International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 11 | Nov -2016 www.irjet.net p-ISSN: 2395-0072

# An Efficient Harvesting scheme for Deep Web Interfaces based on Two-Stage Crawler

**Shinde Pavan B[1], Sonkar Shriniwas K[2]**

[1] *Department of Computer Engineering, Amrutvahini College of Engineering, Maharashtra, India*
[2] *Department of Computer Engineering, Amrutvahini College of Engineering, Maharashtra, India*

---

**Abstract -** *The web pages available in the Internet are growing tremendously so that searching relevant information in the Internet is tedious job. A lot of this information is hidden behind query forms that interface to unexplored databases containing high quality structured data. General search engines cannot extract and index this hidden part of the Web, retrieving this hidden data is challenging task. So that large number of web data resources and the dynamic nature of deep web sites, achieving wide coverage and high efficiency is a challenging task. We propose a two-stage framework, namely SmartCrawler, for effective searching deep web interfaces. First stage of SmartCrawler performs site-based searching for pages with the help of web crawler, avoiding visiting a large number of sites. To produce more relevant results for a focused crawl, SmartCrawler ranks links to prioritize highly relevant pages for a given topic. Then in second stage, it achieves fast in-site searching by extracting most relevant links with an adaptive link-ranking.*

***Key Words***: Smart crawler, Deep web, ranking, adaptive learning

## 1.INTRODUCTION

All over the world the internet is a collection of billions of web server containing large bytes of information or data arranged in N number of servers. Its tedious job locate the deep web databases, because they are not recorded by any search engines, are usually sparsely distributed, and keep constantly changing. To overcome above problem, previous work has proposed two types of crawlers, generic and focused crawlers. The Generic crawlers extract all searchable forms and cannot focus on a specific topic. In Focused crawlers such as Form-Focused Crawler and Adaptive Crawler for Hidden-web Entries can automatically search online databases on a particular topic. Form-Focused Crawler is designed with link, page, and form classifiers for focused crawling of web forms, and then by Adaptive Crawler for Hidden-web Entries with additional components for form filtering and adaptive link learner. The link classifiers in these crawlers perform a major role in achieving higher crawling efficiency than the best-first crawler. These link classifiers are used to predict the distance to the page containing searchable forms, which is difficult to calculate, especially when for the delayed benefit links.

The Crawler performs an advanced level of data analysis and data retrieved from the web. The SmartCrawler is divided into two stages- First is Site locating and second is in-site exploring. In the first stage, Crawler performs site-based searching for center pages with the help of search engines, avoiding visiting a number of pages. To achieve more accurate results for a focused crawl, SmartCrawler ranks websites to prioritized highly relevant website for a given topic. In the second stage, SmartCrawler achieves fast in-site locating to excavate most relevant links with an adaptive link-ranking.

## 2.RELATED WORK

There are many search engines written in every programming and scripting language to serve a variety of search engines depending on the requirement, purpose and functionality for which the crawler is created. The first ever web crawler to be built to fully function is the WebCrawler in 1994. Then a lot of other better and more efficient crawlers were built over the recent years. The most notable of the crawlers currently in operation are as follows. But these first generations have some of the issues in web crawling design; it is not focus on scalability.

**A].Internet archive Crawler:**

Mike Burner designed the Internet Archive Crawler in 1997 as the first paper that focused on the challenges caused by the scale of web [2]. It uses multiple machine to extract the web data and it crawl on millions of URLs [1]. Each crawler process read a list of seed URLs for its assigned servers from disk into per-site queue, and then it uses asynchronous I/O data to extract pages from these queues in parallel.

**B].Google Search Engine:**

Later in 1998, The Google search uses this crawling bot. The original Google crawling system consists of a five searching components which was extracting the relevant information in various process and extract the pages [2].

Each crawler process used asynchronous I/O instructions to extract the data from N number of web servers in parallel [1]. Then all the crawlers transmit downloaded links to a single

Data Server process that compressed the page and store them on disk. It has a URL server that exclusively handles URLs. It checks if the URLs have previously been crawled. It they are not crawled they are added to the queue. Google Crawler was based on C++ and Python language tools. This crawler was designed with the indexing process.

**C].Mercator Web Crawler:**

In 1999 Heydon and Najork design a web crawler which was highly scalable and easily extensible [3][1]. It was written in Java language. The first version was non-distributed and later the distributed version was made available which divide the URL space over the crawlers according to host name and URL server.

