crawler approach.
- Demonstrate the proposed crawler approach with improvements on the older approach
- Demonstrate the search results with proposed ‘Crawler indexing with tree structure’

Old Crawler Approach: We have implemented the old crawler approach in which Robot.txt is referenced for any new updates or addition of web pages in WWW. For the first time all the pages in the web are crawled and stored in the local database of the search engine. In this approach for each webpage URL is maintained with all the keywords in that HTML pages stored in the same row.
Proposed crawler approach: The proposed approach shows that the improvements done are on the indexing of the crawler by storing the URL’s with each keyword in separate row in the local database along with the root word in the root column. This makes the data segregation in more details in the database and looking at the root word the specific url’s are only reported. In case, the root word is referenced to the same URL multiple times, then only one URL search result is displayed.

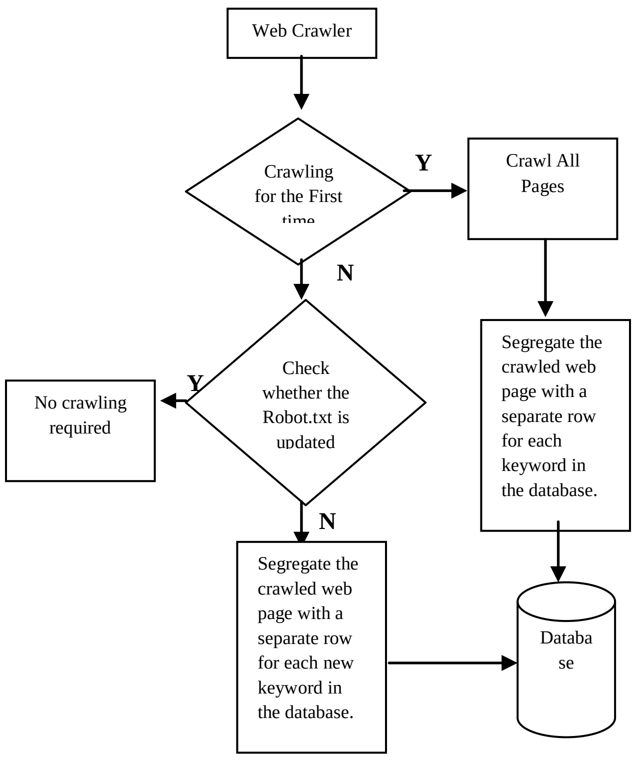
4.1 Flow chart for web crawling


Start
Enter URL
Begin Do While Loop
ISAWebPage(ARG);
N
Y
Return Error
BreakdownURL(ARG);
getWebText(ARG);
RemoveText(ARG);
m_strTitle=Title(ARG);
m_strMetaDescription=MetaDescription(ARG);
m_strMetaKeyords=MetaKewords(ARG);
m_strMetaTitle= MetaTitle(ARG);
m_srtImages=Alttext(ARG);
m_strBody=Body(ARG);;
Call WriteToDBase(ARG): To enter into the table dbo.T_WEBSITE.
Obtain the value of WSID from dbo.T_WEBSITE
Call WriteToDBaseWP(ARG); To enter values in the table dbo.T_WEBPAGE
Call ParsePageForOtherPages(ARG);
Is ExternalURL ?
Y
N
Y
While(m_sPages!=NULL)
N
Stop

4.2 Flowchart for Old Crawler



4.3 Flowchart for proposed Crawler


Web Crawler
Crawling for the First time
Y
Crawl All Pages
N
Check whether the Robot.txt is updated
Y
No crawling required
Segregate the crawled web page with a separate row for each keyword in the database.
N
Segregate the crawled web page with a separate row for each new keyword in the database.
Database

5. RESULTS

5.1 The results obtained with from the base version approach are as under:

Column A = Contain experiment number.
Column B = Contain the updated page(s).
Column C = Contain URL of pages visited by crawler
Column D = Contain the start time of Crawler (Millisecond)
Column E = Contain the time to reach that page. (Millisecond)
Column F = Time spend to visit particular page (Millisecond)

A	B	C	D	E	F
1	Index	http://localhost:9254/CrawWeb/update.jspn	1287490380435	1287490380669	234
		http://localhost:9254/CrawWeb/index.jsp		1287490380747	312
2	P1	http://localhost:9254/CrawWeb/update.jsp	1287490577650	1287490577837	187
		http://localhost:9254/CrawWeb/P1.jsp		1287490577915	265
3	P23	http://localhost:9254/CrawWeb/update.jsp	1287490730645	1287490730817	172
		http://localhost:9254/CrawWeb/p23.jsp		1287490730895	250
4	P11 and P23	http://localhost:9254/CrawWeb/update.jsp	1287477109254	1287477109426	172
		http://localhost:9254/CrawWeb/p11.jsp		1287477109722	468
		http://localhost:9254/CrawWeb/p23.jsp		1287477109987	733
5	P11, P22 and P33	http://localhost:9254/CrawWeb/update.jsp	1287477421879	1287477422237	358
		http://localhost:9254/CrawWeb/p33.jsp		1287477422549	670
		http://localhost:9254/CrawWeb/p22.jsp		1287477422783	904
		http://localhost:9254/CrawWeb/p11.jsp		1287477422846	967

