ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 4, Issue 12, December 2016

Optimized Search Harvesting Using Two Stage Intelligent Focus Crawler

Revati Rajane

M.E Student, Dept. of Computer Engg, RMD Sinhgad School of Engineering, Savitribai Phule Pune University, Pune, Maharashtra, India

ABSTRACT: The rapid growth of the world-wide web poses unprecedented scaling challenges for general purposecrawlers and search engines. In this project, a new hypertext information managementsystem called an intelligent is described. The goal of a focused crawler is to selectively seek outpages that are relevant to a pre-defined set of topics Rather than collecting and indexing all accessiblehypertext documents to be able to answer all possible ad-hoc queries, a smart crawler analyzesits crawl boundary to find the links that are likely to be most relevant for the crawl. It avoids irrelevantregions of the web. This leads to significant savings in hardware and network resources, andhelps keep the crawl more up-to-date. For this purpose it uses Reverse searching strategy.
A two-stage framework, namely intelligent, for efficient harvesting deep web interfaces. In the firststage, intelligent 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,intelligent ranks websites to prioritize highly relevant ones for a given topic. In the second stage,intelligent achieves fast in-site searching by excavating most relevant links.HTML and JavaScriptparser is developed to deal with dynamic pages. Moreover a report on crawled URLs is publishedafter crawling which gives entries of all crawled URLs and Errors found.

KEYWORDS: intelligent Crawler, focused crawler, weight table, World-Wide Web, Search Engine, links ranking

I. INTRODUCTION

Web crawler is defined as an automated program that methodically scans through Internet pagesand downloads any page that can be reached via links. With the exponential growth of the Web,fetching information about a special-topic is gaining importance. The objective of crawling is toquickly and efficiently gather as many useful web pages as possible, together with the link structurethat interconnects them. A Web crawler is a computer program that browses the World Wide Webin a methodical, automated manner or in an orderly fashion.
Intelligent focus Crawler is a topic oriented crawler which attempts to focus the crawling process onpages relevant to the topic. They keep the overall number of downloaded Web pages for processingto a minimum, while maximizing the percentage of relevant pages. The performance of an Intelligentdepends mostly on the richness of links in the specific topic being searched, and focused crawlingusually relies on a general web search engine for providing starting points.A two-stage framework, for efficient crawling is being proposed. In the first stage, intelligent focusCrawler performs site-based searching for center pages with the help of search engines, for whichan approach called as reverse searching. Intelligent focus Crawler ranks websites to prioritize highlyrelevant ones for a given topic. In the second stage, intelligent focus Crawler achieves fast in-sitesearching by excavating most relevant links. Focused crawlers work by combining both the contentof the retrieved Web pages and the link structure of the Web for assigning higher visiting priority topages with higher probability of being relevant to a given topic.
Web crawlers are programs that exploit the graph structure of the Web to move from page topage. The motivation is to design a web crawler that will be able to focus its web visits on particularpages that are of interest. The main problem is to overcome in the process of building afocused crawler lies on the effective identification of topic-relevant web pages and their exploitationin a way that will facilitate the crawler's decision making policy. The motivation for focusedcrawling comes from the poor performance of general-purpose search engines, which depend onthe results of generic web crawlers. Moreover, the focused crawler output, a domain oriented list ofweb sites.There is a need for an efficient crawler that is able to accurately and quickly explore the deep webdatabases based on specific domain. For

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 4, Issue 12, December 2016

example, search services use web crawlers to populatetheir indices, comparison shopping engines use them to collect product and pricing informationfrom online vendors, and the Internet Archive uses them to record a history of the Internet.

II. RELATED WORK

Smart Crawler: A Two-stage Crawler for Efficiently Harvesting Deep-Web Interfaces [1].This paper proposes Coverage for deep web interfaces and maintains highly efficient crawling. Forachieving the given objective it uses two different techniques namely site locating and in site exploring. A focused crawler is the web crawler that tries to download the pages that are related toeach other. The relevancy in data decides the performance of crawler. This technique eliminate theproblem of irrelevant results and hence most accurate results can be found. Study of WebCrawler and its Different Types [2], given paper studies about various types of crawler. Those can be listed as Focused Web Crawler, IncrementalCrawler, Distributed Crawler and Parallel Crawler. Study about Focus Crawler shows that it doesnot waste resources on irrelevant material since it concentrate on a specific topic. This paper showsthat it not have dependency on previous crawler. By using technique called "Surfacing" it achieveshigh coverage with just a small number of queries.Web Crawling Algorithms [3],this paper gives detailed study of various algorithms those can be used as a searching algorithmin crawling process. These algorithms includes Breadth First Search, Best First Search and FishSearch. These algorithm can work efficiently in static as well as less dynamic environments. Apartfrom searching algorithms Page Rank algorithm and Batch Page Rank algorithm are proposed inthe given paper. These algorithms are useful to reduce the network traffic and crawling costs.Novel Architecture of Web Crawler for URL Distribution [4] this paper gives an approach for balancing the load of URLs to improve the performance.URLs canbe distributed for balancing the load on web crawler. For achieving this goal, it uses the techniqueof load balancing. Moreover Paper describes about Different types of Web crawler and the policiesused in the web crawlers.
Domain-Specific Web Site Identification: The CROSSMARC Focused Web Crawler [5], this paperpresents a concept of crawler which uses various techniques like Site navigation, Page-filtering, Link-scoring. These techniques enhances the accuracy of resulting URLs. Present the extraction results according to the user's personal references and constraints. This king of crawling is also defined as Goal Directed Crawling.

