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/ijccn29722018.pdf

https://doi.org/10.30534/ijccn/2018/29722018

A TWO-STAGE KEYWORD BASED CRAWLER FOR GATHERING 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, India, shruthipalmari96@gmail.com
[3] Student, India, sunimeco311@gmail.com
[4] Student, India, shushmagowda22@gmail.com
[5] Student, India, sushmitatadkal@gmail.com

ABSTRACT

Deep web is termed as sites present on web but not accessible by any 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. Keyword based 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 receives input from site locating and starts 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 inclination on visiting some more closely related links in inaccessible web directories, we design a data structure called link tree to achieve broader 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 crawler in web is a structure that moves all over internet for storing and gathering data into database for further set-up and study of data.

The Hidden Web refers to data 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 identified as a indicative gap in the scope of web indexes in light of the fact that web crawlers utilize internet searchers and depend on hyperlinks to find new web pages and normally do not have the ability to perform form submissions. Various records have shown that the Deep Web has more information than the presently searchable Internet. Moreover, 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 stated that 1.9 zettabytes data were found and 0.3 zettabytes were devoured worldwide. 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 resources contain a large quantity of important data and entities, for example, Info mine, 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 Keyword based crawler, for achieving both wide coverage and high efficiency for a focused crawler.

2. RELATED WORK

In the works of Olston and Najork, they methodically provide three steps for crawling deep web: finding location of deep web content sources, choosing relevant sources and obtaining underlying content.

Following their statement, we discuss the two steps closely related to our work as below:

Locating deep web content sources. A latest study depicted that the harvest rate of deep web is low — only 647,000 unique web forms were detected by considering 25 sample million pages from the Google index (about2.5%). Generic crawlers are usually developed for characterizing deep web and directory building of deep web resources that do not limit search on a specific topic but tries to extract all searchable forms. The Database Crawler used in the MetaQuerier is devised for automatically exploring 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. proposed a random sampling of hosts to characterize deep web, using the Host engine Yandex. I-Crawler combines pre-query and post-query approaches for classification of searchable forms.

Selecting relevant sources. Current hidden web directories normally have low coverage for applicable online databases which hinders their ability in fulfilling 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. Advancement to the best-first crawler is proposed, where instead of tracking all links in applicatory pages, the crawler used an extra classifier, the apprentice, to choose the most assuring 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 detect the deep web databases, because they are not enrolled with any search engines and are normally readily distributed and keep constantly changing. To address this problem, previous work has proposed two types of crawlers, generic crawlers and focused crawlers. Generic crawlers extract all searchable forms but do not 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 constructed 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 the above crawlers play a vital role in providing 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 unproductively 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 structure, namely Keyword based crawler, for efficient harvesting deep web interfaces. Keyword based crawler is organized 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 consequently the second in-site researching stage uncovers searchable structures from the site page.

The figure below shows the architecture of proposed system:

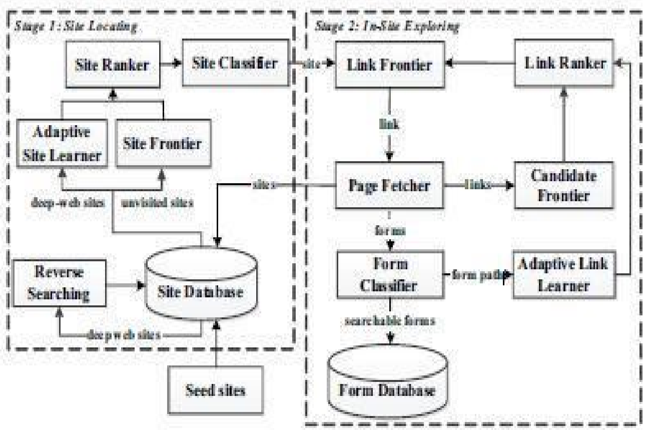

Fig. 1: The two-stage architecture of *SmartCrawler*.

Stage 1: Site locating –

In Site locating stage the Keyword based 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. Keyword based 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.

```
while # of candidate sites less than a threshold do
//pick   a deep website
    site = getDeepWebSite(siteDatabase,seedSites)
    resultP age = reverseSearch(site)
    links = extractLinks(resultP age)
    foreach link in links do
```

```
      page = downloadPage(link)
      relevant = classify(page)
      if relevant then
          relevantSites = extractUnvisitedSite(page)
       Output relevantSites
       end
   end
end.
```

3) Incremental site Prioritizing: Incremental site prioritizing is used to achieve broad coverage on websites. It records 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 consists 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.

```
HQueue=SiteFrontier.CreateQueue(HighPriority)
LQueue=SiteFrontier.CreateQueue(LowPriority)
while siteFrontier is not empty do
   if HQueue is empty then
      HQueue.addAll(LQueue)
       LQueue.clear()
   end
   site = HQueue.poll()
    relevant = classifySite(site)
   if relevant then
      performInSiteExploring(site)
      Output forms and OutOfSiteLinks
      siteRanker.rank(OutOfSiteLinks)
          if forms is not empty then
             HQueue.add (OutOfSiteLinks)
           end
          else
             LQueue.add(OutOfSiteLinks)
          end
    end
end
```

4) Site Frontier: Site Frontier retrieves 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.

5) 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 advanced by an Adaptive Link Learner, which grasps from the URL path leading to applicable forms.

6) 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.

7) Site Classifier: Out-of-site links that are classified as relevant by Site Classifier is present in high priority queue and are judged by Form Classifier to hold searchable forms. If site is judged 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 internal nodes which represent the directory.
Servlet: Dynamic request
Books: Catalogs of books
Docs directory: Help information.