**D].WebFountain crawler:**

In 2001, IBM presented another distributed and modular crawler [4][1]. It was written in C++ and used MPI to facilitate the communication between the various processes. It has three major component Multi-threaded crawling processes, duplicate content and central controlled process. It was deployed on a cluster of 48 crawling machine. It has a controller and ant machines that repeatedly download pages. A non-linear programming method is used to solve freshness maximizing equations. We also have a lot of open source crawlers that are available online and can be used according to needs for non-commercial purposes.

**E].IRLbot Web crawler:**

Yan et al. describe IRLbot, which is single process web crawler [1]. It crawls over two month and downloads the 6.4 billion web sites.

To leverage the large volume information stored in deep web, previous work has proposed a number of tools and techniques, including deep web understanding and integration, hidden web crawlers, and deep web samplers. For all these approaches, the ability to crawl deep web is a key challenge.

## 3. SYSTEM ARCHITECTURE

An Effective harvesting scheme for Deep Web Interfaces based on Two-stage Crawler performs in two stages like web site locating and in-site exploring, as shown in following Figure.

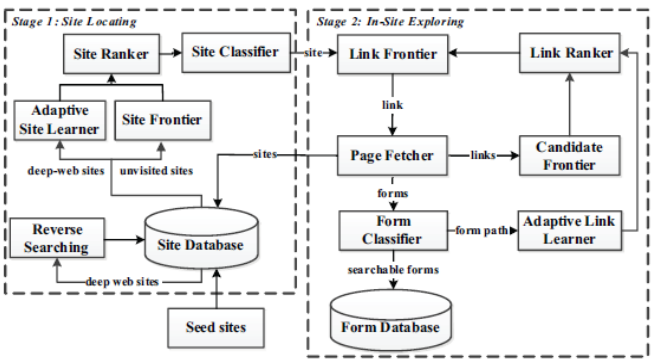


Fig.1. Architecture of SmartCrawler in two stages

At the First stage, SmartCrawler finds the most relevant web site for a given topic and in the second stage will be in-site exploring stage which uncovers searchable content from the site.

**Stage1:** In this stage site locating starts with a seed set of sites in a site database. Seeds sites are candidate sites given for Crawler to start searching, which begins by following links from chosen seed sites to explore other sites and other servers. When the number of unvisited links in the database is less than a threshold during the crawling process, Crawler performs ”reverse searching” of known deep web sites for center pages i.e. highly ranked pages that have many links to other domains and store these pages back to the site database. Site Frontier extracts homepage link from the site databases, which are ranked by Site Ranker to prioritize highly relevant sites. The Site Ranker is improved during crawling by an Adaptive Site Learner, which adaptively learns from features of deep-web sites (web sites containing one or more searchable forms) found. To achieve more correct results for a focused crawl, Site Classifier categorizes links into relevant or irrelevant for a given topic according to the homepage content.

**Stage 2:** After the most relevant site is found in the first stage, the second stage performs efficient in-site exploration for excavating searchable forms. Links of a site are stored in Link Frontier and corresponding pages are extracted and embedded forms are classified by Form Classifier to find searchable forms. Additionally, the links in these pages are extracted into Candidate Frontier. To prioritize links in Candidate Frontier, SmartCrawler sort them with Link Ranker. Note that site locating stage and in-site exploring stage are mutually intertwined. When the crawler discovers a new site, the site’s link is inserted into the Site Database. The Link Ranker is adaptively improved by an Adaptive Link Learner, which learns from the URL path leading to relevant forms.

To address the above problem, we propose two crawling strategies, reverse searching and incremental two-level site prioritizing, to find more sites.

### A. Algorithm

1. Reverse searching for more sites:
Input for system: seed sites and extracted deep websites
Output from System: relevant sites
1 while numbers of candidate sites less than a threshold value
2 // pick a deep website
3 site = getDeepWebSite(siteDatabase,seedSites)
4 resultPage = doReverseSearch(site)
5 links = extracttheLinks(result Page)
6 foreach link in links do
7 page = downloadPages(link)
8 relevant = classifyRelevant(page)
9 if relevant then
10 relevantSites =extractUnvisitedSiteLink(page)
11 Display MostRelevantSites
12 end

13 end
14 end

**B. Algorithm**

1. Incremental site prioritizing.

Making of crawling process resumable and get broad coverage on websites, an incremental site prioritizing strategy is proposed. The idea is to stored learned patterns of deep web sites and form paths for incremental crawling. First we obtain prior knowledge for initializing Site ranker and Link ranker. The unvisited sites are given to Site Frontier and are ranked by Site Ranker, and visited sites are given to extracted site list. The detailed incremental site prioritizing process is described in following Algorithm.

