A Scalable Autonomous Digital Library with Semantic Focused Crawling Mechanism

S.Bavithra, M-TECH(IT),
Dr. Sivanthi Aditanar College of Engineering,
Tiruchendur, Tamil Nadu.

M.Jancyrani Malli, AP/IT
Dr. Sivanthi Aditanar College of Engineering,
Tiruchendur, TamilNadu.

***Abstract*— To build the digital library using social semantic focused crawling, In which the semantic focused crawling mechanism is extended by the deep we harvesting frame work. As the huge amount of information in the deep web grows, there has been increased significance in techniques and tools that help efficiently locate deep web interfaces. The nature of the deep web is dynamic. Because of this and huge amount of web resources obtaining high efficiency and broad coverage is challenging issue. This paper proposes a novel two stage approach, namely, Smart web Crawler for harvesting deep web interfaces. First stage provides broad coverage where Second stage proves high efficiency. In the First Stage, Smart Web Crawler discovers relevant sites for the given topic. In the Second stage, it uncovers searchable forms from the site. Smart Web Crawler achieves higher harvest rate than other crawler.**

Key Words:—Deep web, two-stage crawler, feature selection, ranking, adaptive learning

I.INTRODUCTION

The World Wide Web is an architectural framework for accessing linked documents spread out over millions of machines all over the Internet. The popularity of www is largely dependent on the search engines. Search engines are the gateways to the huge information repository at the internet. Search engine consist of four discrete components: Crawling, Indexing, Ranking and query processing.

The earliest Web search engines relied for retrieving information from Web pages on the bases of matching with the words in the search query. As the Web continues to grow, and as the diversity of users increases search engines must utilize semantic clues to satisfy user's information needs. Semantic search, which takes into account the interests of the user as well as the specific context in which the search is issued, is the next step in providing users with the most relevant information possible.

Currently the general purpose search engines strive as entry points for the web pages perform the coverage of information that is as broad as possible. They use Web crawlers to maintain their index databases. These crawlers are blind and exhaustive in their approach, with comprehensiveness as their major goal. A URL (Uniform Resource Locator) that is a URI (Uniform Resource Identifier) specifies where an identified resource is available. In order to search most relevant information, crawlers can be more selective about the URL they fetch and refer as to be crawled this mechanism

Focused crawling provides a better alternative to generic crawling; by this type of crawling topic driven web search can be done to retrieve relevant web pages to a pre-defined set of topics. The correct selection of seed URL, directs focused crawler to identify the search area in the web. Semantic Focused Crawler utilizes the social web and semantic knowledge to crawl relevant resource from selected portion of web. The factor that may affect the result of focused crawler is the selection of relevant terms by expanding search topics. A particular topics or concept can be represented with many different related words, the relevance of the terms is considered based on the learner's searching intension.

To build the digital library using social semantic focused crawling, In which the semantic focused crawling mechanism is extended by the deep we harvesting frame work.The deep (or hidden) web refers to the contents lie behind searchable web interfaces that cannot be indexed by searching engines. These Hidden data contain a vast amount of valuable information and entities such as may be interested in building an index of the deep web sources in a given domain (such as book). 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.

Besides efficiency, quality and coverage on relevant deep web sources are also challenging. Crawler must produce a large quantity of high-quality results from the most relevant content sources. For assessing source quality, SourceRank ranks the results from the selected sources by computing the agreement between them. When selecting a relevant subset from the available content sources, FFC and ACHE prioritize links that bring

immediate return (links directly point to pages containing searchable forms) and delayed benefit links.

To propose an effective deep web harvesting framework, for 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
- 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. The site locating technique employs a reverse searching technique and incremental two-level site prioritizing technique for unearthing relevant sites, achieving more data sources.

The in-site exploring stage, we design a link tree for balanced link prioritizing, eliminating bias toward web pages in popular directories. The 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.

II.SYSTEM ARCHITECTURE

To efficiently and effectively discover deep web data sources, with a two stage architecture, site locating and in-site exploring, as shown in Figure 1 . Initially the user query is received and it is expanded by using its semantic similarity and then the result from the search engine is given to seed site database.

The 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 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, 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. Site Frontier fetches homepage URLs from the site database, which are ranked by Site Ranker to prioritize highly relevant sites. 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. To achieve more accurate results for a focused crawl, Site Classifier categorizes URLs into relevant or irrelevant for a given topic according to the homepage content.

