ISSN 2319-2720
Volume 7, No.2, April – June 2018
International Journal of Computing, Communications and Networking
Available Online at http://www.warse.org/ijccn/static/pdf/file/ijccn30722018.pdf
https://doi.org/10.30534/ijccn/2018/30722018

IMPLEMENTATION OF SMARTCRAWLER FOR DEEP-WEB SITES

Sagar B[1], Shruthi P[2], Sunitha M[3], Shushma M[4], Sushmita T[5]
[1]Associate Professor, India, sagar11105@gmail.com
[2]Student, EWIT India, shruthipalmari96@gmail.com
[3] Student, EWIT India, sunimeco311@gmail.com
[4] Student, EWIT India, shushmagowda22@gmail.com
[5] Student, EWIT India, sushmitatadkal@gmail.com

ABSTRACT

Deep web is termed as data present on web but inaccessible to search engine. Due to the large volume of web resources and the dynamic nature of deep web, achieving wide coverage and high efficiency is a challenging issue. Smart crawler for hidden web interfaces consist of mainly two stages, first is site locating another is in-site exploring. Site locating starts from seed sites and obtains relevant websites through reverse searching and obtains relevant sites through feature space of URL, anchor and text around URL. Second stage takes input from site locating and goes to find relevant link from those sites. The adaptive link learner is used to find out relevant links with help of link priority and link rank.. To eliminate bias on visiting some highly relevant links in hidden web directories, we design a link tree data structure to achieve wider coverage for a website.

Key words: Adaptive learning, deep web, feature selection, ranking, two-stage crawler.

1. INTRODUCTION

Internet is collection of huge number of web pages present in the form of HTML. To retrieve the necessary information present on web is important issue as the size of collection is very large. The search engine plays important part in our life. Search engine help user to retrieve information. Web search engine a software system that design to search information on www. The results called search results are termed as search engine result pages which consists of mixture of web pages, images and other types of files. A web crawler is system for that move around internet for storing and collecting data into database for further arrangement and analysis of data.

The Deep Web refers to content hidden behind HTML forms. Keeping in mind the end goal to get to such content, a client needs to perform a structure accommodation with legitimate information values. The Deep Web has been recognized as a significant gap in the scope of web indexes in light of the fact that web crawlers utilized via internet searchers depend on hyperlinks to find new web pages and normally do not have the capacity to perform such form submissions. Different records have speculated that the Deep Web has a request of size more information than the presently searchable Internet.Morever, the Deep Web has been a long-standing test for the database group on the grounds that it represents a huge portion of the structured information on the Web.

More recent studies assessed that 1.9 zettabytes were come to and 0.3 zettabytes were devoured worldwide in 2007. A noteworthy portion of this huge amount of information is evaluated to be put away as organized or social information in web databases — deep web makes up around 96% of all the content on the Internet, which is 500-550 times bigger than the surface web. These information contain a vast amount of important data what's more, entities, for example, Infomine, Clusty, BooksInPrint may be keen on building an index of the deep web sources in a given area (for example, book). Since these entities can't get to the exclusive web records of web indexes (e.g., Google and Baidu), there is a requirement for a proficient crawler that has the capacity precisely and rapidly investigate the deep web database.

In this paper, we propose an effective deep web harvesting framework, namely SmartCrawler, for achieving both wide coverage and high efficiency for a focused crawler.

2. RELATED WORK

Olston and Najork systematically present that crawling deep web has three steps: locating deep web content sources, selecting relevant sources and extracting underlying content. Following their statement, we discuss the two steps closely related to our work as below:

Locating deep web content sources. A recent study shows that the harvest rate of deep web is low — only 647,000 distinct web forms were found by sampling 25 million pages from the Google index (about2.5%). Generic crawlers are mainly developed for characterizing deep web and directory construction of deep web resources, that do not limit search on a specific topic, but attempt to fetch all searchable forms. The Database Crawler in the MetaQuerier is designed for automatically discovering query interfaces. Database Crawler first finds root pages by an IP-based sampling, and then performs shallow crawling to crawl pages within a web server starting from a given root page. The IP based sampling ignores the fact that one IP address may have several virtual hosts, thus missing many websites. To overcome the drawback of IP based sampling in the Database Crawler, Denis et al. propose a stratified random sampling of hosts to characterize national deep web, using the Host graph provided by the Russian search engine Yandex. I-Crawler combines pre-query and post-query approaches for classification of searchable forms.