Site Frontier uses two queues to save unvisited sites. First the high priority queue is for out-of-site links that are classified as relevant site by Site Classifier and are judged by Form Classifier to contain searchable forms. The low priority queue is for out-of site links that are classified as relevant site by Site Classifier. The Site Ranker gives relevant scores for prioritizing sites. The low priority queue is used to provide more candidate sites. If the high priority queue is empty, then sites in the low priority queue are transfer into it.

Input : siteFrontier
output: searchable forms and out-of-site links
1 HQueue=SiteFrontier.CreateHQueue(HighPriority)
2 LQueue=SiteFrontier.CreateLQueue(LowPriority)
3 while siteFrontierQueue is not empty then do
4 if HQueue is empty then
5 HQueue.addAllLink(LQueue)
6 LQueue.clear()
7 end
8 site = HQueue.Poll()
9 relevant site = classifySite(site)
10 if relevant link then
11 perform InSite Exploring(site)
12 Output forms and OutOfSiteLinks
13 siteRanker.rankLink(OutOfSiteLinks)
14 if forms link is not empty then
15 HQueue.addLink (OutOfSiteLinks)
16 end
17 else
18 LQueue.add(OutOfSiteLinks)
19 end
20 end
21 end

## 4.MATHEMATICAL MODEL

Let S be the system such that,
S = {s, e, X, Y, A, Q, E…… | Φs}
s - Initial State
e - End State
X - Input = Web URL
Y - Output = Crawler is a focused crawler consisting of two stages:
a) Efficient site locating and
b) Balanced in-site exploring
A - Algorithms:
Reverse searching for more sites
Incremental site prioritizing
Q - Queries
E - Entities
E = {E1, E2}
E1 = User
E2 = Web browser

Comes here Conclusion content comes here Conclusion content comes here Conclusion content comes here . Conclusion content comes here

