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. 3, Issue 11, November 2015

Extracting the Web Data Through Deep Web Interfaces

Namish A. Diwate[1]; Kanchan Varpe[2]

M.E., Dept. of Computer, RMD Sinhgad School of Engineering, Pune., India[1]

Assistant Professor, Dept. of Computer, RMD Sinhgad School of Engineering, Pune., India[2]

ABSTRACT: As deep web grows at a very fast pace, there has been increased interest in techniques that help efficiently locate deep-web interfaces. However, 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. We propose a two-stage framework, namely SmartCrawler, for efficient harvesting deep web interfaces. In the first stage, SmartCrawler performs site-based searching for center pages with the help of search engines, avoiding visiting a large number of pages. To achieve more accurate results for a focused crawl, SmartCrawler ranks websites to prioritize highly relevant ones for a given topic. In the second stage, SmartCrawler achieves fast in-site searching by excavating most relevant links with an adaptive link-ranking. 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. Our experimental results on a set of representative domains show the agility and accuracy of our proposed crawler framework, which efficiently retrieves deep-web interfaces from large-scale sites and achieves higher harvest rates than other crawlers.

KEYWORDS: smartcrawler , ranking, reverse searching, adaptive learning, deep web.

I. INTRODUCTION

A Web Crawler (also known as a robot or a spider) is a system for the bulk downloading of web pages. Web crawlers are used for a variety of purposes. Most prominently, they are one of the main components of web search engines, systems that assemble a corpus of web pages, index them, and allow users to issue queries against the index and find the web pages that match the queries. A related use is web archiving (a service provided by e.g., the Internet archive [3]), where large sets of web pages are periodically collected and archived for posterity. A third use is web data mining, where web pages are analyzed for statistical properties, or where data analytics is performed on them (an example would be Attributor [5], a company that monitors the web for copyright and trademark infringements). Finally, web monitoring services allow their clients to submit standing queries, or triggers, and they continuously crawl the web and notify clients of pages that match those queries. The deep (or hidden) web refers to the contents lie behind searchable web interfaces that cannot be indexed by searching engines. Based on extrapolations from a study done at University of California, Berkeley, it is estimated that the deep web contains approximately 91,850 terabytes and the surface web is only about 167 terabytes in 2003 [1]. More recent studies estimated that 1.9 zettabytes were reached and 0.3 zettabytes were consumed worldwide in 2007 [2], [3]. An IDC report estimates that the total of all digital data created, replicated, and consumed will reach 6 zettabytes in 2014 [4]. A significant portion of this huge amount of data is estimated to be stored as structured or relational data in web databases — deep web makes 96% of all the content on the Internet, which is 500-550 times larger than the surface web [4], [3]. These data contain a vast amount of valuable information and entities such as Infomine [5], Clusty [3], Books In Print [4] may be interested in building an index of the deep web sources in a given domain (such as book). Because these entities cannot access the proprietary web indices of search engines (e.g., Google and Baidu).

II. OBJECTIVES

1) The Objective is to record learned patterns of deep web sites and form paths for incremental crawling.
2) Ranks site URLs to prioritize potential deep sites of a given topic. To this end, two features, site similarity and site frequency, are considered for 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. 3, Issue 11, November 2015

3) focused crawler consisting of two stages: efficient 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 effectively find many data sources for sparse domains.
4) SmartCrawler has an adaptive learning strategy that updates and leverages information collected successfully during crawling.

III. WEB CRAWLER

A web crawler (also known as a robot or a spider) is a system, a program that traverses the web for the purpose of bulk downloading of web pages in an automated manner. Web crawlers are prominently one of the main components of web search engines that assemble a corpus of web pages or creates a copy of all the visited pages, index them, and allow users to issue queries against the index, provide fast searches and find the web pages that match the queries. Interacting with hundreds of thousands of web servers and name servers, crawling is considered as the most fragile application since it is beyond the control of the system.

Crawler follows very simple steps yet very effective work in maintenance, checking of the downloaded links and also the validation of HTML codes as follows It starts with the list of URL's to visit, called seeds and downloads the web page.

IV. PROBLEM DEFINITION

An effective deep web harvesting framework, namely SmartCrawler, for achieving both wide coverage and high efficiency for a focused crawler. Based on the observation that deep websites usually contain a few searchable forms and most of them are within a depth of three our crawler is divided into two stages: site locating and in-site exploring. The site locating stage helps achieve wide coverage of sites for a focused crawler, and the in-site exploring stage can efficiently perform searches for web forms within a site.

V. RELATED WORK

Existing strategies were dealing with creation of a single profile per user, but conflict occurs when user's interest varies for the same query Eg. When a user is interested in banking exams in query "bank" may be slightly interested in accounts of money bank where not at all interested in blood bank. At such time conflict occurs so we are dealing with negative preferences to obtain the fine grain between the interested results and not interested. Consider following two aspects:

1) Document-Based method:

These methods aim at capturing users' clicking and browsing behaviour. It deals with click through data from the user i.e. the documents user has clicked on. Click through data in search engines can be thought of as triplets (q, r, c)

Where,

q = query
r = ranking
c = set of links clicked by user.