Selecting relevant sources. Existing hidden web directories usually have low coverage for relevant online databases which limits their ability in satisfying data access needs. Focused crawler is developed to visit links to pages of interest and avoid links to off-topic regions. The classifier learns to classify pages as topic-relevant or not and gives priority to links in topic relevant pages. However, a focused best-first crawler harvests only 94 movie search forms after crawling 100,000 movie related pages. An improvement to the best-first crawler is proposed, where instead of following all links in relevant pages, the crawler used an additional classifier, the apprentice, to select the most promising links in a relevant page. The baseline classifier gives its choice as feedback so that the apprentice can learn the features of good links and prioritize links in the frontier. The FFC and ACHE are focused crawlers used for searching Interested deep web interfaces. The FFC contains three classifiers: a page classifier that scores the relevance of retrieved pages with a specific topic, a link classifier that prioritizes the links that may lead to pages with searchable forms, and a form classifier that filters out non-searchable forms.

3. EXISTING SYSTEM

It is challenging to locate the deep web databases, because they are not registered with any search engines, are usually sparsely distributed, and keep constantly changing. To address this problem, previous work has proposed two types of crawlers, generic crawlers and focused crawlers. Generic crawlers fetch all searchable forms and cannot focus on a specific topic. Focused crawlers such as Form-Focused Crawler (FFC) and Adaptive Crawler for Hidden-web Entries (ACHE) can automatically search online databases on a specific topic. FFC is designed with link, page, and form classifiers for focused crawling of web forms, and is extended by ACHE with additional components for form filtering and adaptive link learner.

The link classifiers in these crawlers play a pivotal role in achieving higher crawling efficiency than the best-first crawler. However, these link classifiers are used to predict the distance to the page containing searchable forms, which is difficult to estimate, especially for the delayed benefit links (links eventually lead to pages with forms). As a result, the crawler can be inefficiently led to pages without targeted forms.

Disadvantages:

1. Consuming large amount of data.
2. Time wasting while crawling in the web.
3. The crawler led to pages without targeted forms.
4. Set of retrieved results are very heterogeneous.

4.PROPOSED SYSTEM

We propose a two-stage framework, namely Smart Crawler, for efficient harvesting deep web interfaces.

SmartCrawler is arranged with a two stage building, site page finding and in-site exploring, The essential site page discovering stage finds the most relevant site page for a given subject, and subsequently the second in-site researching stage uncovers searchable structures from the site page.

The figure below shows the architecture of proposed system:

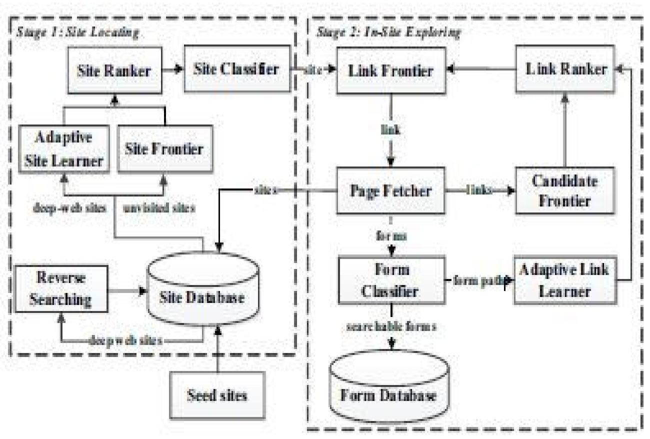

Fig. 1: The two-stage architecture of *SmartCrawler*.

Stage 1: Site locating –

In Site locating stage the smart crawler performs the operation to find out the relevant sites related to the fired query. It has number of steps involved to give the final result of this stage.

1) Seed Sites: It is initial stage of the architecture. Here, seed sites are the candidate sites which are given to start crawling. It begins with following URL of the query and explores other pages and other domains.

2) Reverse searching: Pages with high rank and links to many other pages is called as centre page of site. Some threshold is defined for seed sites, if number of visited sited is less than the threshold then Reverse Searching is performed to know the centre pages of the known deep web sites. Feed these pages back to the site database. The randomly picked site uses general search engine facility to find centre pages and other relevant sites. Smart crawler first extract links on the page then download these pages and analyze these pages to decide whether the links are relevant or not. Following algorithm is used for reverse searching:

Algorithm

Input: seed sites and harvested deep websites.

