International Journal of Computer Applications (0975 – 8887)
Volume 150 – No.1, September 2016

Deep Web Crawler: Exploring and Re-ranking of Web Forms

Rashmi K. B.
PG Scholar, Dept. of CS&E
Bangalore Institute of Technology
Bangalore, Karnataka, India

Vijaya Kumar T.
Associate Professor, Dept. of CS&E
Bangalore Institute of Technology
Bangalore, Karnataka, India

H. S. Guruprasad, PhD
Professor and Head, Dept. of CS&E
BMS college of Engineering
Bangalore, Karnataka, India

ABSTRACT

A huge portion of the web known as deep web is accessible via search interfaces to myriads of databases on the web. Deep web crawl is concerned with the problem of surfacing hidden content behind search interfaces on the web. Given the dynamic nature of the web, where data sources are constantly changing, it is crucial to discover these resources. The paper proposes a two level application namely deep web crawler for gathering relevant searchable forms. In the first level deep web crawler explores the forms based on reverse searching for a given seed site, ranking the sites to prioritize highly relevant sites and by extracting the links to find the forms. In the next level, it searches the forms based on preference and the result is enhanced by re ranking, given the user feedback.

Keywords

Deep web, adaptive learning, ranking

1. INTRODUCTION

The internet has vast amount of information, there is a need of an efficient mechanism for finding relevant information. Web crawlers offer that provision to the search engine. Web crawlers are the programs used for indexing the content by scanning over the internet. This is applied only for surface web which is used and indexed by conventional search engines. The hidden web also called as deep web. It is part of the internet content which is non indexed by conventional search engines and it is the response to a submitted query or forms. Therefore, they are non searchable by these engines.

Internet is the quickest developing mean for information. In the year 2000 it was found that the public web information was 20 to 50 terabytes. In the year 2003 it was 167 terabytes, which was three times more than in the year 2000. The deep web was approximately 91,850 terabytes. This calculation is based on the studies made at University of California, Berkeley [1]. Based on the report on growth of data by International Data Corporation (IDC), the total quantity of digital data produced in the year 2007 was 281 billion gigabytes [2]. In 2014 alone, approximately 6 trillion terabytes of digitized information was created, replicated and consumed [3].

According to Bergman, substantial portion of this large amount of digital data considered as deep web, which are the pages that do not exist on conventional search engines. They are created dynamically in response to the search queries. Hidden web is a portion of World Wide Web which cannot be approached via link-crawling search engines like Google. This part of internet can be accessed by filling query forms. Hidden web is approximately 400-500 times more than the surface web. This information resides in the databases and 95% of it is not publicly accessible [4].

Deep web databases are not indexed by any search engines, not densely dispersed and they keep changing as it is dynamic data. Hence it is challenging to locate these database contents. Existing approaches to address this problem are of two types, generic crawlers and focused crawlers. Generic crawlers are domain independent which retrieves all possible searchable forms irrespective of the topic. Focused crawlers are domain specific, results in searchable forms focusing on specific topic. This includes Form-Focused Crawlers called as FFC and Adaptive Crawler for Hidden-Web Entry Points called as ACHE. FFC has three classifiers namely link, page and form classifiers for crawling the deep web content for particular domain. ACHE is extended with Adaptive link learning approach. Link classifiers in above crawlers predict the distance to the links which later leads to pages with searchable forms i.e., delayed benefit links. This may affect the efficiency of crawlers by leading to links without any forms.

This paper work proposes a focused crawler for deep web harvesting to gather searchable forms which are the entry points to the databases. It is been observed that only few searchable forms are found in deep websites and most of them within the depth of three [5]. Hence used the stopping criteria which prevent unproductive crawling. The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 describes the system overview. Section 4 concludes the paper.