## 5.RESULTS

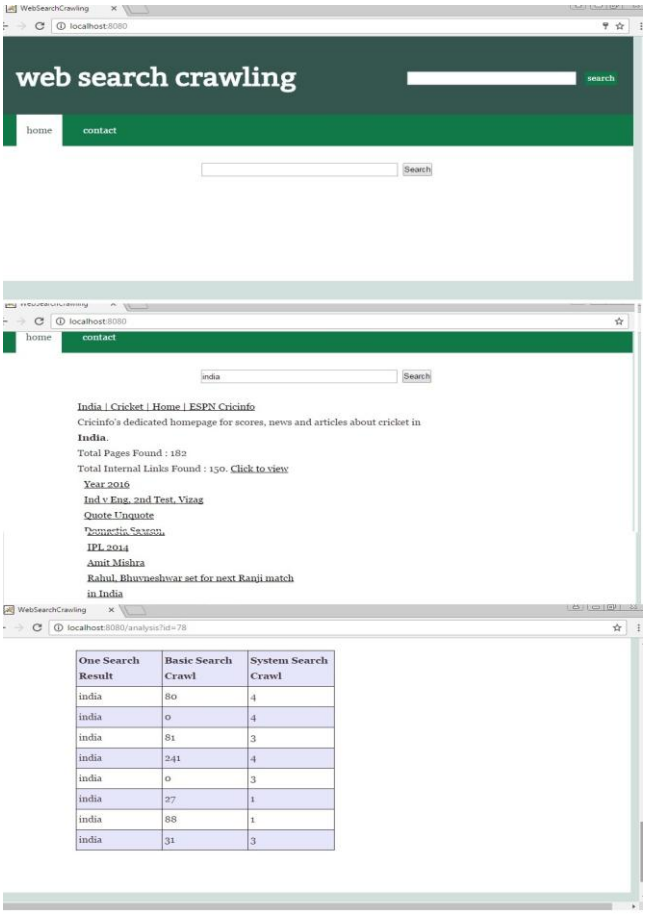


Fig.2 Smart crawler results

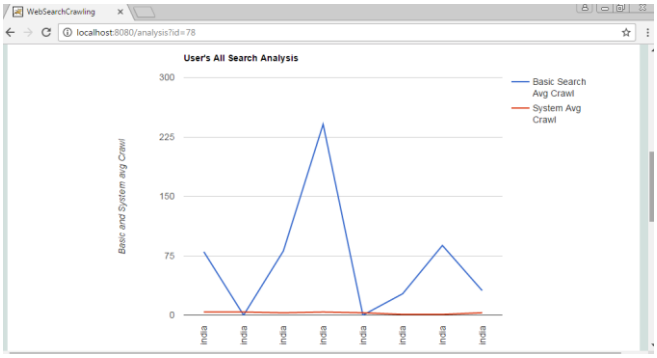


Fig 3: Comparison of basic crawler and smart crawler

## CONCLUSION

An effective harvesting framework for deep-web interfaces, namely Smart-Crawler is proposed. It has been shown that above approach achieves each wide coverage for deep web interfaces and maintains highly efficient pages harvesting. Smart Crawler is a focused crawler consisting of 2 stages: efficient web site locating and then balanced in-site exploring. This Smart Crawler performs site-based locating by reversely searching the known deep web sites for center pages, which can effectively find many data sources for sparse domains. By ranking deep web sites and by focusing the crawling on a topic, SmartCrawler achieves more accurate results. Our experimental results display how two stage smart crawlers achieves higher harvest rates than other basic crawlers.

## REFERENCES

[1] Feng Zhao, Jingyu Zhou, Chang Nie, Heqing Huang, Hai Jin "SmartCrawler: A Two Stage Crawler For Efficiently harvesting Deep-Web interfaces" IEEE Transactions on Services Computing Volume:99 PP Year: 2015

[2] Olston and M. Najork , "Web Crawling", Foundations and Trends in Information Retrieval, vol. 4, No. 3 ,pp. 175–246, 2010 R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[3] M. Burner, "Crawling towards Eternity: Building an Archive of the World Wide Web," Web Techniques Magazine, vol. 2, pp. 37-40, 1997.

[4] Allan Heydon and Marc Najork. Mercator: A scalable, extensible web crawler. World Wide Web Conference, 2(4):219–229, April 1999.

[5] 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, pages 106–113, Hong Kong, May 2001. Elsevier Science.

[6] Roger E. Bohn and James E. Short. How much information? 2009 report on american consumers. Technical report, University of California, San Diego, 2009.

[7] Martin Hilbert. How much information is there in the "information society"? Significance, 9(4):8–12, 2012.

[8] Michael K. Bergman. White paper: The deep web: Surfacing hidden value. Journal of electronic publishing, 7(1), 2001

[9] Yeye He, Dong Xin, Venkatesh Ganti, Sriram Rajaraman, and Nirav Shah. Crawling deep web entity pages. In Proceedings of the sixth ACM international conference on Web search and data mining, pages 355–364. ACM, 2013.

[10] Infomine.UC Riverside library. http://lib-www.ucr.edu/,2014.

[11] Clusty's searchable database directory. http://www.clusty.com/, 2009.

[12] Booksinprint. Books in print and global books in print access. http://booksinprint.com/, 2015.

[13] Kevin Chen-Chuan Chang, Bin He, and Zhen Zhang. Toward large scale integration: Building a metaquerier over databases on the web. In CIDR, pages 44–55, 2005.

[14] Denis Shestakov. Databases on the web: national web domain survey. In Proceedings of the 15th Symposium on International Database Engineering & Applications, pages 179–184. ACM, 2011.

[15] Denis Shestakov and Tapio Salakoski. Host-ip clustering technique for deep web characterization. In Proceedings of the 12th International Asia-Pacific Web Conference (APWEB), pages 378–380. IEEE, 2010.

[16] Denis Shestakov and Tapio Salakoski. On estimating the scale of national deep web. In Database and Expert Systems Applications, pages 780–789. Springer, 2007

[17] Shestakov Denis. On building a search interface discovery system. In Proceedings of the 2nd international conference on Resource discovery, pages 81–93, Lyon France, 2010.Springer.

[18] Luciano Barbosa and Juliana Freire. Searching for hiddenweb databases. In WebDB, pages 1–6, 2005.