ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

Smart Crawler: A Deep Web Harvesting Approach

Radhika Bairagade[1], Nikita Afre[2], Nirmala singh[3], Durga Bhamare[4]

B.E Student, Dept. of Computer Engineering, Savitribai Phule Pune University, Nasik, India[1,2,3,4]

ABSTRACT: The internet is a huge collection of billions of web pages and a vast amount of web pages lie in the deep web. As growth of deep web is tremendous so there has been increased interest in techniques that will help efficiently to discover deep-web interfaces. However, due to massive amount of web resources and the dynamic nature of web pages, achieving wide coverage and high efficiency is challenging. We tend to propose a two-stage framework, specifically *SmartCrawler*, for efficient harvesting deep web interfaces. In the first stage, our crawler performs site-based searching for sorting center pages via search engines which avoids accessing an oversized variety of web pages. To generate results more precisely *SmartCrawler* ranks websites to prioritize highly relevant ones for a given topic. Within the second stage, *SmartCrawler* achieves quick in-site searching by excavating most relevant links with an adaptive link-ranking. To exclude bias on visiting some highly relevant links in hidden web directories, a link tree data structure is designed. The experimental results on a set of representative queries show the agility and accuracy of our proposed crawler framework, which neatly retrieves deep-web interfaces from large-scale sites.

KEYWORDS: Focused crawler, ranking, adaptive learning

I. Introduction

The Web is a vast collection of web pages. The web pages contain large bytes of information or data arranged in N number of servers using Hyper Text Markup Language. In contrast to traditional collections such as libraries, the Web has no centrally systematize content structure. This data can be downloaded using web crawler. A Web crawler is a computer program that searches the World Wide Web in a defined manner. Other terms for Web crawlers are ants, automatic indexers, bots, and Web spider. Web crawlers are mainly used to create a copy of all the visited pages. These pages are later processed by a search engine that which does the indexing of the downloaded pages to provide fast searches.

A typical web crawler starts by parsing a defined web page: noticing any hypertext links on that page that point to other web pages. The Crawler then scans those pages for new links, and so on, recursively. A crawler is a software or script or automated program which exists on a single machine. The crawler simply sends HTTP requests for documents to other machines on the Internet, similarly as a web browser do when the user clicks on links. All the crawler really does is to automate the process of following links.

Deep-web crawl is a problem of surfacing rich information behind the web search interface of various sites across the Web. It was evaluated by various accounts that the deep-web has as much as an order of magnitude more content than that of the surface web [1] [2]. While crawling the deep-web can be immensely useful for a variety of tasks including web indexing [3] and data integration [2], crawling the deep-web content is known to be hard. The difficulty in surfacing the deep-web has inspired a long and fruitful line of research[4][5][6][13][14][15].

II. Related work

In the History of web-crawling, a web crawler is basically a software that starts from a set of seed site (URLs), and downloads all the web pages associated with the URLs. The traditional definition of a web crawler assumes that all the content of a web application is reach through these URLs.

In late 1993, Web crawlers were written and this gave a birth to four web crawlers: Jump Station, RBSE spider, World Wide Web Wanderer, World Wide Web Worm [7]. These four Crawlers mainly collected information

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

and statistic about the web using a set of seed site (URLs). In addition to collecting data and stats about the state of the web, next year in 1994, the two new web crawlers get appeared: Web Crawler and MOMspider. These two web crawlers introduced concepts of politeness and black-lists to traditional web crawlers. From World Wide Web Worm to Web Crawler, the number of indexed pages increased from 110,000 to 2 million. Later in the coming years a few commercial web crawlers are made available: Infoseek, Excite, Lycos, HotBot and AltaVista. In 1998, Brin and Page [8] tried to address the issue of scalability by introducing a large scale web crawler called Google. Google addressed the problem of scalability in several ways: It leveraged many low level optimizations to reduce disk access time through techniques such as compression and indexing. Architecturally, Google used a master-slave architecture with a master server dispatching URLs to a set of slave nodes. The slave nodes retrieve the assigned pages by downloading them from the web. The first implementation of Google reached 100 page downloads per second.