After the most relevant site is found in the first stage, the second stage performs efficient in-site exploration for excavating searchable forms. Links of a site are stored in Link Frontier and corresponding pages are fetched and embedded forms are classified by Form Classifier to find searchable forms. Additionally, the links in these pages are extracted into Candidate Frontier. To prioritize links in Candidate Frontier, Crawler ranks them with Link Ranker. Note that site locating stage and in-site exploring stage are mutually intertwined. When the crawler discovers a new site, the site’s URL is inserted into the Site Database. The Link Ranker is adaptively improved by an Adaptive Link Learner, which learns from the URL path leading to relevant forms

III.TWO STAGES OF CRAWLING

A. Site Locating

The site locating stage finds relevant sites for a given topic, consisting of site collecting, site ranking, and site classification.

Site Collecting:

The site locating stage starts with a seed set of sites in a site database. Seeds sites are candidate sites given 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, 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. Site Frontier fetches homepage URLs from the site database, which is ranked by Site Ranker to prioritize highly relevant sites.

Site Ranker:

Once the Site Frontier has enough sites, the challenge is how to select the most relevant one for crawling. Site Ranker assigns a score for each unvisited site that corresponds to its relevance to the already discovered deep web sites.

Site Classifier:

After ranking, Site Classifier categorizes the site as topic relevant or irrelevant for a focused crawling. If a site is classified as topic relevant, a site crawling process is launched. Otherwise, the site is ignored and a new site is picked from the frontier.

B. In-Site Exploring

Once a site is regarded as topic relevant, in-site exploring is performed to find searchable forms. The goals are to quickly harvest searchable forms and to cover web directories of the site as much as possible. To achieve these goals, in-site exploring adopts two crawling strategies for high efficiency and coverage. Links within a site are prioritized with Link Ranker and Form Classifier classifies searchable forms.

Link Ranker:

Link Ranker prioritizes links so that Crawler 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.

Form Classifier:

Classifying forms aims to keep form focused crawling, which filters out non-searchable and irrelevant forms .DSFC(domain-specific form classifier) judges whether a form is topic relevant or not based on the text feature of the form, that consists of domain-related terms.

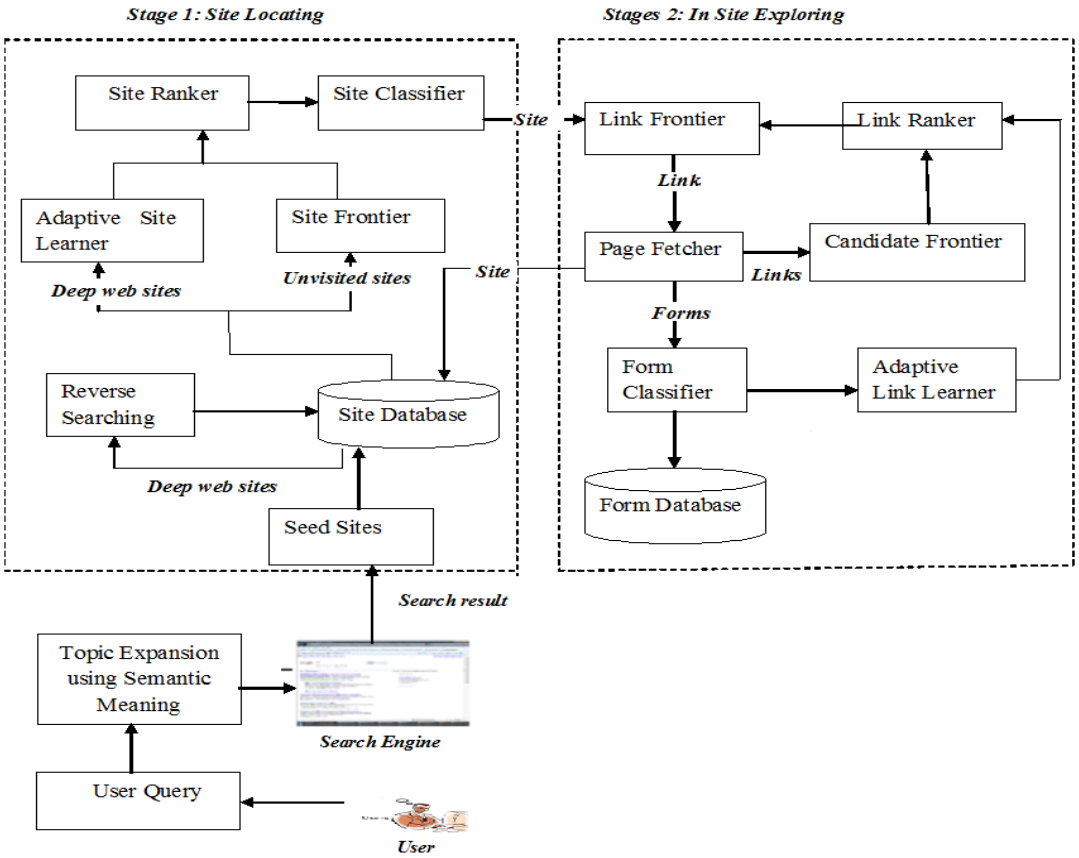

Figure 1: System Architecture

IV.ALGORITHMS

A. Reverse searching Algorithm

This algorithm is introduced when the crawler initializes and when the size of site frontier decreases to threshold. In Smart Web Crawler System, Firstly the result page from search engine is parsed to extract links. Then these pages are downloaded and analysed to decide whether the links are relevant or not. This can be done using heuristic rules. They are given as follows:

- If page contains related searchable forms then it is relevant.

