A Survey on SmartCrawler: A Deep-Web Harvesting Approach

Radhika Bairagade, Nirmala Singh, Nikita Afre, Durga Bhamare

***Abstract*— The number of web pages available in the Internet is growing tremendously day to day. In this case searching relevant information in the Internet is hard task. A lot of this information is hidden behind query forms that interface to unexplored databases containing high quality structured data. Traditional search engines cannot access and index this hidden part of the Web, retrieving this hidden information is challenging task.**

Therefore, we propose a two-stage framework, namely SmartCrawler, for effectively harvesting deep web interfaces. In the first stage that is site locating, centre pages are searched with the help of search engines which in turn avoid visiting a large number of pages. To achieve more precise results for a focused crawl, SmartCrawler ranks websites to prioritize highly relevant ones for a given topic. In the second stage, adaptive link-ranking achieves fast in-site searching by excavating most relevant links.

To eliminate bias on visiting some highly related links in hidden web directories, we design a link tree data structure to acquire wider coverage for a website. The experimental results on a set of representative domains show the agility and accuracy of proposed crawler framework, which efficiently retrieves deep-web interfaces from large-scale sites and obtains higher harvest rates than other crawlers.

***Index Terms*— Deep web, Crawler, Adaptive learning, Form Classifier, Ranker**

Manuscript received September, 2015.

***Radhika Bairagade**, Department of Computer Engineering, Savitribai Phule Pune University/ Late. G. N. Sapkal College of Engineering., Nasik, India, 9623237626.*

***Nirmala Singh**, Department of Computer Engineering, Savitribai Phule Pune University/ Late. G. N. Sapkal College of Engineering., Nasik, India, 7507238126.*

***Nikita Afre**, Department of Computer Engineering, Savitribai Phule Pune University/ Late. G. N. Sapkal College of Engineering., Nasik, India, 9226818405.*

***Durga Bhamare**, Department of Computer Engineering, Savitribai Phule Pune University/ Late. G. N. Sapkal College of Engineering., Nasik, India, 8552072801.*

I. INTRODUCTION

Basically, meaning of crawler is crawls around the ground. In web crawling, the crawler crawls around the web-pages, gathers and categorizes information on the World Wide Web. The crawler contains of three parts: First is the spider, also called as crawler. The spider visits the pages, fetches the information and then follows the links in other pages within a site. The spider returns to crawled site over regular interval of time. The information found in the first stage will be given to the second stage, the index. It is also well-known as catalog. The index is like a database, containing every copy of web-page that crawler finds. If a web-page changes then the copy is updated with new information in the database. Third part is software. This is a program that sifts millions of web pages recorded in the index to find matches to search and level them in order of what it believes as most relevant.

Deep web[12],[14] also called as dark web or invisible web. Deep web[15] are the contents on the web which is not indexed in a search engine. It is a collection of websites that are publicly available but hide the IP addresses of a server that run on them. Thus they can be visited by the user, but it is difficult to find out who are behind those sites. Deep web is something you cannot locate with a single search.

It is difficult task to locate deep web interfaces, because they are not recorded by any search engines. They are usually rarely distributed and keep constantly changing. To deal with above problem, previous work has proposed two types of crawlers which are generic crawlers and focused crawlers[8],[9],[10],[11],[13]. Generic crawler fetches all the searchable forms and do not focus on a specific topic whereas Focused crawlers are the crawler which focuses on a specific topic. Form-focused crawler (FFC)[5] and Adaptive crawler for hidden web entries (ACHE)[6] aims to efficiently and automatically detect other forms in the same domain. The main components of FFC are link, page, form classifiers and frontier manager for focused crawling of web-forms. ACHE extends the focused strategy of FFC with additional components as form filtering and adaptive link learner. The link classifiers play a pivotal role for achieving higher crawling efficiency than the best-first crawler [7]. The accuracy of focused crawlers is low in terms of retrieving relevant forms. For instance, an experiment conducted for database domains, it has been shown that the curacy of Form-Focused Crawler is around 16 percent [5],[6]. Thus it is essential to develop smart crawler that are able to quickly discover relevant contents from the deep web as much as possible.