2. RELATED WORK

According to Bergman, exploring the content over the Internet is like carting the net over the surface of the ocean. Deep inside valuable information resides which is left out because most of the content is dynamically generated and not searchable by standard search engines. These engines use crawling functionality for indexing. They index only static pages and cannot find deep web which are returned in response to a submitted query. Deep web contains domain specific relevant information. Bergman makes an effort to figure out the size of deep web by several methods which includes Analysis procedure, Overlap analysis and Page views [4].

There are two classes of approaches to identify search interfaces to online databases: pre-query and post-query approaches. Pre-query approaches identify searchable forms on web sites by analyzing the features of web forms. Post-query techniques identify searchable forms by submitting the probing queries to forms and analyzing the result pages. Bergholz and Chidlovskii [6] gave an example of the post-query approach for the automated discovery of search interfaces. They implemented a domain specific crawler that starts on indexable pages and detects forms relevant to a given domain. Next, the Query Prober submits some domain-specific phrases (called "positive" queries) and some nonsense words ("negative" queries) to detected forms and then assesses whether a form is searchable or not by comparing the resulting pages for the positive and negative queries. Cope et al. [7] proposed a pre-query approach that uses automatically generated features to describe candidate forms and uses the decision tree learning algorithm to classify them based on the generated set of features.

In [8], Kevin et al., describes the metaquerier system which is a generic crawler for finding and querying databases. It is explained that most forms resides close to the home page of the website, defined as depth. It is the least possible count of moves from home page of the main site to the web page consists of searchable forms.

In [9], Luciano Barbosa et al., describes a "Form Crawler Architecture" which is a focused crawler called as Form Focused Crawler (FFC). This is used to find forms by searching for the specific topic which prevents unproductive crawling. FFC uses three classifiers namely page, link and form classifier. A page classifier is used to grade the relevant pages for a specific topic. A link classifier identifies the links with forms and also the links which eventually leads to forms. It ranks the links based on their significance with the topic. A form classifier is used to separate out non searchable forms and useless forms. It involves the choices of relevant features and forms to train the link classifier by manual selection.

In [10], Luciano Barbosa et al., proposes ACHE (Adaptive Crawler for Hidden-Web Entry Points) architecture. Most of the deep web content remains in the databases and there is a need to explore the searchable forms which are the entry points to the deep web databases. They implemented adaptive learning process where crawling is enhanced by using prior knowledge. ACHE is a focused crawler with an adaptive learning approach which enhances the harvest rates by automatic feature selection compared to crawlers with manual tuning. As the deep web is sparsely distributed, back crawling method is required to improve the learning process by increasing sample paths. The limitation of this approach is having limited sample path.

In [11], Shestakov Denis., concentrate on three classes of problems around the deep web: characterization of deep web, finding and classifying deep web resources and querying web databases. He proposes the system for discovering and classifying search interfaces named as I-crawler [12]. I-Crawler includes stages like Site/Page Analyzer to know which site to be processed first, Interface Identification for finding the forms, Interface Classification for page classifier as searchable or non-searchable and Form Database for the storage purpose. This is a generic crawler which gathers all searchable forms without particular domain.

There is a survey about web crawling and its challenges and solutions. According to Olston and Najork, a deep web crawler makes an effort to find searchable html forms which are not produced by hyperlinks of conventional crawlers. It includes identifying sites which contain forms that in turn lead to hidden content, selecting the relevant content and finally extract the content which was selected [13]. Following their statement, paper discusses the two steps closely related to the proposed work as locating deep web content sources and selecting relevant sources.

There is a survey about estimating the size of deep web by considering one national web domain in particular Russian web. Denis discusses about the limitation of neglecting virtual hosting where IP address shared among many websites. Host-IP clustering method is employed to overcome this drawback. This includes grouping of hosts who shares the same IP address [14].

The main aim of crawling over the deep web is to find the content which is hidden behind the surface web. Conventional literature is more about deep web sites in the form of textual documents for example PubMed, but significant information is concealed in structured entities for example online shopping sites. In [15], Yeye He et al., briefs the prototype system developed which specifies crawling the entity oriented content. This system include components like query generation, URL deduplication and empty page filtering. The number of sites obtained is reduced because URL template generation is only based on HTML "GET" forms but not "POST" forms.