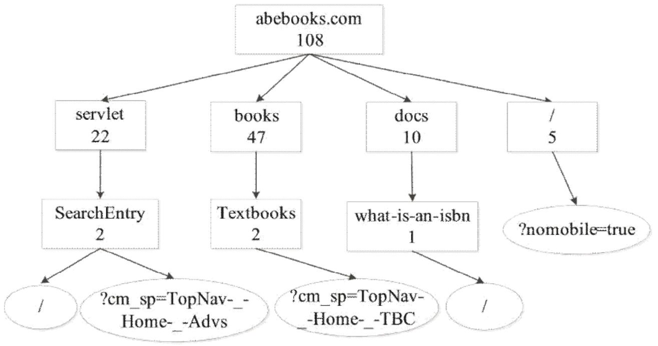

Fig. 2: Part of the link tree extracted from the homepage of http://www.abebooks.com, where ellipses represent leaf nodes and the number in a rectangle denotes the number of leaf nodes in its decedents.

1) 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.

2) Link Ranker: Link Ranker prioritizes links so that Keyword based crawler can discover searchable forms in less time. A high relevance score is awarded to a link that is most similar to links that directly point to pages with searchable forms.

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

4) Candidate Frontier: The links in web pages are downloaded 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 filters out the non-searchable forms.

5. MODULES

After accurate analysis the system has been classified into the following modules:

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

a. Two-stage crawler:

It is challenging to establish the deep web databases, because they are not enrolled with any search engines, are usually sparsely allotted, and remains constantly changing. To solve this problem, previous work has classified two types of crawlers, generic crawlers and focused crawlers. Generic crawlers fetch all searchable forms and cannot concentrate on a particular topic. Focused crawlers such as Form-Focused Crawler (FFC) can accordingly search onstream on a particular topic. FFC is formed with link, page, and form classifiers for focused crawling of web forms, and is continued by with further components for form refining and adaptive link learner. The link classifiers in these crawlers play a major role in achieving greater crawling effectiveness compared to the best-first crawler.

Although, these link classifiers are used to estimate the distance to the page which consists of searchable forms, and is tough to estimate, primarily for the delayed benefit links (links finally lead to pages with forms). As a result, the crawler can be inefficiently led to pages without targeted forms.

b. Site Ranker:

When merged with above stop-early policy. We address this problem by prioritizing highly relevant links with link ranking. Although, link ranking may bias for highly relevant links in certain directories. Our solution is to construct a link tree for a balanced link prioritizing. In general, each directory usually represents one type of files on web servers and it is beneficial to visit links in different directories. For links that only change in the query string part, are reviewed as the same URL because links are often spread unevenly in server directories, prioritizing links by the relevance can potentially bias toward some directories. Initially, the links under books might be allocated a high priority, because “book” is a main feature word in the URL. Simultaneously with the fact that most links come into existence in the books directory, it is feasible that links in other directories will not be chosen due to low relevance score. Therefore, the crawler may miss searchable forms in those directories.

c. Adaptive learning:

Adaptive learning algorithm that execute on stream feature selection and uses these properties to naturally build link rankers. In the site locating stage, high relevant sites are prioritized and the crawling is focused on a subject using the contents of the root page of sites, attaining more exact results. Throughout the InSite exploring stage, relevant links are prioritized for fast in-site searching. An extensive performance is done on an evaluation of Keyword based crawler over real web data in 1 representative domains. This evaluation results in crawling framework, is very effective, attaining considerably higher harvest rates than the.

Figure 3 shows the adaptive learning method that is invoked periodically. For example, the crawler has visited a pre-defined amount of deep web sites or drawn a pre-defined number of forms. When a site crawling is accomplished, feature of the site is designated for updating FSS if the site contains relevant forms. During in-site exploring, features of links comprising new forms are removed for updating FSL.

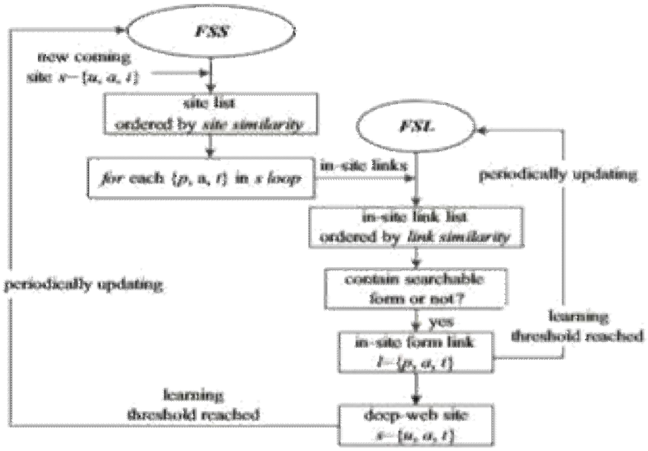

Fig-3: Adaptive LearningProcess

6. CONCLUSION

This paper indicates that our approach achieves each wide scope for deep net interfaces and preserves extremely efficient crawling. In this paper, we introduce an effective harvesting framework for deep-web interfaces, namely Keyword based crawler. Our preliminary results on a representative set of domains show the performance of the projected two-stage crawler that accomplish greater harvest rates than different crawlers. Keyword based crawler is two stage crawler containing site locating by reverse searching with center most pages and in site exploring consists adaptive link ranking and link tree for wider scope.

REFERENCES

[1] Balakrishnan Raju, Kambhampati Subbarao, and Jha Manish Kumar. **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 search over 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