If the number of seed sites or fetched deep web sites in the page is larger than user defined threshold then page is relevant.

B. Incremental Site Prioritizing Algorithm

Incremental Site Prioritizing One can make crawling process resumable and achieve deep coverage on websites. This can be achieved by using incremental site prioritizing. The main idea of incremental site prioritizing is to capture the learned patterns of deep web sites and form paths for incremental crawling. Firstly, the prior knowledge is used for initializing Site Ranker and Link Ranker.

Then unvisited sites are assigned to the Site Frontier and unvisited sites are prioritized by Site Ranker and visited sites are inserted to fetched site list. Smart Web Crawler accurately classify out-of-site links. Site Frontier uses two queues to save unvisited sites. These two queues are High Priority Queue and Low Priority Queue. Site Ranker is assigned relevant scores for prioritizing sites. The Low Priority Queue is used to provide more candidates sites.

Pseudo code for Reverse Searching Algorithm

```
Algorithm 1: Reverse searching
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 resultPage= reverseSearch(site)
5 links = extractLinks(resultP age)
6   foreachlinks 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
```

Pseudo code for Incremental Site Prioritizing Algorithm

```
Algorithm 2: Incremental Site Prioritizing.
Input: siteFrontier
Output: searchable forms and out-of-site links
1 HQueue=SiteFrontier.CreateQueue(HighPriority)
2 LQueue=SiteFrontier.CreateQueue(Low Priority)
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        end
15        if forms are not empty then
16            HQueue.add (OutOfSiteLinks)
17        end
18        else
19            LQueue.add (OutOfSiteLinks)
20        end
21 end
```

V.FEATURE SELECTION AND RANKING

Crawler encounters a variety of web pages during a crawling process and the key to efficiently crawling and wide coverage is ranking different sites and prioritizing links within a site. This section first discusses the online feature construction of feature space and adaptive learning process of Crawler, and then describes the ranking mechanism.

A. Online Construction of Feature Space

In patterns of links to relevant sites and searchable forms are learned online to build both site and link rankers. The ability of online learning is important for the crawler to avoid biases from initial training data and adapt to new patterns. The feature space of deep web sites (FSS) is defined as:

$$FSS=\{U,A,T\}, \quad (1)$$

where U, A, T are vectors corresponding to the feature context of URL, anchor, and text around URL of the deep web sites. The feature space of links of a site with embedded forms (FSL) is defined as:

$$FSL=\{P,A,T\}, \quad (2)$$

where A and T are the same as defined in FSS and P is the vector related to the path of the URL, since all links of a specific site have the same domain.

Each feature context can be represented as a vector of terms with a specific weight. The weight w of term t can be defined as:

$$wt,d=1+logtft,d, \quad (3)$$

where tft,d denotes the frequency of term t appears in document d, and d can be U, P, A, or T. We use term frequency (TF) as feature weight for its simplicity and our experience shows that TF works well for our application.

To automatically construct FSS and FSL online, we use the following feature selection method using top-k features:

- When computing a feature set for P, A, and T, words are first stemmed after removing stop words. Then the top-k most frequent terms are selected as the feature set.
- When constructing a feature set for U, a partition method based on term frequency is used to process URLs, because URLs are not well structured.

Firstly, the top-level domain of URL (e.g. com, co.uk) is excluded. Secondly, after stemming terms, the most frequent k terms are selected from the URL features. Thirdly, the frequent set appears as a substring of the URL, the URL is split by the frequent term. For example, "abebooks" and "carbuyer" are terms that appear in URL of the book and auto domains, as "book" and "car" are frequent terms, the URL is split into "abe", "book" and "car", "buyer", respectively.

B. Adaptive Learning

Crawler has an adaptive learning strategy that updates and leverages information collected successfully during crawling. As shown in Figure 2, both Site Ranker and Link Ranker are controlled by adaptive learners. Periodically, FSS and FSL are adaptively updated to reflect new patterns found during crawling .As a result, Site Ranker and Link Ranker are updated. Finally, Site Ranker re-ranks sites in Site Frontier and Link Ranker updates the relevance of links in Link Frontier. 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. In the implementation, the learning thresholds are 50 new deep websites or 100 searchable 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.