A framework for efficiently harvesting deep web named SmartCrawler is designed in this paper. SmartCrawler performs an advanced level of data analysis and data extracted from the web. The SmartCrawler is divided into two stages:
Site locating and in-site exploring. In the first stage, SmartCrawler performs site-based searching for centre pages with the help of search engines, avoiding visiting a large number of pages. To achieve more detailed results for a focused crawl, SmartCrawler ranks websites to prioritized
highly relevant once for a given topic. In the second stage, SmartCrawler achieves fast in-site searching to excavate most relevant links with an adaptive link-ranking.

Site locating technique uses reverse searching technique and incremental two-level site ranking technique for unearthing relevant sites and to achieve more data sources. During the in-site exploring stage, a link tree is designed for balanced link prioritizing, eliminating bias toward web-pages in popular directories.

Adaptive learning algorithm will performs online feature selection and automatically construct link rankers. In the site locating stage, highly relevant sites are prioritized and then crawling is focused on a given topic using the contents of the root page of sites and achieving more accurate results. During the in-site exploring stage, related links are prioritized for fast in-site searching.

II. Related Work

There are many literature in the area of web crawlers. In late 1994, The RBSE (Repository Based Software Engineering project first launch the Web Crawler based on two programs: first was “spider”, it maintain a queue in a relational database, and second was “mite”, it is a modified www ASCII browser that download the pages from web[1]. Then the second WebCrawler was publicly available full-text index of a subset of the web which was based on lib-WWW to download pages, and other program to parse and order URLs for breadth first exploration of web graph.

But this first generation have some of the issues in web crawling design. However design of first crawler did not focus on scalability.

Later so many web crawler are made available. Some of them are: Lycos Infoseek, Exite, AltaVista and HotBot all these crawler are used to index ten millions of pages.

A. Internet archive Crawler

In 1997, Mike Burner designed the Internet Archive Crawler[2] was the first paper that focused on the challenges caused by the scale of web. It uses multiple machine to crawl the web and it crawl on 100 million URLs[1]. Each crawler process read a list of seed URLs for its assigned sites from disk into per-site queue, and then it uses asynchronous I/O instructions to fetch pages from these queues in parallel. It has also deal with the problem of changing DNS records, so it keeps the historical archive of hostname to IP mapping.

B. Google Crawler

Later in 1998, The original Google crawling system consist of a five crawling components which was running in various process and download the pages[2].

Each crawler process used asynchronous I/O instructions to fetch the data from up to 300 web servers in parallel. Then all the crawlers transmit downloaded pages to a single Store Server process that compressed the page and store them on disk[1]. Google Crawler was based on C++ and Python. This crawler was integrated with the indexing process(text parsing was done for full-text indexing and also for URL extraction).

C. Mercator Web Crawler

Heydon and Najork present a web crawler which was highly scalable and easily extensible [3][1]. It was written in Java. The first version was non-distributed and later the distributed version was made available which split up the URL space over the crawlers according to host name and avoid the potential bottleneck of a centralized URL server.

D. WebFountain crawler

In 2001, another distributed and modular crawler represented by IBM[4][1]. It has three major component, Multi threaded crawling processes, duplicate content and central controlled process responsible for assigning work. It was written in C++ and used MPI to facilitate the communication between the various process. It was deployed on a cluster of 48 crawling machine.

E. IRLbot Web crawler

Recently, Yan et al. describe IRLbot, which is single process web crawler [1]. It is able to scale to extremely large web collection without performance degradation. It crawl over two month and downloads the 6.4 billion web pages.

III. Proposed Work

In this paper SmartCrawler contain a novel two-stage framework to address the problem of searching for hidden-web resources.

But to improve accuracy of form classifier, pre-query and post-query approaches for classifying deep-web forms are combined. Additionally, the links in these pages are extracted into Candidate Frontier. To prioritize links in Candidate Frontier, SmartCrawler ranks them with Link Ranker. When the crawler discovers a new site, the site’s URL 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.

IV. System Architecture

For efficiently harvesting deep web data sources, SmartCrawler is designed with a two-stage architecture, site locating and in-site exploring as shown in figure 1. In the first stage site locating finds the most relevant site for a given topic, and in the second stage in-site exploring stage uncovers searchable forms from the site.