2) Concept-based methods:

These methods aim at capturing users' conceptual needs. Users browsed documents and search histories. User profiles are used to represent users' interests and to infer their intentions for new queries.

Disadvantages of Existing System:

1) Deep-web interfaces.
2) Achieving wide coverage and high efficiency is a challenging issue.

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. 3, Issue 11, November 2015

VI. PROPOSED SYSTEM MECHANISM

To efficiently and effectively discover deep web data sources, SmartCrawler is designed with two stage architecture, site locating and in-site exploring, as shown in Figure 1. The first site locating stage finds the most relevant site for a given topic, and then the second in-site exploring stage uncovers searchable forms from the site. Specifically, the site locating stage starts with a seedset of sites in a site database. Seeds sites are candidate sites given for SmartCrawler to start crawling, which begins by following URLs from chosen seed sites to explore other pages and other domains. When the number of unvisited URLs in the database is less than a threshold during the crawling process, SmartCrawler performs ”reverse searching” of known deep websites for center pages (highly ranked pages that have many links to other domains) and feeds these pages back to the site database. Site Frontier fetches homepage URLs from the site database, we going to rank the relevant information.

VII. SYSTEM ARCHITECTURE: (TWO STAGE ARCHITECTURE)

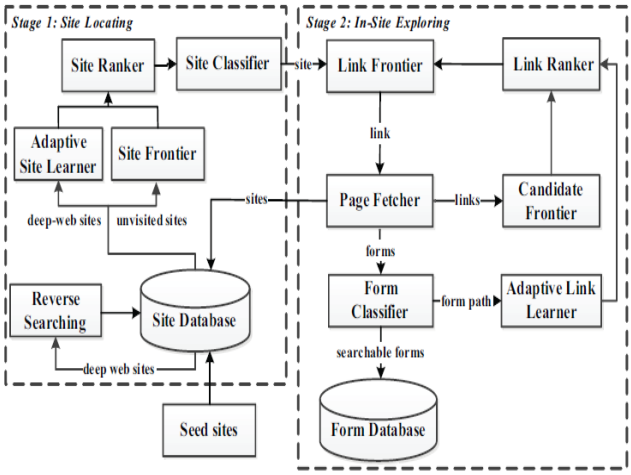

Fig.1 System Architecture: (Two stage Architecture)

To efficiently and effectively discover deep web data sources, SmartCrawler is designed with two stage architecture, site locating and in-site exploring, as shown in Figure. The first site locating stage finds the most relevant site for a given topic, and then the second in-site exploring stage uncovers searchable forms from the site. Specifically, the site locating stage starts with a seed set of sites in a site database. Seeds sites are candidate sites given for SmartCrawler to start crawling, which begins by following URLs from chosen seed sites to explore other pages and other domains. When the number of unvisited URLs in the database is less than a threshold during the crawling process, SmartCrawler performs ”reverse searching” of known deep websites for center pages (highly ranked pages that have many links to other domains) and feeds these pages back to the site database.

VIII. ALGORITHMS & TECHNIQUES USED

Algorithm 1: Reverse searching for more sites.
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)

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. 3, Issue 11, November 2015

5 links = extractLinks(resultP age)
6 foreachlink 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
Algorithm 2: Incremental Site Prioritizing.
Input :siteFrontier
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

IX. MODULE INFORMATION

1 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) 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.
2. 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. Figure 2 illustrates an example of a link tree constructed from the

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. 3, Issue 11, November 2015

homepage of http://www.abebooks.com. Internal nodes of the tree represent directory paths. In this example, servlet directory is for dynamic request; books directory is for displaying different catalogs of books; Amdocs directory is for showing help information. 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.

3. 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 in site 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 1representativedomains and compared with ACHE and site-based crawler. Our evaluation shows that our crawling framework is very effective, achieving substantially higher harvest rates than the state-of-the-art ACHE crawler. The results also show the effectiveness of the reverse searching and adaptive learning.

X. CONCLUSION

In this paper we have studied how to build an effective web crawler. The study carried out based on crawl ordering which reveals that the incremental crawler performs better and is more powerful because it allows re-visitation of pages at different rates. Crawling at other environment, such as peer-to-peer has been a future issue to be dealt with.

REFERENCES

1. Peter Lyman and Hal R. Varian. How much information? 2003. Technical report, UC Berkeley, 2003.
2. Roger E. Bohn and James E. Short. How much information? 2009 report on american consumers. Technical report, University of California, San Diego, 2009.
3. Martin Hilbert. How much information is there in the"information society"? Significance, 2012.
4. Idc worldwide predictions 2014: Battles for dominance – and survival on the 3rd platform. http://www.idc.com/research/Predictions14/index.jsp, 2014.
5. Michael K. Bergman. White paper: The deep web: Surfacing hidden value. Journal of electronic publishing, 2001.
6. Yeye He, Dong Xin, VenkateshGanti, SriramRajaraman, and Nirav Shah. Crawling deep web entity pages. In Proceedings of the sixth ACM international conference on Web search and data mining, 2013.
7. Feng Zhao, Jingyu Zhou, Chang Nie, Heqing Huang, Hai Jin. SmartCrawler: A Two-stage Crawler for Efficiently Harvesting Deep-Web Interfaces. Services Computing, IEEE Transactions on (Volume:PP , Issue:99).