The proposed system is motivated by the model "Smart Crawler" developed by Feng Zhao et al. SmartCrawler is a framework for locating the deep web interfaces. It includes Site Locating for fetching enough sites for crawling, In-site Exploring for finding web forms within a site [16]. The paper contributes some enhancement by combining pre and post query constraints which improves the accuracy of form classification.

3. SYSTEM OVERVIEW
3.1 System Architecture

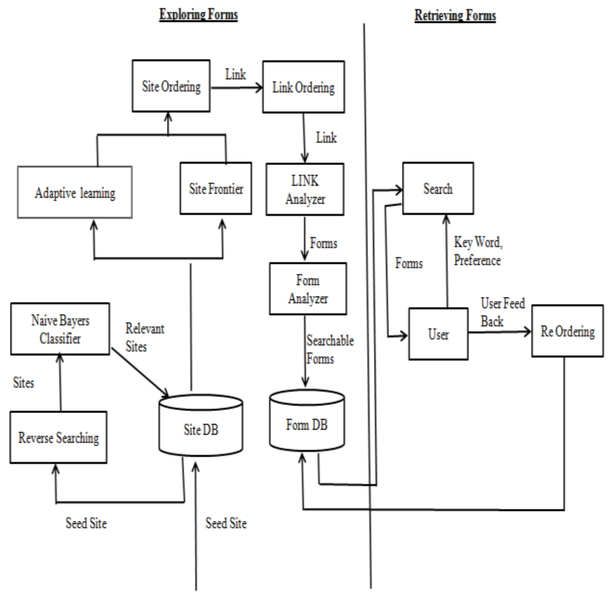

Fig. 1: System architecture

To efficiently and effectively discover deep web data sources, Deep web crawler is designed with two levels, exploring forms and retrieving forms, as shown in Figure 1.

The main aim of exploring forms is to collect the relevant sites for a given seed site and keyword, to find the searchable forms out of it. The main aim of retrieving forms is to fetch the relevant searchable forms from the database and reordering those forms based on user feedback.

Seed site must be preconfigured and added to site database. Reverse searching will be done for known deep website that is seed site and the resultant sites will fed into naive bayes classifier for finding the relevant sites based on given topic, according to home page content. Site frontier fetches homepage URL from the site database which is unvisited and provides to the site ordering. Adaptive learning used to keep track of deep sites. It provides deep site URL to the site ordering. Site ordering prioritizes highly relevant sites and it is improved by an adaptive site learner. Link ordering ranks the links which are extracted from the ranked sites. Link analyzer reads the webpage from the URL link, by fetching page content and check for forms. Form analyzer explores relevant searchable forms by leaving login forms. These searchable forms are added to the form database. User will provide keyword with preference to get relevant result. Search function collects all sorted forms from the database. These forms will be re-ranked based on user feedback rating.

3.2 Reverse Searching

Reverse searching module is implemented in order to increase the number of sites to crawl. This is achieved by making use of existing search engines feature such as Google's "link" facility. For instance, link: www.abebooks.com this provides all web pages which have links pointing to the abebooks home page.

3.3 Naive Bayes Classifier

Naive Bayes Classifier is a probabilistic classifier based on Bayes theorem. This module is used to find relevant sites out of all sites resulted from reverse searching. The seed site home page content is downloaded to text file and important terms are selected by filtering out stop words like adjectives, articles, prepositions and excluding irrelevant tags. Naive bayes class is trained with all the important terms. The probability of match is found between given site and with the training set. The higher probability sites are considered as relevant.

