IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 5, Issue 4, April 2016*

# Smart Crawler: A Two-Stage Crawler for Efficiently Harvesting Deep-Web Interfaces

**Melisa Vidiera[1], Janhavi V[2]**

PG Scholar, Dept of Computer Science and Engineering, Vidyavardhaka College of Engineering, Mysuru, India[1]

Associate Professor, Dept of Computer Science and Engg, Vidyavardhaka College of Engineering, Mysuru, India [2]

**Abstract:** Due to extensive usage of Internet, substantial amount of data has extended widely over web, which serve access to particular data or to fetch more relevant data. It would be challenging to the search engine to provide quick results that is most relevant to the users. To search the relevant data and to reduce amount of time in fetching data, here propose the "Smart Crawler". This returns most relevant data from the popular and most specific websites. It uses multiple search engines that processes the query provided by the user, cluster the results collected in a single platform and performs two stage crawling on data and URLs. In which in-site map generation is done to obtain relevant site with techniques such as reverse searching and page ranking.

**Keywords:** Deep Web, two stage crawler, ranking, in-site exploring, adaptive learning.

## I. INTRODUCTION

Internet is an indivisible and essential part of our day to day life. Internet is mainly used to communication and to get answers for most of the question arises in our daily life. World Wide Web is information space which contains resources and documents that has particular URL. Many search engines are available to extract contents of web and to provide the answers for all types of queries; the popular ones are Google, Yahoo, MSN.

The data that are most relevant to the users, often present in Deep Web. Deep web is also known as dark web and it is concealed web that consist innumerable pages that can be accessed by public, but their IP addresses will be hidden. They cannot be discovered in a single search attempt and it is tough to identify who are people behind that web sites. These contain database information namely catalogues and the reference that are not indexed by any search engine. Most of the search engines fail to exact the data from the deep web. They tend to concentrate in returning much number of results rather in returning accurate and most relevant result to the users that is very much expected. As the size space of web increases, the valuable information cannot be indexed and accessed by the search engines. Based on a study done at University of California, Berkeley, at 2003, it is estimated that overall deep web is of 91,850 terabytes of size and the surface web is only about 167 terabytes. Deep web is about 96% of all the content on the Internet, which is mostly 550 times larger than surface web. To overcome this problem there is a need of efficient harvesting of deep web which explore quickly and return accurate results to the users. It is challenging for search engines to discover and explore the database of deep web as they are not registered with any search engines.

## II. EXISTING SYSTEM

Traditional Crawler browses the World Wide Web and obtains data from indexing. It copies all the pages that have been visited by the crawler which can be processed later by the search engines. The existing system is a manual or semi automated system.

Disadvantages of traditional crawler are, the crawler consumes large amount of data which leads in high traffic and crawling these large size of data waste time. Users are constrained by the limits in bandwidth and processing which leads crawler not to visit deep web and it can omit the websites that has appropriate data.

## III. RELATED WORKS

Luciano Barbosa and Juliana Freire. [1] This paper describes new adaptation crawling techniques that locate the entry points to deep web source efficiently. Hidden web resources are distributed sparsely which make it difficult to locate. This problem can be overcome by focusing on the keyword which is provided by the users that how many times the keyword has been repeated in deep web sites. In the new methodology described in this paper, crawling learns the pattern of appropriate link automatically and then as crawling process progresses, crawler adapt their focus which reduces manual intervention for setup and tuning.

Dr. Jill Ellsworth [2]. This paper focuses problems encountered in deep web. In traditional search engines the contents of deep web is not obtained in the single search result. Search engine fails to access dynamic contents of the deep web pages. Hence deep internet is also known as invisible or hidden internet. Deep web pages have lot of important data and are publicly available but it is not registered to most of search engines. IP addresses are not known to the search engines hence we can't know the people behind deep websites and hence there exists authentication issues.

Raju Balakrishnan and Subbbarao Kambhampati[3].This paper focus on the challenge in extracting information from deep web. Main challenge of deep web is to choose

the relevant data that users expect to obtain as result from the search engine. Enormous amount of useful data are hidden in deep web pages which are not indexed by search engine. Two main issues arise here are, deficiency of crawler to understand whether the information of deep web is trustworthy or not. The second issue is the relevancy of data obtained in contemplates to the importance of results or not. Choosing reliable and relevant data as an answer to the query is very critical issue. The relevancy assessment is primarily bases on similarity of the data fetched to the data that is appropriate to the users.