Fig 1 Old Crawler timings

A	B	C	D	E	F
1	Index	http://localhost:9254/CrawWeb/update.jsp	1287490380435	1287490380639	204
		http://localhost:9254/CrawWeb/index.jsp		1287490380727	292
2	P1	http://localhost:9254/CrawWeb/update.jsp	1287490577650	1287490577827	167
		http://localhost:9254/CrawWeb/P1.jsp		1287490577890	240
3	P23	http://localhost:9254/CrawWeb/update.jsp	1287490730645	1287490730787	142
		http://localhost:9254/CrawWeb/p23.jsp		1287490730875	230
4	P11 and P23	http://localhost:9254/CrawWeb/update.jsp	1287477109254	1287477109386	132
		http://localhost:9254/CrawWeb/p11.jsp		1287477109702	448
		http://localhost:9254/CrawWeb/p23.jsp		1287477109947	693
5	P11, P22 and P33	http://localhost:9254/CrawWeb/update.jsp	1287477421879	1287477422197	318
		http://localhost:9254/CrawWeb/p33.jsp		1287477422509	630
		http://localhost:9254/CrawWeb/p22.jsp		1287477422753	874
		http://localhost:9254/CrawWeb/p11.jsp		1287477422816	967

Fig 2 Proposed Crawler timings

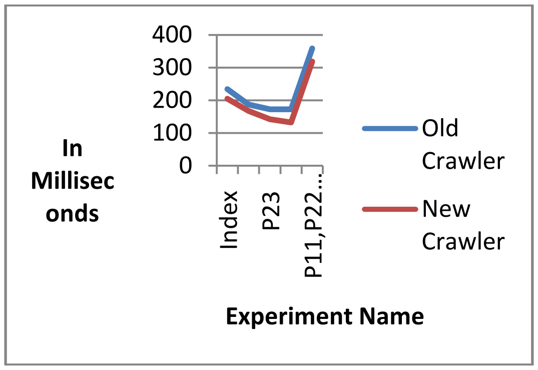
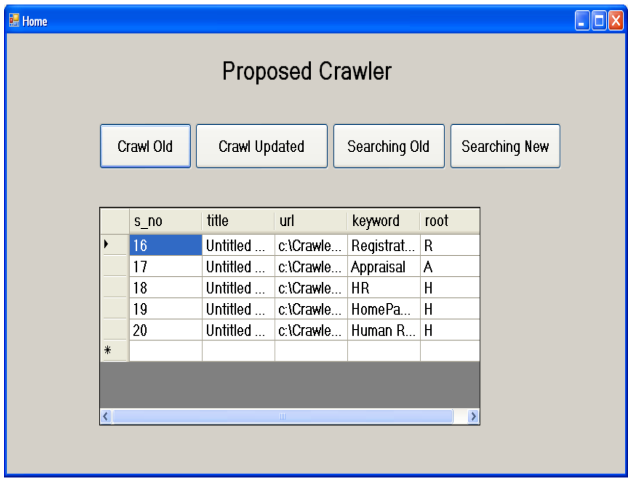


Fig 3 Graphical Time Difference

5.2 Components List and description

- **Home Page**
  - First Screen of our implemented project , It provide the Links to move to another screens of our project as well as it displays the data in the form of table which contains indexed data from our proposed algorithm.
- **Crawl_Old.cs**
  - Another screen , i.e Crawl Old which following the base algorithm of our project , it crawls the websites on the basis of old algorithm and display the data in the form of table like structure.
- **Crawl_New.cs**
  - Another screen, i.e Crawl New This Screen Performs crawling on the basis of our proposed algorithm, it splits the different keywords and initializes them root and stores in the database.
- **Searching.cs**
  - Searching Screen is the form in which we can place our query keywords and perform searching on them. Searching will be done using our proposed algorithm it reduces the time of searching.
- **Searching Old.cs**
  - Searching old screen is based on searching data gathered by base algorithm. It searches the complete database to search the keywords.
  - That’s why it take more time.