The procedure followed in the algorithm is explained in form of steps are as follows:

Input: Training terms of seed site, Test terms of new site

Output: Accuracy of the classifier'

```
begin
   call train function of the classifier with training terms
      for each new site
         call match function of the classifier for test terms
         return accuracy
      end for
end
```

3.4 Adaptive Learning

Adaptive learning is a strategy that updates and provides the information gathered during crawling. This is used while ranking sites and links. Feature Space for Sites (FSS) and Feature Space for Links (FSL) are loaded and ranking process is done by similarity metric with FSS and FSL respectively for sites and links. If crawled site contains searchable forms FSS is updated with this site. The links extracted from this site which contains forms are updated in FSL, as shown in Figure 2. This plays a major role in finding relevant sites and links while exploring searchable forms.

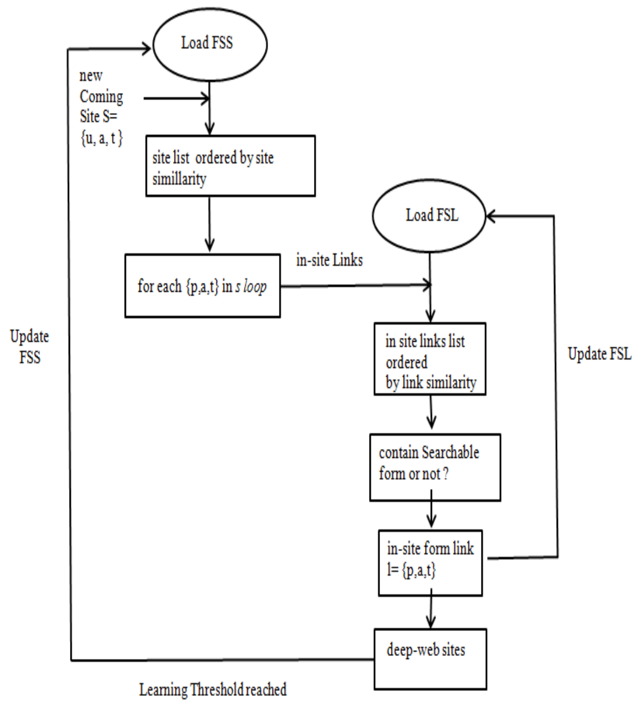
Load FSS
new Coming Site S= {u, a, t}
site list ordered by site simillarity
Load FSL
in-site Links
for each {p,a,t} in s loop
Update FSS
in site links list ordered by link similarity
Update FSL
contain Searchable form or not ?
in-site form link l= {p,a,t}
deep-web sites
Learning Threshold reached

Fig. 2: Adaptive learning

3.5 Site Ordering

Site ordering is sorting the sites based on ranks. Ranking mechanism involves two features Site similarity and Site frequency. This is accomplished by online construction of attributes for the site.

The attributes of a site specified as FSS= {U, A, T}

Where U is the features of URL,

A is the Anchor and

T is the text around URL of a site.

Features of URL considered are protocol, authority, host, path, query and reference. Anchor includes extracting the links of a site. Term is the vector of important terms of a site excluding stop words.

The site similarity is found between known deep site and new site. It is based on URL score, Link score and Term score. Consider the known deep site with attributes {U, A, T} and new site s with attributes {Uns, Ans, Tns} whose rank is to be find out. Site similarity of new site with the known deep site is calculated as follows

$SS(s) = Sim(U, Uns) + Sim(A, Ans) + Sim(T, Tns)$

Where Sim (a1, a2) is the function to compute score based on cosine similarity between the two vectors a1 and a2 as

$Sim(a1, a2) = a1 \cdot a2 / |a1| X |a2|$

The site frequency calculates the number of times new site s appears in known deep site.

The rank of a new site is calculated by the linear combination of site similarity and site frequency.

$Rank(new\_site) = \alpha \times ST(new\_site) + (1 - \alpha) \times \log(1 + SF(new\_site))$,

Where $0 \leq \alpha \leq 1$

3.6 Effectiveness Of Deep web crawler

Table 1 shows the results of searching for the book domain after crawling, given the seed site.