Then In 1999,the issue of scalability was further addressed by Allan Heydon and Marc Najork in a tool called Mercator [9] .But the Mercator attempted to address the problem of extendability of web crawlers. To overcome from this problems the second version of Mercator crawled 891 million pages. Later, Mercator got integrated into AltaVista in 2001. Later, IBM introduced WebFountain [10] in 2001.WebFountain was a fully distributed web crawler and its objective was to index the web and also to create a local copy of it. In a simulation, WebFountain managed to scale with a growing web. This simulated web originally had 500 million pages and it grew to twice its size every 400 days. In 2002, Using this data structure to handle the URL Seen test, Polybot managed to scan 120 million pages. In addition to Polybot and UbiCrawler, in 2002 Tang et al. [11] introduced pSearch. pSearch uses two algorithms called P2P Vector Space Model and P2P Latent Semantic Indexing to crawl the web on a P2P network.

Recently, an extremely scalable web crawler called IRLbot ran for 41.2 days on a quad-CPU AMD Opteron 2.6 GHz server and it crawled over 6.38 billion web pages [12].

III. PROPOSED ALGORITHM

A. *System Design:*

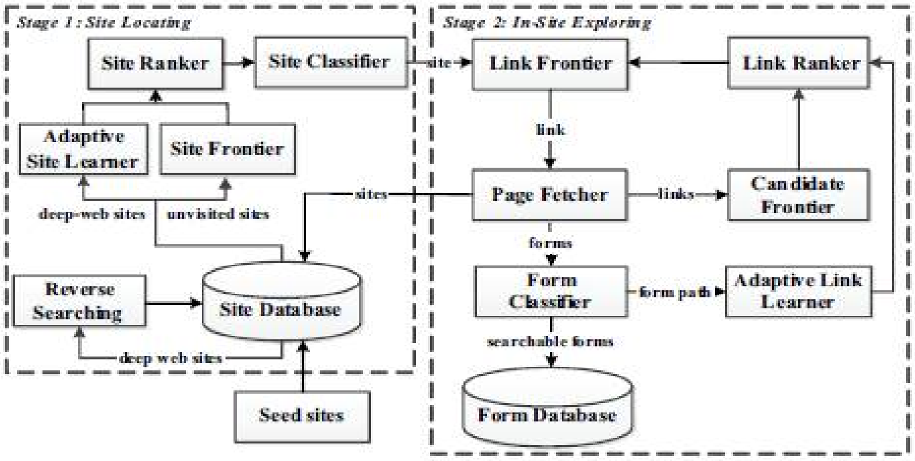

Figure 1: The two-stage architecture of *SmartCrawler*

Exploring deep net knowledge sources includes a 2 stage architecture, web site locating and in-site exploring, as shown in Figure1. At the First stage, Crawler finds the most relevant web site for a given topic, then the second phase will be in-site exploring stage which discovers searchable content from the site.

1. Locating the site

Locating the site consist of finding the relevant site for a given subject, this stage consist of following parts:

- Web Site Collecting
- Site Classification
- Site Ranking

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

Site Classifier

After ranking Site Classifier classifies the site as topic relevant or irrelevant for a focused crawl, which is similar to page classifiers in FFC [5] and ACHE [6]. If a site is classified as topic relevant, a site crawling process is launched. Otherwise, the site is neglected and a new site is picked from the frontier.In *SmartCrawler*, we determine the topical relevancy of a site based on the contents of its homepage. When a new site comes, the homepage content of the site is extracted and parsed by discarding stop words and stemming.

Site Ranking

SmartCrawler ranks site URLs to prioritize potential deep sites of a given topic. For ranking two features, *site similarity* and *site frequency* are considered. Site similarity measures the topic similarity within a new site and known deep web sites. Site frequency indicates the popularity and authority of particular site and a high frequency site is potentially more important. Because seed sites are carefully selected, relatively high scores are given to them.

Given the homepage URL of a new site $s = \{ Us, As, Ts, \}$ the site similarity to known deep web sites *FSS*, can be describe as follows:

$$ST(s) = Sim(U,Us) + sim(A,As) + sim(T, Ts); \qquad (1)$$

where function *Sim* computes the similarity of the related feature between *s* and known deep web sites. The function *Sim*(.) is computed as the cosine similarity within two vectors *V*1 and *V*2:

$$Sim(V1, V2) = \frac{V1.V2}{|V1|\times|V2|} \qquad (2)$$

Old crawler's finds only newly found links to sites, but Smartcrawler minimize the number of visited URLS, also maximizes number of deep searches. The issue with the other system is that even most popular site does not return number of deep web searches.