Cheng Sheng, Nan Zhang, Yufei Tao and Xin Jin[4]. This paper proposes the ace algorithm for progression of hidden information in deep web. This algorithm specified in the paper extract all the tuples from hidden pages. The implemented algorithm is economical which means that the system perform the process with the lesser usage of queries. This implies to worst case too.

M. Burner[5].This paper focuses on the range of web. The system specified uses multiple crawler to crawl 100 million URLs. Each crawler takes seed URL as an input and then uses asynchronous input output instructions to fetch pages from the queue. This fetching of pages is done in parallel manner. The main problem faced during fetching data is changing DNS record, this can be overcome my archiving history of hostnames and their associated IPs.

Olston and M. Najork [6]. Google crawler is based on C++ and python. This has five crawling components that run in various processes to download the pages. Each crawler uses asynchronous input output instructions to fetch data from about 300 web servers. This fetching process is done parallel. Google crawler also adopted indexing method.

Allan Heydon and Marc Najork [7]. The crawler proposed in this paper was developed with high scalability and extensibility. This is based on java. When the first version of crawler is developed it was non-distributive in nature. In the latter versions, distributed mechanism was included in crawler. In distributive mechanism URL space is split up over the crawler with respect to the host name. This method avoids bottleneck in centralized server of URL.

Jenny Edwards, Kevin S. McCurley[8].In 2001 a modular and distributed crawler was presented by IBM. This crawler was written in C++. This crawler incorporated three components namely, Multithreaded crawling process, Duplicated context and Central controlled process. Central control process is responsible for overall operations of the system and it assigns work to other modules present in the system. The crawler uses MPI to facilitate the communication between processes. This mechanism was utilized in a cluster that had 48 crawling machines.

H.-T. Lee, D. Leonard, X. Wang, and D. Loguinov [9]. IRLbot web crawler is a single process web crawler. This was able to scale to extremely huge web contents without undergoing degradation. It crawls over two months and download 6.4 billion of web pages.

## IV. PROPOSED SYSTEM

Smart Crawler is two stage crawler that efficiently harvest deep web. Seed web sites are given as input to crawler. At the first stage crawler determines the most relevant data according to the keyword given by the users. At second stage in-site exploration is done that divulge searchable forms from the sites.
Once the search is made, the relevant URL is stored in the site database. Site locating is done by reverse searching technique, which search for the data second time but this time in reverse manner. This is used to obtain maximum amount of appropriate data. Reverse search is triggered when there are less results than that of threshold specified. To achieve more coverage of web, in-site searching is done in the directories. In the in-site exploring stage, crawler crawls through the pages and finds the hyperlinks present in the pages and add it to the database yet it limits visits to the large number of the sites. This uses Stop Early mechanism and Balanced Link Tree. Stop Early method stops the crawler from visiting to non appropriate websites. Here, simple breadth first method is used that is not efficient. It results in incomplete directory visits and omits highly relevant links. Links are often unevenly distributed across web, this results in biasing on some directories. This problem is overcome by merging trees or directories.

Ranking is done in two phases. First Link Ranker prioritizes links so that the crawler can locate pages that are searchable. The high relevance score is given to those sites which is most similar to that of searchable form pages. At the next phase crawling is focused using Form Classifier that filters irrelevant forms and non-searchable forms from the database.

Site ranking and Link ranking is done using two features that are, Site similarity and Site Frequency. Site similarity measures the similarity of the topic between new site which is encountered by the crawler and deep web sites which are known to crawler. Site frequency is the frequency of the site which appears in other sites that depicts the popularity of the site.
The additional features make Crawling yet better includes, Site Crawling, Accessibility Crawling using site map generation and Security Testing using Web Authentication Crawling. Stress Crawling is executed to evaluate a system, or component at or beyond the limits of its specified requirements. It is used to evaluate system responses at activity peaks that can exceed systems limitations, and to verify if the system crashes or it is able to recover from such conditions. Stress testing differs from performance and load testing because the system is executed on or beyond its breaking points, while performance and load testing simulate regular user activity. Failures found by stress testing are mainly due to faults in the running environment.
Web site map generation is done using accessibility crawling test, that can be considered as a particular type of usability testing whose aim is to verify that access to the content of the application is allowed even in presence of reduced hardware/ software configurations on the client

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 5, Issue 4, April 2016*

side of the application (such as browser configurations disabling graphical visualization, or scripting execution), or of users with physical disabilities (such as blind people). In the case of Web applications, accessibility rules such as the one provided by the Web Content Accessibility Guidelines have been established, so that accessibility testing will have to verify the compliance to such rules. The application is the main responsible for accessibility, even if some accessibility failures may be due to the configuration of the running environment (e.g., browsers where the execution of scripts is disabled).