Table 1. Experimental result of crawling over the book domain

Total Forms	Searchable Forms	Pre query Constraints	Filtered Forms
143	118	No preference	118
143	118	Author	100
143	118	Author, publisher, price	72
143	118	Irrelevant preference	0

3.7 Preference Impact

Figure 3 shows the impact of preference. It fetches all searchable forms when no preference is given and finds relevant forms when preferences are given and finds no forms when irrelevant preference given for the domain.

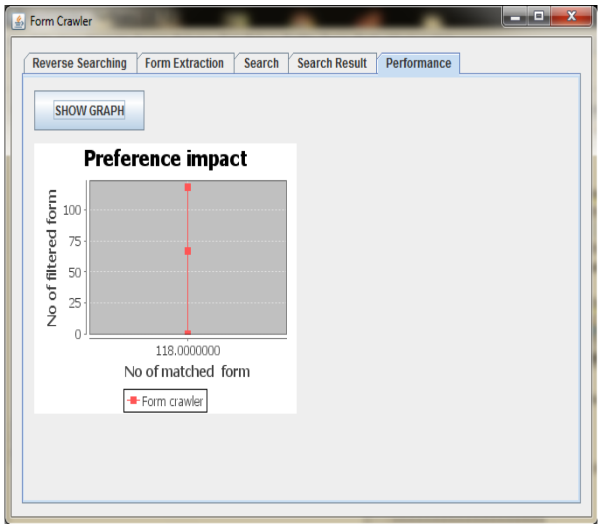

Fig. 3: Preference impact graph

4. CONCLUSION

The paper proposes an application that focuses on exploring searchable forms. Deep web crawler is capable of executing extensive search by focusing the search on a given domain. Deep web crawler is a focused crawler comprises of two levels namely exploring forms and searching forms. Deep web crawlers explore forms by reverse searching, naive bayes classifier and ranking mechanisms. Reverse searching collects the sites and naïve bayes classifier gets the relevant sites. Relevant sites are ranked based on site similarity and site frequency. Relevant ranked sites are used to extract the links to locate the searchable forms. Deep web crawler searches the forms based on the preference mentioned, and the result is enhanced with user feedback. Adaptive learning is applied which enhances the harvesting rate by updating the feature space of site and link used for crawling. Deep web crawler improves the accuracy of crawling by combining pre and post query approach by providing preference and user feedback. User feedback is given as rating to each form by considering the quality of form results.

5. 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] Idc worldwide predictions 2014: Battles for dominance and survival on the 3rd platform. http://www.idc.com/research/Predictions14/index.jsp, 2014.

[4] Michael K. Bergman. White paper: The deep web: Surfacing hidden value. Journal of electronic publishing, 7(1), 2001.

[5] 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.

[6] Bergholz A. and Childlovskii B. Crawling for Domain-Specific Hidden Web Resources. In: Proc. of WISE 2003, pp. 125–133 (2003).

[7] Cope J., Craswell N. and Hawking D. Automated Discovery of Search Interfaces on the Web. In: Proc. of ADC 2003, pp. 181–189 (2003).

[8] 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.

[9] Luciano Barbosa and Juliana Freire. Searching for hidden-web databases. In WebDB, pages 1–6, 2005.

[10] 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.

[11] Shestakov, D.: Characterization of National Deep Web. TUCS Technical Report 892(2008).

[12] 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.

[13] Olston Christopher and Najork Marc. Web crawling. Foundations and Trends in Information Retrieval, 4(3):175–246, 2010.

[14] 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.

[15] Yeye He, Dong Xin, Venkatesh Ganti, Sriram Rajaraman and Nirav shah. Crawling deep web entity pages. In proceedings of the sixth ACM international conference on web search and data mining, pages 355-364. ACM, 2013.

[16] Feng Zhao, Jingyu Zhou, Chang Nie, Heqing Huang and Hai Jin. SmartCrawler: A Two-stage Crawler for Efficiently Harvesting Deep-Web Interfaces. IEEE Transactions on Services Computing, 2015.