2. In-Site Exploring

Once a site is considered as topic relevant, in-site exploring is performed to find searchable forms. The goals are to rapidly harvest searchable forms and to cover web directories of the site as much as possible. To achieve these goals, in-site exploring follows two crawling strategies for high efficiency and coverage.
- Link Ranker
- Form classifier

Link Ranker

For prioritizing links of a site, the link similarity is computed similar to the site similarity as explained above. The difference includes:
1) Link prioritizing is based on the feature space of links with searchable forms (*FSL*);
2) For URL feature *U*, only path part is considered since all links have the same domain;
3) The frequency of links is not taken into consideration.

Given a new link $l = \{Pl, Al, Tl\}$, the link similarity to the feature space of known links with searchable forms *FSL* is defined as:

$$LT(l) = Sim(P, Pl) + sim(A,Al) + sim(T, Tl); \qquad (3)$$

where function *Sim*(.) (Equation 2) scores the similarity of the related feature between *l* and the known in-site links with forms.

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

Form Classifier

Classifying forms includes keeping form focused crawling, which filters out non-searchable and irrelevant forms. *SmartCrawler* adopts the HIFI strategy to filterate relevant searchable forms with a composition of simple classifiers [16]. HIFI consists of two classifiers, a searchable form classifier (SFC) and a domain-specific form classifier (DSFC). SFC is a domain-independent classifier to filter out non-searchable forms by using the constructed feature of forms. DSFC determines whether a form is topic relevant or not based on the text feature of the form, that consists of domain-related terms. The technique of partitioning the feature space allows selection of more effective learning algorithms for each feature subset. In our implementation, SFC uses decision tree based C4.5 algorithm [5], [17] and DSFC employs SVM [18].

B. *Proposed Algorithm:*

Finding out-of-site links from visited webpages may not be enough for the Site Frontier. To address these problem, we propose two crawling strategies, *reverse searching* and *increamental site prioritizing* to find more sites.

Reverse Searching

We randomly pick a known seed site and use general search engine's technique to find centre pages and other relevant sites. In our system, the result page within a search engine is first parsed to extract links. Then these pages are downloaded and sorted out to determine whether the links are relevant or not using the following rules:
– If the page contains related searchable forms, it is relevant.
– If the number of fetched deep web sites in the page is larger than a user defined threshold, the page is relevant.

Algorithm:
Reverse searching for more sites.
input for the system: seed sites and harvested deep websites
output from the system: relevant sites

```
1  while number of candidate sites less than a threshold do
2      // pick a deep website
3      site = getDeepWebSite(siteDatabase, seedSites)
4      resultpage = reverseSearch(site)
5      links = extractLinks(resultpage)
6      for each link in links do
7          page = downloadPage(link)
8          relevant = classify(page)
9          if relevant then
10             relevantSites =
               extractUnvisitedSite(page)
11             Output relevantSites
12         end
13     end
14 endIncremental site prioritizing:
```

Incremental site prioritizing

An incremental site prioritizing strategy is proposed, to make crawling process presumable and succeed broad coverage on websites. First, the prior knowledge is used for initializing Site Ranker and Link Ranker. Then, unvisited sites are assigned to Site Frontier and are prioritized by Site Ranker. Visited sites are added to fetched site list.

Algorithm:

```
Incremental Site Prioritizing.
input for the system: siteFrontier
output from the system: searchable forms and out-of-site links
HQueue=SiteFrontier.CreateQueue(HighPriority)
LQueue=SiteFrontier.CreateQueue(LowPriority)
while siteFrontier is not null do
|       if HQueue is empty then
|       |       HQueue.addAll(LQueue)
|       |       LQueue.clear()
|       end
|       site = HQueue.poll()
|       relevant = classifySite(site)
|       if relevant then
|       |       performInSiteExploring(site)
|       |       Outputforms and OutOfSiteLinks
|       |       siteRanker.rank(OutOfSiteLinks)
|       |       if forms is not empty then
|       |       |       HQueue.add (OutOfSiteLinks)
|       |       end
|       |       else
|       |       |       LQueue.add(OutOfSiteLinks)
|       |       end
|       end
end
```

IV. Simulation Results

We have performed an comprehensive performance evaluation of our crawling framework over real web data in a fix pattern. Our goals include:
- Evaluating the efficiency of *SmartCrawler* in obtaining relevant deep websites and searchable forms,
-Assessing the performance of adaptive learning,and
-Aanalyzing the effectiveness of site collecting.