5.3 Snapshots of Proposed Crawler Application



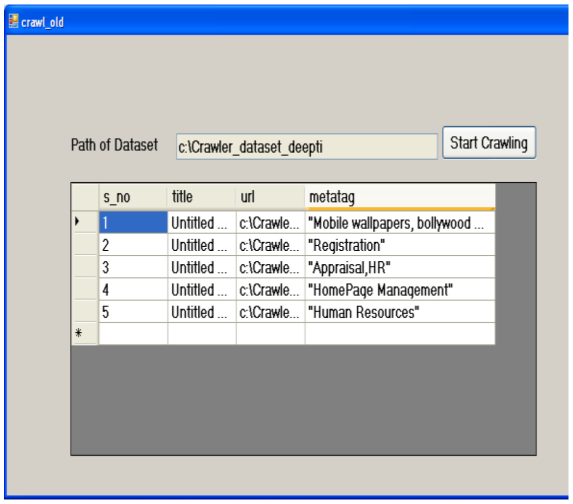
s_no	title	url	keyword	root
16	Untitled ...	c:\Crawle...	Registrat...	R
17	Untitled ...	c:\Crawle...	Appraisal	A
18	Untitled ...	c:\Crawle...	HR	H
19	Untitled ...	c:\Crawle...	HomePa...	H
20	Untitled ...	c:\Crawle...	Human R...	H

Fig 4 Home Page

5.4 Navigation Details

The below section describes the navigation of all the screens for the 'Proposed Crawler application'

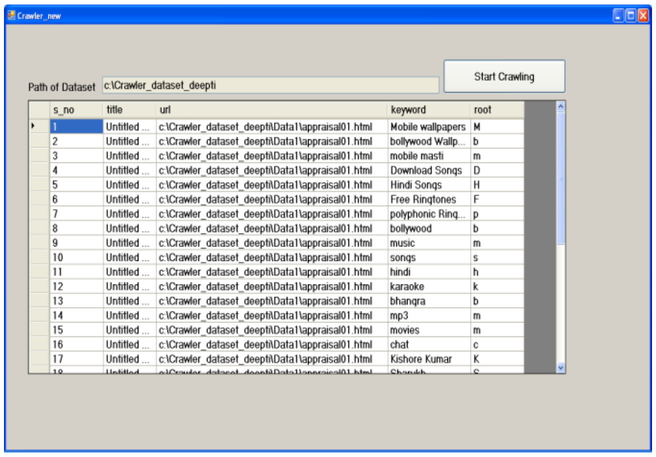
- Crawl Old → Crawl Old.cs (Base Crawler Page)
- Crawl New → Crawl New.cs (Proposed Crawler Page)
- Searching Old → Searching Old.cs (Base Crawler Searching Page)
- Searching New → Searching New.cs (Proposed Crawler Searching Page)



Path of Dataset c:\Crawler_dataset_deepti

s_no	title	url	metatag
1	Untitled ...	c:\Crawle...	"Mobile wallpapers, bollywood ...
2	Untitled ...	c:\Crawle...	"Registration"
3	Untitled ...	c:\Crawle...	"Appraisal,HR"
4	Untitled ...	c:\Crawle...	"HomePage Management"
5	Untitled ...	c:\Crawle...	"Human Resources"