Output: relevant sites.

```
1 while # of candidate sites less than a threshold do
2 //pick   a deep website
3     site = getDeepWebSite(siteDatabase,seedSites)
4     resultP age = reverseSearch(site)
5     links = extractLinks(resultP age)
6     foreach 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 end.
```

3) Incremental site Prioritizing: Incremental site prioritizing is used to achieve broad coverage on websites. It record the learned pattern of deep sites and form the path for crawling. Basic knowledge is used to initialize both rankers such as site ranker and link ranker. Unvisited sites given to site frontier later prioritize by site ranker and added to the list fetched site. Two queues are used to classify out of site links such high priority queue and low priority queue respectively. High priority queue consist of out of site links which are classified as relevant and judge by form classifier and low priority queue consist of links that are only judged as relevant. Algorithm for Incremental site Prioritizing is given below:

Algorithm:
Input: Site Frontier.
Output: searchable forms and out-of-site links.

```
1 HQueue=SiteFrontier.CreateQueue(HighPriority)
2 LQueue=SiteFrontier.CreateQueue(LowPriority)
3 while siteFrontier is not empty do
4     if HQueue is empty then
5        HQueue.addAll(LQueue)
6         LQueue.clear()
7     end
8     site = HQueue.poll()
9      relevant = classifySite(site)
10    if relevant then
11       performInSiteExploring(site)
12       Output forms and OutOfSiteLinks
13       siteRanker.rank(OutOfSiteLinks)
14          if forms is not empty then
15             HQueue.add (OutOfSiteLinks)
16           end
17          else
18             LQueue.add(OutOfSiteLinks)
19          end
20    end
21 end
```

3) Site Frontier: Site Frontier fetches the homepage URLs from the site database which is further ranked by Site Ranker to prioritize the highly relevant sites. Finding out-of-site links from visited web pages may not be enough for the Site Frontier.

4) Adaptive link learner: Site ranker and link ranker are controlled by Adaptive link learner. The feature space is decided for deep web sites and links known as FSS and FSL respectively. 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. The Link Ranker is adaptively improved by an Adaptive Link Learner, which learns from the URL path leading to relevant forms.

5) Site Ranker: Site ranker is used to rank unvisited site from deep website. There are two parameters that are used for ranking mechanism are Site Similarity and Site Frequency. Site Similarity depends on the topic similarity between known deep site and new site. Site Frequency is occurrence of site in another web site.

6) Site Classifier: The high priority queue is for out-of-site links that are classified as relevant by Site Classifier and are judged by Form Classifier to contain searchable forms. If site is judge as topic relevant then site crawling process is started otherwise new site is picked from site frontier.

Stage 2: In-Site Exploring –

After finding most relevant sites in stage 1 the stage 2 perform the in-site exploration to find searchable forms.

1) Link Frontier: Link frontier takes sites as input which are classified by site classifier. Link frontier mainly works for finding links with in centre pages. Criteria for stop early are given as-

Crawling Strategies: Mainly two crawling strategies are present Stop early and Balance link prioritizing.

Stop Early:
SC1: when reached maximum depth.
SC2: maximum crawling pages in each depth are reached.
SC3:Predefined number of forms are found in each depth.
SC4: No searchable forms till threshold value.

Balance link prioritizing:
Here, link tree is constructed. Root node is the selected site and internal leaf node is each directory present in the website. The simple breadth-first visit of links is not efficient, whose results are in omission of highly relevant links and incomplete directories visit.We solve this problem by prioritizing highly relevant links with link ranking and to build a link tree for a balanced link prioritizing. For example,consider an internal nodes which represent the directory. Servlet: Dynamic request
Books: Catalogs of books
Docs directory: Help information.

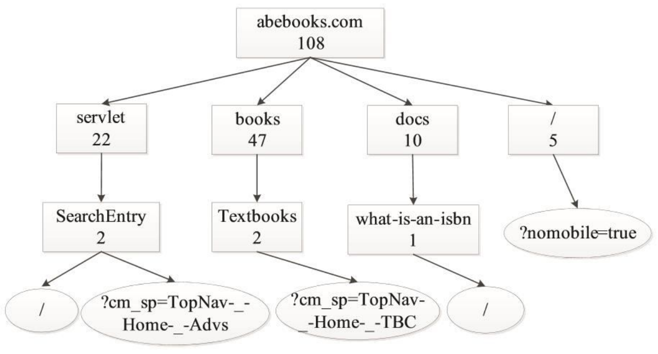

Fig. 2: Part of the link tree extracted from the homepage of http://www.abebooks.com, where ellipses represent leaf nodes

and the number in a rectangle denotes the number of leaf nodes in its decedents.

2) Link Ranker: Link Ranker prioritizes links so that SmartCrawler can quickly discover searchable forms. A high relevance score is given to a link that is most similar to links that directly point to pages with searchable forms.

3) Page Fetcher: Page Fetcher directly fetch out centre page of the web site.

