Volume 4, Issue 5, May 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Efficient Crawling the Deep Web

Priyanka Jain
Student
Computer and Science Engineering
Maharshi Dayanand University,
Rohtak, India

Megha Bansal
Assistant Professor
Computer and Science Engineering
Maharshi Dayanand University,
Rohtak, India

Abstract – Crawlers are programs that automatically traverse the Web, retrieving pages and building a local repository of the portion of the web that they visit. Current-day crawlers retrieve content only from the publicly indexed web, i.e., the set of web pages reachable purely by following hypertext links, ignoring search forms and pages that require authorization or prior registration. In particular, they ignore the tremendous amount of high quality content "hidden" behind search forms, in large searchable electronic databases. In this project we develop a crawler capable of querying and extracting the data from the hidden web database. This system will allow a user to submit desired query or keywords on a form-based interface and crawler extract required data from the hidden web not the hyperlinks that traditional crawlers does. This system will also update the rank of page after searching it by using weighted rank formula and which is further modified by us by adding no of duplicates to increase the rank of page.

Keywords— Hidden web, dynamic query interface, page rank, weighted page rank, data extraction.

I. INTRODUCTION

In this Era of internet everyone wants the best and relevant information, but sometime some pages are remain unsearched or not searched by conventional search engines. These unsearched pages are known as Deep Web Pages. There is lots of information buried in the Deep Web[2], and its quantity and quality are far beyond the "surface web" that traditional search engines can reach. The Deep Web resources may be dynamic content, unlinked content, private web, contextual web, limited access content, etc.

Deep web crawling is the process of collecting data from search interfaces by issuing queries. It has been studied from two perspectives. One is the study of the macroscopic views of the deep web, such as the number of the deep web data sources .the shape of such data sources (e.g., the attributes in the HTML form), and the total number of pages in the deep web. When crawling the deep web, which consists of tens of millions of HTML forms, usually the focus is on the coverage of those data sources rather than exhaustively crawling the content inside one specific data source. That is, the breadth, rather than the depth, of the deep web is preferred when the computing resource of a crawler is limited. In this kind of breadth-oriented crawling, the challenges are locating the data sources, learning and understanding the interface and the returned results so that query submission and data extraction can be automated. Another category of crawling is depth-oriented, focusing on one designated deep web data source, with the goal to garner most of the documents from the given data source.

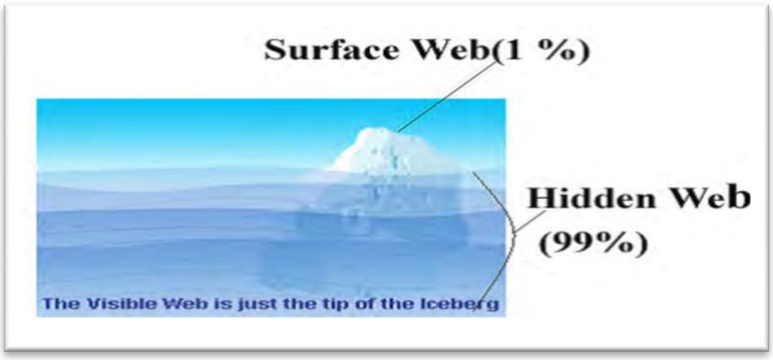

Fig.1 The Deep Web and Surface Web

II. OBJECTIVE

The aim of the project is to develop a crawler capable of querying and extracting the data from the hidden web database. This system will allow a user to submit desired query or keywords on a form-based interface and crawler extract required data from the hidden web not the hyperlinks that traditional crawlers does. In this it will also update the rank of page after searching it.

III. DESIGN OF EFFICIENT DEEP WEB CRAWLER(Related Work)

In this project, we will divide it in three modules .First module will help us to generate the dynamic query interface and second will help us to extract the data from the web and third will update the rank of page by using page rank algorithm.

A. *Dynamic Query Interface :* In this module user selects the URLs of websites from the given URL list and after selecting the one , two or more URLs , a dynamic query interface[3..7] is generated. This Interface focused mainly on analysing domain-specific Deep Web pages, concentrating on identifying those sections of the web pages that contain forms for querying purposes. The interface that contains the common attributes of the selected websites's query forms. After the query interface was developed, it is used to submit user defined queries to multiple Deep Web databases. Search Query Interface is considered as an entrance to the websites that are powered by backend databases. User can find the desired information by submitting the queries to these interfaces. These queries are constructed as SQL queries to fetch data from hidden sources and send it back to user with desired results. The proposed approach is presented in four phases. Firstly, different query interfaces are analysed to select the attribute for submission. In the second phase, queries are submitted to interfaces. Third phase extracts the data by identifying the templates and tag structures. Fourth phase integrates the data into one repository with all duplicate records removed. Query interfaces typically use HTML forms to retrieve data from databases. The building blocks of HTML forms are elements such as tables, frames, and fields. Form fields have various formats but those of interest are the standard HTML input fields: text boxes, text areas, selection lists, radio buttons and check boxes. These typically require a user to fill them in. A field has three important properties, form name, a text label and drop down lists that are exploited in schema integration and label matching.
An example domain for a field with label "**To City**" in the airline domain could be a dropdown selection list of possible destination cities (or data instances).Most HTML forms retrieve data using Common Gateway Interface (CGI) requests. CGI HTTP GET and HTTP POST requests are encoded in the HTML code. I use get method in this project.
<form name="aspnetForm" method="get">
These requests are converted to database queries with parameters supplied by a user query. Thus CGI requests access the deep Web. The terms query interface and query form refer to the same thing, i.e. HTML forms, and are used interchangeably in the literature.