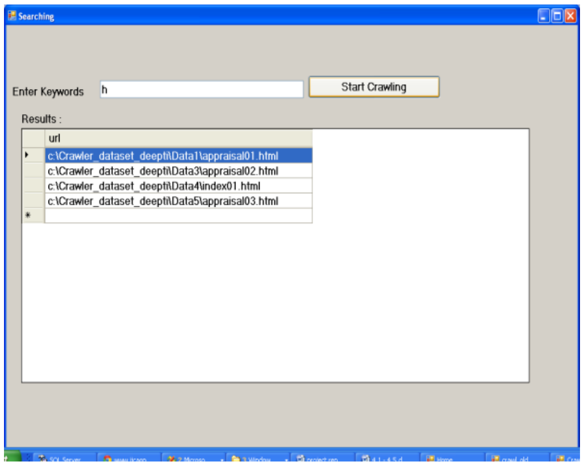
Fig 5 Base Crawler



Path of Dataset c:\Crawler_dataset_deepti

s_no	title	url	keyword	root
1	Untitled ...	c:\Crawler_dataset_deepti\Data1\appraisal01.html	Mobile wallpapers	M
2	Untitled ...	c:\Crawler_dataset_deepti\Data1\appraisal01.html	bollywood Wallp...	b
3	Untitled ...	c:\Crawler_dataset_deepti\Data1\appraisal01.html	mobile masti	m
4	Untitled ...	c:\Crawler_dataset_deepti\Data1\appraisal01.html	Download Songs	D
5	Untitled ...	c:\Crawler_dataset_deepti\Data1\appraisal01.html	Hindi Songs	H
6	Untitled ...	c:\Crawler_dataset_deepti\Data1\appraisal01.html	Free Ringtones	F
7	Untitled ...	c:\Crawler_dataset_deepti\Data1\appraisal01.html	polyphonic Ring...	p
8	Untitled ...	c:\Crawler_dataset_deepti\Data1\appraisal01.html	bollywood	b
9	Untitled ...	c:\Crawler_dataset_deepti\Data1\appraisal01.html	music	m
10	Untitled ...	c:\Crawler_dataset_deepti\Data1\appraisal01.html	songs	s
11	Untitled ...	c:\Crawler_dataset_deepti\Data1\appraisal01.html	hindi	h
12	Untitled ...	c:\Crawler_dataset_deepti\Data1\appraisal01.html	karaoke	k
13	Untitled ...	c:\Crawler_dataset_deepti\Data1\appraisal01.html	bhangra	b
14	Untitled ...	c:\Crawler_dataset_deepti\Data1\appraisal01.html	mp3	m
15	Untitled ...	c:\Crawler_dataset_deepti\Data1\appraisal01.html	movies	m
16	Untitled ...	c:\Crawler_dataset_deepti\Data1\appraisal01.html	chat	c
17	Untitled ...	c:\Crawler_dataset_deepti\Data1\appraisal01.html	Kishore Kumar	K
18	Untitled	c:\Crawler_dataset_deepti\Data1\appraisal01.html	Sharukh	S

Fig 6 Proposed Crawler

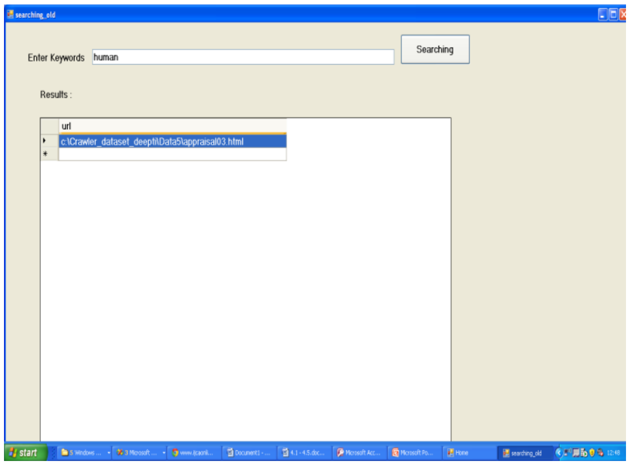


Enter Keywords h

Results :

url
c:\Crawler_dataset_deepti\Data1\appraisal01.html
c:\Crawler_dataset_deepti\Data3\appraisal02.html
c:\Crawler_dataset_deepti\Data4\index01.html
c:\Crawler_dataset_deepti\Data5\appraisal03.html

Fig 7 Searching



Enter Keywords human

Results :

url
c:\Crawler_dataset_deepti\Data5\appraisal03.html

Fig 8 Search Results

6. CONCLUSION

The Websites submitted to the Crawler were crawled without any issues. The number of WebPages and the rates, at which they are crawled, depends on the speed of the Internet. All the search results in response to a query are successfully retrieved. The time taken for the retrieval of results is a function of the size of the database. So, the aim has been achieved by developing a search tool that gives the most relevant output in response to a query. The algorithm proposed and implemented in this paper is portable, cost effective and efficient. It gives its users, relevant results at a faster rate. It also has a user friendly interface.

7. REFERENCES

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

[2] Fabrizio Silvestri, Raffaele Perego and Salvatore Orlando. Assigning Document Identifiers to Enhance Compressibility of Web Search Engines Indexes. In the proceedings of SAC, 2004.

[3] Oren Zamir and Oren Etzioni. Web Document Clustering: A feasibility demonstration. In the proceedings of SIGIR, 1998.

[4] A. Jain and R. Dubes. Algorithms for Clustering Data. Prentice Hall, 1988

[5] Berners-Lee, T., Hendler, J. and Lassila, O., “The Semantic Web,” Scientific American.284(5):35-43, 2001.

[6] O. Zamir, O. Etzioni, O. Madanim, and R.M. Karp, “Fast andIntuitive Clustering of Web Documents,” Proc. Third Int’l Conf. Knowledge Discovery and Data Mining, pp. 287-290, Aug. 1997.

[7] Wang Jicheng, Huang Yuan, Wu Gangshan and Zhang Fuyan, ‘Web Mining: Knowledge Discovery on the Web’ ,IEEE (1999).

[8] Frawley, W., Piatetsky-Shapiro, G., and Matheus, C., Knowledge Discovery in Databases: An Overview. Ai Magazine, Vol. 13 (1992), pp.57-70.

[9] Changshang Zhou, Wei Ding, Na Yang, Double Indexing Mechanism of Search Engine based on Campus Net, Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing (APSCC'06)

[10] Quan, T. T., Hui, S. C., Fong, A. C. M., and Cao, T. H. (2004). Automatic generation of ontology for scholarly semantic Web. In: Lecture Notes in Computer Science. Vol. 3298. (pp. 726–740).