4) Candidate Frontier: The links in web pages are extracted into Candidate Frontier. The working of candidate frontier is similar as site frontier.

5) Form Classifier: Form classifier filters out non-searchable and irrelevant forms. The HIFI strategy is used to filter forms. HIFI consists of two classifier, Searchable form classifier (SFC)and domain specific form classifier(DSFC). SFC is domain independent and it filter out the non searchable forms.

6) Adaptive Link Learner: The Link Ranker is adaptively improved by an Adaptive Link Learner, which learns from the URL path leading to relevant forms.

7)Form Database: Form database contains collection of sites; it collects all data which got input from Form Classifier. Finally the result obtained is the most relevant forms are obtained in deep web interfaces which are the desired result of the proposed system.

5. MODULES

After careful analysis the system has been identified to have the following modules:

a. two-stage crawler.
b. Site Ranker
c. Adaptive learning

a. Two-stage crawler:

It is challenging to locate the deep web databases, because they are not registered with any search engines, are usually sparsely distributed, and keep constantly changing. To address this problem, previous work has proposed two types of crawlers, generic crawlers and focused crawlers. Generic crawlers fetch all searchable forms and cannot focus on a specific topic. Focused crawlers such as Form-Focused Crawler (FFC) can automatically search online databases on a specific topic. FFC is designed with link, page, and form classifiers for focused crawling of web forms, and is extended by with additional components for form filtering and adaptive link learner. The link classifiers in these crawlers play a pivotal role in achieving higher crawling efficiency than the best-first crawler.

However, these link classifiers are used to predict the distance to the page containing searchable forms, which is difficult to estimate, especially for the delayed benefit links (links eventually lead to pages with forms). As a result, the crawler can be inefficiently led to pages without targeted forms.

b. Site Ranker:

When combined with above stop-early policy. We solve this problem by prioritizing highly relevant links with link ranking. However, link ranking may introduce bias for highly relevant links in certain directories. Our solution is to build a link tree for a balanced link prioritizing. Generally each directory usually represents one type of files on web servers and it is advantageous to visit links in different directories. For links that only differ in the query string part, we consider them as the same URL because links are often distributed unevenly in server directories, prioritizing links by the relevance can potentially bias toward some directories. For instance, the links under books might be assigned a high priority, because “book” is an important feature word in the URL. Together with the fact that most links appear in the books directory, it is quite possible that links in other directories will not be chosen due to low relevance score. As a result, the crawler may miss searchable forms in those directories.

c. Adaptive learning:

Adaptive learning algorithm that performs online feature selection and uses these features to automatically construct link rankers. In the site locating stage, high relevant sites are prioritized and the crawling is focused on atopic using the contents of the root page of sites, achieving more accurate results. During the insite exploring stage, relevant links are prioritized for fast in-site searching. We have performed an extensive performance evaluation of Smart Crawler over real web data in 1 representative domains.Our evaluation shows that our crawling framework is very effective, achieving substantially higher harvest rates than the.

Figure 3 illustrates the adaptive learning process that is invoked periodically. For instance, the crawler has visited a pre-defined number of deep web sites or fetched a pre-defined number of forms. When a site crawling is completed, feature of the site is selected for updating FSS if the site contains relevant forms. During in-site exploring, features of links containing new forms are extracted for updating FSL.

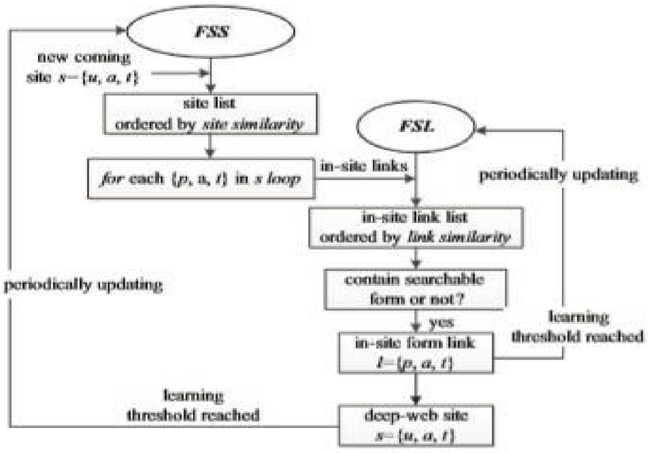
FSS
new coming site $s=\{u, a, t\}$
site list ordered by *site similarity*
FSL
in-site links
for each $\{p, a, t\}$ in *s loop*
periodically updating
in-site link list ordered by *link similarity*
periodically updating
contain searchable form or not?
yes
learning threshold reached
in-site form link $l=\{p, a, t\}$
learning threshold reached
deep-web site $s=\{u, a, t\}$

Fig-3:Adaptive Learning Process

6. CONCLUSION

We have shown that our approach achieves each wide coverage for deep net interfaces and maintains extremely efficient crawling.In this paper, we propose an effective harvesting framework for deep-web interfaces, namely SmartCrawler. Our experimental results on a representative set of domains show the effectiveness of the projected two-stage crawler that achieves higher harvest rates than different crawlers. Smart crawler is two stage crawler consisting site locating by reverse searching with centre pages and in site exploring consists adaptive link ranking and link tree for wider coverage.