B. *Data Extraction :* Since the hidden web is the biggest source for structured data and is not publicly indexed yet, accessing the same is a challenging task especially when the pages are created dynamically through search interfaces. The problem is how to design an interface for hidden web that can take query respective to any domain and return the result corresponding to that domain only. This Project proposes the solution to it. First task is to read the files containing the data of hidden web that is in structured form of various different domains and to index this data. Then a search query interface is designed that can take the query regarding any domain and return the data specific to that domain only. Several online databases provide dynamic query-based data access through their query interfaces, instead of static URL links. This Query interface is considered as an entrance to Hidden Web, as the tremendous amount of information is hidden behind these search forms in web pages and traditional crawler cannot replicate the query submission carried out by human beings. A Web search interface for e-commerce typically contains some HTML form control elements such as textbox i.e. a single line text input, radio button, checkbox and selection list i.e. a pull-down menu that allow a user to enter search information. A Search interface would provide uniform access to the data sources of a given domain of interest. Here, a Query is considered which contains some terms and it should not be blank. The User enters the query in the text box of the Search Query interface which is then accessed by Query Processor. The two main steps done by Query Processor are extracting the Query String and then tokenizing it. The Tokens are then matched with the attribute values stored in the repository to retrieve the posting lists. These lists are then intersected by the query processor to return the result to the user with the help of a Result Page. This architecture is shown in Fig 2.

C. *PageRank Updating :* Page Rank is a numeric value that represents how important a page is on the web. Page Rank is the Google's Method of measuring a page's "importance". Google uses the Page rank to adjust result so that more important pages moves up in the results page of user's search result display. This module will help us to update the rank of page by using page ranking algorithm[7] .it will update the rank of page after searching the page.

The original formula of PageRank algorithm is:

$$\mathbf{PR(U)=1-d+d(PR(V_1)/N(V_1)+\ldots PR(V_n)/N(V_n))}$$

Where PR (u) = Page rank of page u.

$PR(v_i)$= page rank of page v which link to page

$N(v_i)$= Number of outbound link on page v

D= damping factor which can be set between zero and one.

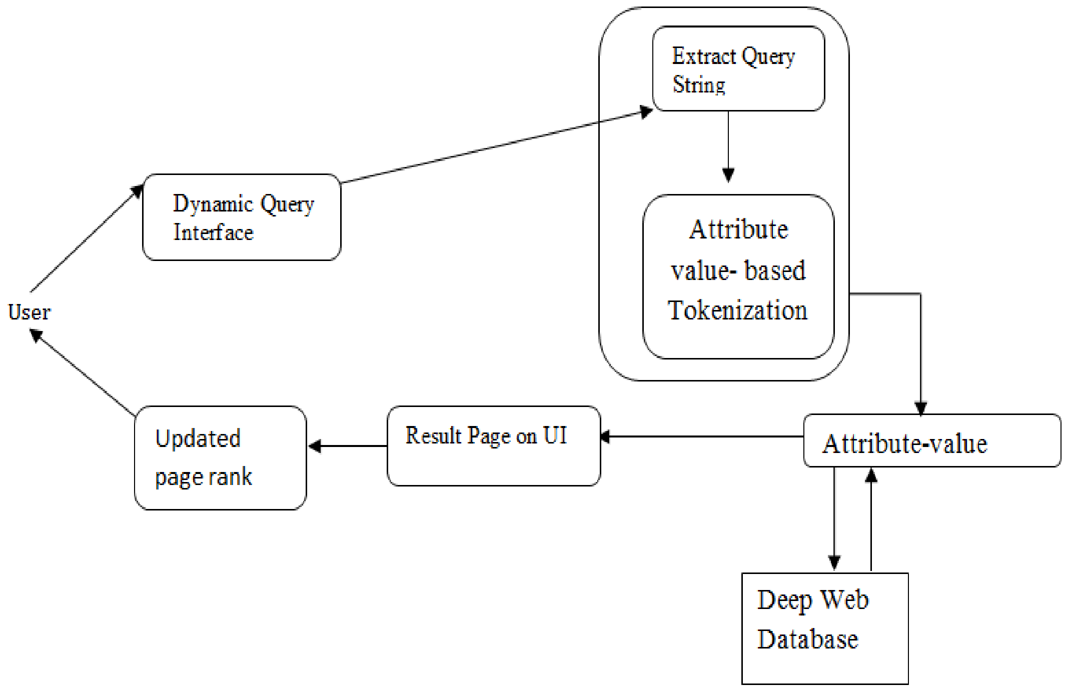

Fig2: Architecture for data extraction

D. *Weighted PageRank*: The *weighted PageRank algorithm* (WPR)[8], an extension to the standard PageRank algorithm, is introduced. WPR takes into account the importance of both the inlinks and the outlinks of the pages and distributes rank scores based on the popularity of the pages.

