Smart Crawler Frame Work for Effective and Efficiently Harvesting Deep Web Interface

B. Bhavitha
Dept of CSE
Besant Theosophical College

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 Smart Crawler, for efficient harvesting deep web interfaces. In the first stage, Smart Crawler 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, Smart Crawler ranks websites to prioritize highly relevant ones for a given topic. In the second stage, Smart Crawler 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.**

INTRODUCTION :

The deep 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. More recent studies estimated that 1.9 zeta bytes were reached and 0.3 zeta bytes were consumed worldwide in 2007. An IDC report estimates that the total of all digital data created, replicated, and consumed will reach 6 zeta bytes 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 up about 96% of all the content on the Internet, which is 500-550 times larger than the surface web. These data contain a vast amount of valuable information and entities such as Info mine, Crusty ,Books In Print 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), there is a need for an efficient crawler that is able to accurately and quickly explore the deep web databases. 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.

Our main contributions are:

1. We propose a novel two-stage framework to address the problem of searching for hidden-web resources. Our site locating technique employs a reverse searching technique (e.g., using Google's "link:" facility to get pages pointing to a given link) and incremental two-level site prioritizing technique for unearthing relevant sites, achieving more data sources. During the in-site exploring stage, we design a link tree for balanced link prioritizing, eliminating bias toward web pages in popular directories.

2. We propose an 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 a topic 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.

1. PURPOSE:

There is a need for an efficient crawler that is able to accurately and quickly explore the deep web databases.

2. SCOPE:

Our evaluation shows that our crawling framework is very effective, achieving substantially higher harvest rates than the state-of-the-art ACHE crawler.

3. *OVERVIEW* :

Web Testing tools are generally used to gather performance and stability information about the web application running on a particular server. Web Testing is divided into three main categories: Performance Testing Stability or Stress Testing Capacity Planning.

EXISTING SYSTEM:

The existing system is a manual or semi automated system, i.e. The Textile Management System is the system that can directly sent to the shop and will purchase clothes whatever you wanted. The users are purchase dresses for festivals or by their need. They can spend time to purchase this by their choice like color, size, and designs, rate and so on. They but now in the world everyone is busy. They don't need time to spend for this. Because they can spend whole the day to purchase for their whole family . So we proposed the new system for web crawling.

Consider following two aspects:

1) Document-Based method :These methods aim at capturing users' clicking and browsing behaviors. 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 :

1) Deep-web interfaces.

2) Achieving wide coverage and high efficiency is a challenging issue

PROPOSED SYSTEM:

The system proposes a two-stage framework, namely Smart Crawler, for efficient harvesting deep web interfaces. In the first stage, Smart Crawler 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, Smart Crawler ranks websites to prioritize highly relevant ones for a given topic. In the second stage, Smart Crawler 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. Propose an effective harvesting framework for deep-web interfaces, namely Smart Crawler. We have shown that our approach achieves both wide coverage for deep web interfaces and maintains highly efficient crawling. Smart Crawler is a focused crawler consisting of two stages: efficient site locating and balanced in-site exploring. Smart Crawler 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. By ranking collected sites and by focusing the crawling on a topic, Smart Crawler achieves more accurate results.

MODULE DESCRIPTION:

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

- Two-stage Crawler.
- Site Ranker.
- Adaptive Learning.

TWO-STAGE CRAWLER:

- It is a two stage approach which is known as smart crawler. The two stages are site-locating and in-site exploring searching.
- In the first stage, Smart Crawler 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, Smart Crawler ranks websites to prioritize highly relevant ones for a given topic.
- In the second stage, Smart Crawler 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.

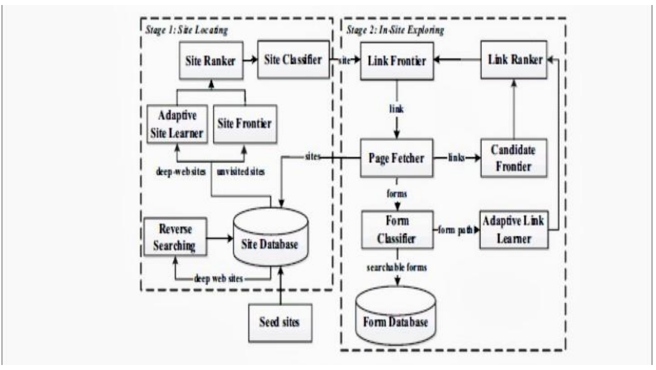
SITE Ranker:

- It assigns a score for unvisited site that corresponds to its relevance to the already discovered deep web sites.