REFERENCES

[1] Balakrishnan Raju, Kambhampati Subbarao, and Jha Manishkumar. **Assessing relevance and trust of the deep web sources and results based on inter-source agreement**. ACM Transactions on the Web, 7(2):Article 11, 1–32, 2013.

[2] Mustafa Emmre Dincturk, Guy vincent Jourdan, Gregor V. Bochmann, and Iosif Viorel Onut. **A model-based approach for crawling rich internet applications**. ACM Transactions on the Web, 8(3):Article 19, 1–39, 2014.

[3] Denis Shestakov and Tapio Salakoski**. On estimating the scale of national deep web**. In Database and Expert Systems Applications, pages 780–789. Springer, 2007. https://doi.org/10.1007/978-3-540-74469-6_76

[4] Kevin Chen-Chuan Chang, Bin He, Chengkai Li, Mitesh Patel, and Zhen Zhang. **Structured databases on the web: Observations and implication**s. ACM SIGMOD Record, 33(3):61–70,2004. https://doi.org/10.1145/1031570.1031584

[5] Wensheng Wu, Clement Yu, AnHai Doan, and Weiyi Meng. **An interactive clustering-based approach to integrating source query interfaces on the deep web**. In Proceedings of the 2004 ACM SIGMOD international conference on Management of data, pages 95–106. ACM, 2004. https://doi.org/10.1145/1007568.1007582

[6] Eduard C. Dragut, Thomas Kabisch, Clement Yu, and Ulf Leser**. A hierarchical approach to model web query interfaces for web source integration**. Proc. VLDB Endow., 2(1):325–336, August 2009. https://doi.org/10.14778/1687627.1687665

[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. https://doi.org/10.1016/S1389-1286(99)00052-3

[8] Jayant Madhavan, David Ko, Łucja Kot, Vignesh Ganapathy, Alex Rasmussen, and Alon Halevy**. Google's deep web crawl**. Proceedings of the VLDB Endowment, 1(2):1241–1252, 2008. https://doi.org/10.14778/1454159.1454163

[9] Olston Christopher and Najork Marc**. Web crawling. Foundations and Trends in Information Retrieval**, 4(3):175–246, 2010 https://doi.org/10.1561/1500000017

[10] Cheng Sheng, Nan Zhang, Yufei Tao, and Xin Jin. **Optimal algorithms for crawling a hidden database in the web**.Proceedings of the VLDB Endowment, 5(11):1112–1123, 2012. https://doi.org/10.14778/2350229.2350232

[11] Panagiotis G Ipeirotis and Luis Gravano. Distributed searchover the hidden web: **Hierarchical database sampling and selection.** In Proceedings of the 28th international conference on Very Large Data Bases, pages 394–405. VLDB Endowment, 2002.

[12] Nilesh Dalvi, Ravi Kumar, Ashwin Machanavajjhala, and Vibhor Rastogi. **Sampling hidden objects using nearest-neighbor oracles**. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1325–1333. ACM, 2011. https://doi.org/10.1145/2020408.2020606

[13] Jayant Madhavan, Shawn R. Jeffery, Shirley Cohen, Xin Dong, David Ko, Cong Yu, and Alon Halevy. **Web-scale data integration**:You can only afford to pay as you go. In Proceedings of CIDR, pages 342–350, 2007.

[14] Mohamamdreza Khelghati, Djoerd Hiemstra, and Maurice Van Keulen**. Deep web entity monitoring**. In Proceedings of the 22nd international conference on World Wide Web companion, pages 377–382. International World Wide Web Conferences Steering Committee, 2013. https://doi.org/10.1145/2487788.2487946

[15] Soumen Chakrabarti, Kunal Punera, and Mallela Subramanyam**. Accelerated focused crawling through online relevance feedback**. In Proceedings of the 11th international conference on World Wide Web, pages 148–159, 2002 . https://doi.org/10.1145/511446.511466

[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. https://doi.org/10.1145/1242572.1242631

[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.