This algorithm assigns a larger rank values to the more important pages rather than separating the rank value of a page evenly among its outgoing linked pages.

Each outgoing link gets a value proportional to its consequence. The importance is assigned in terms of weight values to the incoming and outgoing links and are denoted as $W^{in}(v, u)$ and $W^{out}(v, u)$ respectively. $W^{in}(v, u)$ as shown in (1) is the weight of link (v, u) calculated based on the number of incoming links of page u and the number of incoming links of all orientations pages of page v.

$$W^{in}_{(v,u)} = \frac{I_u}{\sum_{p \in R(v)} I_p} \qquad \ldots.. (1)$$

Where Iu and Ip are the number of incoming links of page u and page p respectively. R (v) denotes the allusion page list of page v.

$$W^{out}(_{v,u}) = \frac{O_u}{\sum_{p \in R(v)} O_p} \qquad \ldots.. (2)$$

$w^{out}(v,u)$ is as shown in (2) is the weight of link(v, u) calculated based on the number of outgoing links of page u and the number of outgoing links of all reference pages of v. Where Ou and Op are the number of outgoing links of page u and p correspondingly.

The formula as proposed by Wenpu et al for the WPR is as shown in (3) which is a modification of the PageRank formula.

$$PR(u) = (1 - d) + d \sum_{v \in B(u)} PR(v) W^{in}_{(v,u)} W^{out}(_{v,u}) \qquad \ldots\ldots (3)$$

IV. PROPOSED WORK

In this paper , we are going to add a new factor in weighted PageRank formula which increase the rank of page more than normal weighted PageRank does. We have introduced the concept of increasing the rank through duplicate pages which is referred in the same page which is shown in equation(4)

$$\mathbf{D\ (v, u) = D\ (u)/D\ (v)} \qquad \ldots\ldots (4)$$

Here D(u) and D (v) are the no. of duplicates.

The new formula is shown in equation (5):

$$\mathbf{PR(u) = (1-d)+d \ \sum PR(v) * W(in)*W(out)*D} \qquad \ldots.. (5)$$

V. CONCLUSION

Deep Web Crawling is basically used for searching the hidden web pages which is not searched by traditional search. But sometimes this crawler is also not able to find the deep web pages. so we have developed efficient deep web crawler ,by using dynamic query interface to generate query for searching and extract the data from the web and after searching the page it updates the rank of page and we have added no of duplicates to our Weighted PageRank formula to make it more efficient . So through this system we can do efficient crawling.

ACKNOWLEDEMENT

No paper is created entirely by an individual, many people have helped to create this thesis and each of their contribution has been valuable. I wish to express my deep sense of gratitude to my gratitude supervisors Ms. Megha Bansal, Assistant Professor, Computer Sc. & Engineering Department**.** I owe their overwhelming debt for helping me to plan and discuss the contents of this work. Their ideas, stimulating comments, interpretations and suggestions increased my cognitive awareness and helped considerably in the fruition of my objective. I remain obliged to them for her help and able guidance through all stages of this paper. Their constant inspiration and encouragement to my efforts shall always be acknowledged. Her observation and comments were highly enlightening.

REFRENCES

[1] BrightPlanet.com. http://www.brightplanet.com.

[2] The Deep Web: Surfacing Hidden value.http://www.completeplanet.com/Tutorials/DeepWeb/.

[3] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: A new approach to topic-specific web resource discovery. In *Proc. of the* 8*th Intl. WWW Conf.* , 1999.

[4] J. Cho and H. Garcia-Molina. The evolution of the web and implications for an incremental crawler. In *Proc. of the* 26*th Intl. Conf. on Very Large Databases*,2000.

[5] J. Cho and H. Garcia-Molina. Synchronizing a database to improve freshness. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, 2000.

[6] J. Cho, H. Garcia-Molina, and L. Page. Efficient crawling through url ordering. In *Proc. of the* 7*th Intl.WWW Conf.*, 1998.

[7] Raymond Kosala, Hendrick Block33,"web mining research: A Survey",ACM sigkdd Explorations NEWS Letter, June200, volume @.

[8] Wenpu Xing and Ali Ghorbani, "Weighted PageRank Algorithm", Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR '04), IEEE, 2004.

[9] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles,and M. Gori. Focused crawling using context graphs.In *Proc. of the* 26*th Intl. Conf. on Very Large Databases*, pages 527–534, Sept. 2000.

[10] D. Florescu, A. Y. Levy, and A. O. Mendelzon.Database techniques for the world-wide web: A survey.*SIGMOD Record*, 27(3):59–74, 1998.