ADAPTIVE LEARNING:

- It updates and leverages information collected successfully during crawling.

SYSTEM ARCHITECTURE:



To efficiently and effectively discover deep web data sources, Smart Crawler is designed with two stage architecture, site locating and in-site exploring, as shown in Figure 3.1. 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 seed set of sites in a site database. Seeds sites are candidate sites given for Smart Crawler 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, Smart Crawler performs ”reverse searching” of known deep web sites for center pages (highly ranked pages that have many links to other domains) and feeds these pages back to the site database.

ALGORITHMS

1. ALGORITHM 1: REVERSE SEARCHING FOR MORE SITES INPUT:

```
Seed sites and harvested deep websites
Output: relevant sites
While # of candidate sites less than a threshold do
  // pick a deep website
  Site = get Deep Web Site (site Database, seed Sites)
  Result Page = reverse Search (site)
  Links = extract Links (result Page)
  For each link in links do
    Page = download Page (link)
    Relevant= classify (page)
    If relevant then
      Relevant Sites=extract Unvisited Site (page)
      Output relevant Sites
        End
      End
  End
```

ALGORITHM 2:

INCREMENTAL SITE PRIORITIZING:

```
Input: site Frontier
Output: searchable forms and out-of-site links
H Queue=Site Frontier. Create Queue (High Priority)
L Queue=Site Frontier. Create Queue (Low Priority)
if H Queue is empty then
H Queue. Add All (L Queue)
L Queue . clear()
End
Site = H Queue. poll()
Relevant = classify Site(site)
if relevant then
Perform In Site Exploring(site)
Output forms and Out Of Site Links
Site Ranker .rank(Out Of Site Links)
if forms is not empty then
H Queue .add (Out Of Site Links)
End
Else
L Queue. add  (Out Of Site Links)
End
End
End
```

CONCLUSION:

We propose an effective harvesting framework for deep-web interfaces, namely Smart Crawler. We have shown that our approach achieves both wide coverage for deep web interfaces and maintains highly efficient crawling. Smart Crawler is a focused crawler consisting of two stages: efficient site locating and balanced in-site exploring. Smart Crawler 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. By ranking collected sites and by focusing the crawling on a topic, Smart Crawler achieves more accurate results. The in-site exploring stage uses adaptive link ranking to search within a site; and we design a link tree for eliminating bias toward certain directories of a website for wider coverage of web directories. Our experimental results on a representative set of domains show the effectiveness of the proposed two stage crawler, which achieves higher harvest rates than other crawlers. In future work, we plan to combine pre-query and post-query approaches for classifying deep-web forms to further improve the accuracy of the form classifier.

REFERENCE:

[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, 9(4):8–12, 2012.
[4] Ida 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, 7(1), 2001.
[6] Yeye He, Dong Xin, Venkatesh Ganti, Sriram Rajaraman, and Nirav Shah. Crawling deep web entity pages. In Proceedingsof the sixth ACM international conference on Web search and data mining, pages 355–364. ACM, 2013.
[7] Infomine. UC Riverside library. http://lib-www.ucr.edu/, 2014.
[8] Clusty’s searchable database dirctory. http://www.clusty. com/, 2009.
[9] Booksinprint. Books in print and global books in print access. http://booksinprint.com/, 2015.
[10] 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.
[11] 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.
[12] 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.
[13] Denis Shestakov and Tapio Salakoski. On estimating the scale of national deep web. In Database and Expert Systems Applications, pages 780–789. Springer, 2007.
[14] 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.
[15] Luciano Barbosa and Juliana Freire. Searching for hidden-web databases. In WebDB, pages 1–6, 2005.
[16] 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.
[17] 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.
[18] 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.
[19] Olston Christopher and Najork Marc. Web crawling. Foundations and Trends in Information Retrieval, 4(3):175–246, 2010.
[20] Balakrishnan Raju and Kambhampati Subbarao. Sourcerank: Relevance and trust assessment for deep web sources based on inter-source agreement. In Proceedings of the 20th international conference on World Wide Web, pages 227–236, 2011.