Security testing is done using web authentication crawling. The objective of security testing is to verify the effectiveness of the overall Web system defences against undesired access of unauthorized users, as well as their capability to preserve system resources from improper uses, and to grant the access to authorized users to authorized services and resources. System vulnerabilities affecting the security may be contained in the application code, or in any of the different hardware, software, middle-ware components of the systems. Both the running environment and the application can be responsible for security failures. In the case of Web applications, heterogeneous implementation and execution technologies, together with the very large number of possible users, and the possibility of accessing them from anywhere may make Web applications more vulnerable than traditional ones and security testing more difficult be accomplished.

## V. ENHANCED ARCHITECTURE

User request web server and expect to obtain the relevant data, user request by providing keyword and request for site map generation, stress crawling, rank of the authenticated website. In the other end, web server receives the user request and crawl all the related data with respect to the keyword provided by the user. Web server in turn request world wide web and the database, and finds site stress and web resource that is authenticated and returns the relevant data to the user along with the web site map.

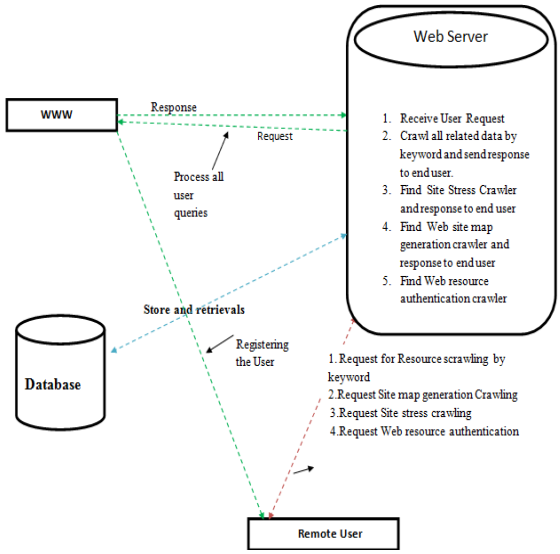


Fig. 1 architecture of enhanced crawling system

## VI. CONCLUSION

This paper proposes better approach to exact the most appropriate data in the deep web sites in an efficient manner. Smart Crawler achieves wide coverage of deep web and returning the accurate results to the users without visiting unnecessary sites and consumes less time which overcomes the cons of the traditional crawler. Smart Crawler reversely performs site-based locating for deep web for centre pages. To achieve correct results it uses ranking mechanism. In-site exploration stage uses adaption of Link ranking and eliminates bias towards specific trees using link tree. The future work includes improving in accuracy of the classifier that uses mix pre-query and post-query approaches.

## REFERENCES

[1] An Adaptive Crawler for Locating Hidden-Web Entry Point, Luciano Barbosa and Juliana Freire.
[2] Understanding the Deep Web, Dr. Jill Ellsworth.
[3] Relevance and Trust Assessment for Deep Web Sources Based on Inter-Source Agreement, Raju Balakrishnan and Subbbarao Kambhampati.
[4] Optimal Algorithms for locomotion a Hidden info within the Web, Cheng Sheng, Nan Zhang, Yufei Tao and Xin Jin.
[5] M. Burner, "Crawling towards Eternity: Building an Archive of the World Wide Web," Web Techniques Magazine
[6] Olston and M. Najork, "Web Crawling", Foundations and Trends in Information Retrieval.
[7] Allan Heydon and Marc Najork. Mercator: A scalable, extensible web crawler. World Wide Web Conference
[8] Jenny Edwards, Kevin S. McCurley, and John A. Tomlin. An adaptive model for optimizing performance of an incremental web crawler. In Proceedings of the Tenth Conference on World Wide Web
[9] H.-T. Lee, D. Leonard, X. Wang, and D. Loguinov, "IRLbot: Scaling to 6 billion pages and beyond,"
[10] A.A. Andrews, J. OVutt, R.T. Alexander, Testing Web applications by modeling with FSMs. Software Systems and Modeling, Springer-Verlag Ed., 2005, 4(2).
[11] A. Bangio, S. Ceri, P. Fraternali, Web modeling language (WebML): a modeling language for designing Web sites, in: Proceedings of the Ninth International Conference on the WWW (WWW9).
[12] R.V.Binder, Testing Object-Oriented Systems-Models, Patterns, and Tools, Addison-Wesley, Boston, MA, USA, 1999.
[13] J. Conallen, Building Web Applications with UML, Addison-Wesley Publishing Company, Reading, MA, 2000.