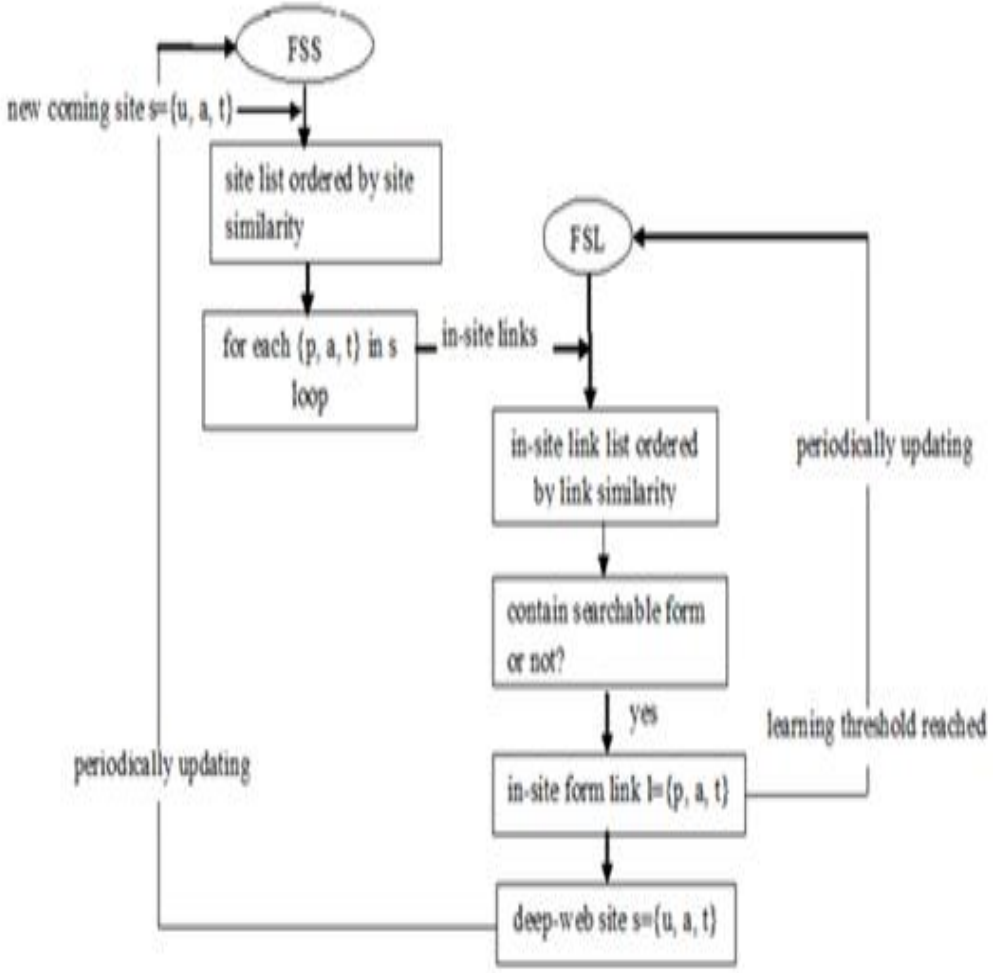

Figure 2: Adaptive Crawler

VI.CONCLUSION

This paper proposed the semantic focused crawling mechanism with extended deep we harvesting frame work. It achieves both high efficiency and broad coverage. Web Crawler is a focused crawler, which consists of two stages such as Site locating and In-site Exploring. Web Crawler minimizes the number of visited URLs and simultaneously maximizes the number of deep websites. In future work, we have a tendency to conceive to mix pre-query and post-query approaches for classifying deepweb forms to additional improve the accuracy of the shape classifier.

REFERENCES

[1] Feng Zhao, Chang Nie, Heqing, Hai Jin, "SmartCrawler: A Two-stage Crawlers for Efficiently Harvesting Deep-Web Interfaces", IEEE TransactionsOn Services Computing, vol. 9, no. 4, July/August. 2016.

[2] Lucianco Barbosa and Juliana Freire, "Searching for Hidden-Web Databases", In WebDB, pages 1-6, 2005.

[3] Lucianco 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.

[4] Lucianco 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.

[5] Kevin Chen-ChuanChang, Bin He, and Zhen Zhang, "Toward large scale integration: Building a metaquerier over databases on the web", In CIDR, pages 44–55, 2005.

[6] 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.

[7] 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.

[8] SoumenChakrabarti, Martin Van den Berg, and Byron Dom, "Focused crawling: a new approach to topic-specic web resource discovery", Computer Networks,31(11):1623–1640, 1999.

[9] Peter Lyman and Hal R. Varian,"How much information? 2003", Technical report,UC Berkeley, 2003.

[10] Roger E. Bohn and James E. Short. "How much information? 2009 report on american consumers", Technical report ,University of California ,San Diego,2009.

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

[12] 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, pages 355–364. ACM, 2013.