Experimental Setup

We have implemented *SmartCrawler* in .NET framework and evaluated our approach over a fix pattern(Top 10 pattern). To evaluate the performance of our crawling framework, we compare *SmartCrawler* to the SCDI(site-based crawler for deep web interfaces), ACHE and SmartCrawler.

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

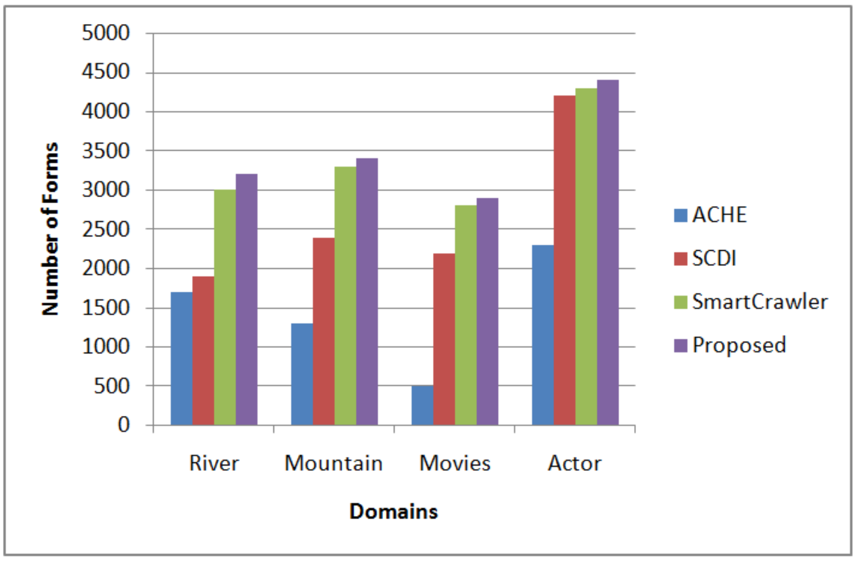

Figure 2: The numbers of relevant deep websites harvested by ACHE, SCDI, *SmartCrawler* and Proposed System

To evaluate the performance of our crawling framework, we compare our proposed crawler to the SCDI (site-based crawler for deep web interfaces) and ACHE and SmartCrawler[6].

- **ACHE.** It is an adaptive crawler for harvesting hidden deep-web entries with offline-online learning to link classifiers. We adapt the similar stopping criteria as *SmartCrawler*, i.e., the maximum visiting pages and a predefined number of forms for each site.

- **SCDI.** We designed an experimental system similar to *SmartCrawler*, named SCDI, which shares the same stopping criteria with *SmartCrawler*. Different from *SmartCrawler*, SCDI follows the out-of-site links of relevant sites by site classifier without employing incremental site prioritizing strategy. It also does not employ reverse searching for collecting sites and use the adaptive link prioritizing strategy for sites and links.

- **SmartCrawler.** *SmartCrawler* is our proposed crawler for harvesting deep web interfaces. Similar to ACHE, *SmartCrawler* uses an offline-online learning strategy, with the difference that *Smart-Crawler* leverages learning results for site ranking and link ranking. During in-site searching, more stop criteria are specified to avoid unproductive crawling in *SmartCrawler*.

- **Proposed Crawler.** Our crawler is implemented on fixed pattern(top #). SmartCrawler can find out relevant sites related to a specified set of domains while our crawler is implemented dynamically. As experimental result have shown that our crawler gives slightly better performance than SmartCrawler while harvesting deep web sites.

Domains	ACHE	SCDI	SmartCrawler	Proposed
River	1700	1900	3000	3200
Mountain	1300	2400	3300	3400
Movies	500	2200	2800	2900
Actor	2300	4200	4300	4400

Table 1: Table of comparision between ACHE, SCDI, SmartCrawler and Proposed System

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

We compared the number of forms harvested by *SmartCrawler*, ACHE and SCDI under common sites.The results of the number of common sites and searchable forms found in these common sites are showed in Figure 2. Since SCDI, ACHE and *SmartCrawler* shares the same 100 seed sites, the forms fetched in the same sites can reflect the effectiveness of proposed crawling strategies. Figure 2 shows *SmartCrawler* can retrieve more searchable forms than ACHE and SCDI when searching the same set of sites in almost all domains.