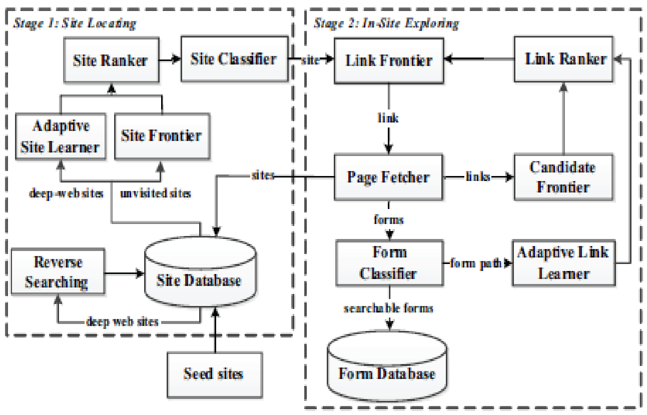

Fig . The two stage architecture of SmartCrawler

Initially, the site locating stage starts with a seed set of sites in a site database. Seeds sites are candidate sites given to SmartCrawler to start crawling, which begins by following URLs from chosen seed sites to search other pages and other domains. When the number of unvisited URLs in the database is less than a threshold value during the crawling process then SmartCrawler performs "reverse searching" of known deep web sites for center pages (highly ranked pages that have many links to other domains) and provides these pages to the site database. The site frontier will fetch web-page URLs from the site database. The un-visited sites are given to site frontier and are prioritized by site ranker, whereas the visited sites are added to fetched site list. Site Ranker assigns a score for each unvisited site that corresponds to its relevance to the already discovered deep web interfaces. The Site Ranker is improved during crawling by an Adaptive Site Learner. It will adaptively learns from features of deep-web sites (web sites containing one or more searchable forms) found. To achieve more accurate results for a focused crawl, Site Classifier categorizes URLs into relevant or irrelevant for a given topic according to the homepage content.
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 fetched. Then embedded forms are classified by Form Classifier to find searchable forms. To improve accuracy of form classifier, pre-query and post-query approaches for classifying deep-web forms are combined. Additionally, the links in these pages are extracted into Candidate Frontier. To prioritize links in Candidate Frontier, SmartCrawler ranks them with Link Ranker. When the crawler discovers a new site, the site's URL 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.

V. Conclusion

An effective harvesting framework for deep-web interfaces, namely Smart-Crawler is proposed. It has been shown that above approach achieves both wide scope for deep web interfaces and maintains highly efficient crawling. SmartCrawler is a focused crawler consists of two stages: site locating and balanced in-site exploring. SmartCrawler performs site-based locating by reversely searching the known deep web sites for center pages, which can efficiently find many data sources for sparse domains. SmartCrawler achieves more accurate results by ranking collected sites and focusing the crawling on a given topic. The in-site exploring stage uses adaptive link-ranking to search within a site and design a link tree for eliminating bias toward certain directories of a website for wider coverage of web directories. The experimental results on a representative set of domains shows the effectiveness of the proposed two-stage crawler, which in turn achieves higher harvest rates than other crawlers.

Acknowledgment

We would like to take this opportunity to express my profound gratitude and deep regard to Prof. J. V. Shinde, for her exemplary guidance, valuable feedback and constant encouragement throughout writing this paper. Working under her was an extremely knowledgeable experience for us.We would also like to give our sincere gratitude to all the friends and colleagues who filled in the survey, without which this survey would be incomplete.

REFERENCES

[1] Olston and M. Najork , "Web Crawling", Foundations and Trends in Information Retrieval, vol. 4, No. 3 ,pp. 175–246, 2010

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

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

[4] 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.

[5] Luciano Barbosa and Juliana Freire. Searching for hidden-web databases.In *WebDB*, pages 1–6, 2005.

[6] Luciano Barbosa and Juliana Freire. An adaptive crawler for locating hidden-web entry points. In *Proceedings of*

the 16th international conference on World Wide Web, pages 441–450. ACM, 2007.

[7] Soumen Chakrabarti, Martin Van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 31(11):1623–1640, 1999.

[8] 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.

[9] 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.

[10] 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.

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

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

[13] 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.

[14]Bright planet’s searchable database directory. http://www.completeplanet.com/, 2013.

[15] Y. Wang, T. Peng, W. Zhu, “Schema extraction of Deep Web Query Interface” , *IEEE Transaction On Web Information Systems and Mining,* WISM International Conference 2009.