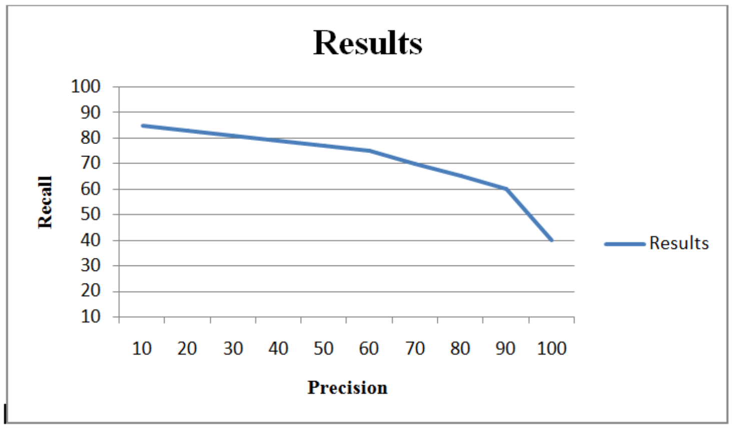

Figure 3: Accuracy of Smart Crawler

The 10-fold cross validation is used to evaluate the accuracy of site classifier and C4.5 algorithm is used for filtering non-searchable forms. After filtering out non-searchable forms, DSFC is used to judge whether the form is topic relevant or not. The average accuracy of *SmartCrawler(proposed system)* is 93.17%, while ACHE is 90.24%. This is because *SmartCrawler* avoids crawling unproductive forms.

V. Conclusion and Future Work

The simulation results showed that the proposed algorithm performs better for effective harvesting framework for deep-web interfaces, namely *SmartCrawler*. A deep web harvesting 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 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.

References

[1] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang. Accessing the deep web. Commun. ACM, 50, 2007.
[2] J. Madhavan, S. R. Jeffery, S. Cohen, X. luna Dong, D. Ko, C. Yu, and A. Halevy. Web-scale data integration: You can only afford to pay as you go. In Proceedings of CIDR, 2007.
[3] J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Halevy. Google's deep web crawl. In Proceedings of VLDB, 2008.
[4] L. Barbosa and J. Freire. Siphoning hidden-web data through keyword-based interfaces. In Proceedings of SBBD, 2004.
[5] L. Barbosa and J. Freire. Searching for hidden web databases. In Proceedings of WebDB, 2005.
[6] L. Barbosa and J. Freire. An adaptive crawler for locating hidden-web entry points. In Proceedings of WWW, 2007
[7] O. A. McBryan, "Genvl and wwww: Tools for taming the web," in In Proceedings of the First International World Wide Web Conference, 1994, pp. 79–90.
[8] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," in Proceedings of the seventh international conference on World Wide Web 7, ser. WWW7. Amsterdam, The Netherlands, The Netherlands: Elsevier Science Publishers B. V., 1998, pp. 107–117.
[9] A. Heydon and M. Najork, "Mercator: A scalable, extensible web crawler," World Wide Web, vol. 2, pp. 219–229, 1999.
[10] J. Edwards, K. McCurley, and J. Tomlin, "An adaptive model for optimizing performance of an incremental web crawler," 2001.
[11] J. Li, B. Loo, J. Hellerstein, M. Kaashoek, D. Karger, and R. Morris, "On the feasibility of peer-to-peer web indexing and search," Peer-to-Peer Systems II, pp. 207–215, 2003
[12] H. tsang Lee, D. Leonard, X. Wang, and D. Loguinov, "Irlbot: Scaling to 6 billion pages and beyond," 2008.

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

[13] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang. Accessing the deep web. Commun. ACM, 50, 2007.
[14] J. Madhavan, S. R. Jeffery, S. Cohen, X. luna Dong, D. Ko, C. Yu, and A. Halevy. Web-scale data integration: You can only afford to pay as you go. In Proceedings of CIDR, 2007.
[15] J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Halevy. Google's deep web crawl. In Proceedings of VLDB, 2008.
[16] Luciano Barbosa and Juliana Freire. Combining classifiers to identify online databases. In *Proceedings of the 16th international conference on World Wide Web*, pages 431–440. ACM, 2007.
[17] Jared Cope, Nick Craswell, and David Hawking. Automated discovery of search interfaces on the web. In *Proceedings of the 14th Australasian database conference-Volume 17*, pages 181–189. Australian Computer Society, Inc., 2003.
[18] Dumais Susan and Chen Hao. Hierarchical classification of Web content. In *Proceedings of the 23rd Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 256–263, Athens Greece, 2000.

BIOGRAPHY

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

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

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

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