III. EXISTING METHODOLOGY

A. *DESCRIPTION SYSTEM ARCHITECTURE:*

The number of Web pages is increasing in a very fast rate. This growth has urged the developmentof retrieval tools like search engines to get the information from WWW. Web crawling is oneof main component in Web information retrieval. Web crawling is a program which traverses theWorldwide Web (WWW) in a methodically, automated manner to generate a copy of all the visitedpages for latter processing by a search engine. Due to limited bandwidth storage, and computationalresources, and to the dynamic nature of the web, search engines cannot index every Web page, andeven the covered portion of the web cannot be monitored continuously for changes. In fact, a recentestimate of the visible Web is at around 9.2 billion static pages as of March 2007. This estimateis more than triple the 2 billion pages that the largest search engine, Google, reports at its Website. Therefore, it is essential to develop effective agents to conduct real time searches for users.Topic specific crawlers have become important tools to support applications such as specializedWeb portals, online searching, and competitive intelligence. These crawlers are designed to retrievepages that are relevant to the triggering topic. Generally employing single crawler to gather all pagesis inevitably difficult. Therefore, many search engines often run multiple processes in parallel toperform the task. This type of spider is referred as a parallel spider. This approach can considerablyimprove the collection efficiency.

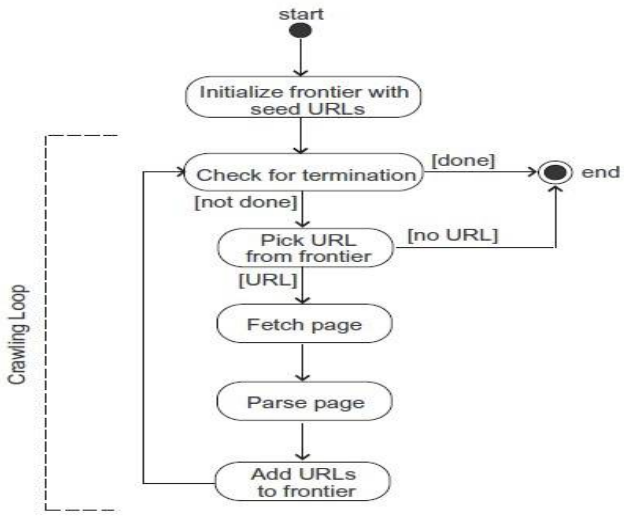
ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 4, Issue 12, December 2016



1.1 System Architecture

Figure 1.1, shows the flow of a basic sequential crawler .The crawler maintains a list of unvisitedURLs called the frontier. The list is initialized with seed URLs, which may be provided by a useror another program. Each crawling loop involves picking the next URL to crawl from the frontier,fetching the page corresponding to the URL through HTTP, parsing the retrieved page to extractthe URLs and application-specific information, and finally adding the unvisited URLs to the frontier.Before the URLs are added to the frontier they may be assigned a score that represents theestimated benefit of visiting the page corresponding to the URL. The crawling process may be terminatedwhen a certain number of pages have been crawled. If the crawler is ready to crawl anotherpage and the frontier is empty, the situation signals a dead-end for the crawler. The crawler has nois seen as a large graph with pages at its nodes and hyperlinks as its edges. A crawler starts at afew of the nodes (seeds) and then follows the edges to reach other nodes. The process of fetchinga topic. Page and extracting the links within it is analogous to expanding a node in graph search.A topical crawler tries to follow edges that are expected to lead to portions of the graph that are relevant to a topic.

Frontier:

The frontier is the to-do list of a crawler that contains the URLs of unvisited pages. In graph searchterminology the frontier is an open list of unexpanded (unvisited) nodes. Although it may be necessaryto store the frontier on disk for large -scale crawlers, the frontier will be represented as anin-memory data structure for simplicity. Based on the available memory, one can decide the maximumsize of the frontier. Because of the large amount of memory available on PCs today, a frontiersize of a 100,000 URLs or more is not exceptional. Given a maximum frontier size, there is a needof a mechanism to decide which URLs to ignore when this limit is reached. Note that the frontiercan fill rather quickly as pages are crawled. One can expect around 60,000 URLs in the frontierwith a crawl of 10,000 pages, assuming an average of about 7 links per page. The frontier may be implemented as a FIFO queue, in which case a breadth-first crawler thatcan be used to blindly crawl the Web is being used. The URL to crawl next comes from the head ofthe queue, and the new URLs are added to the tail of the queue. Because of the limited size of thefrontier, Make sure that do not add duplicate URLs into the frontier. A linear search to find out if anewly extracted URL is already in

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 4, Issue 12, December 2016

the frontier is costly. One solution is to allocate some amount ofavailable memory to maintain a separate hash-table (with URL as key) to store each of the frontierURLs for fast lookup. The hash-table must be kept synchronized with the actual frontier. A moretime -consuming alternative is to maintain the frontier itself as a hash-table (again with URL askey). This would provide fast lookup for avoiding duplicate URLs. However, each time the crawlerneeds a URL to crawl, it would need to search and pick the URL with the earliest time stamp (thetime when a URL was added to the frontier). If memory is less of an issue than speed, the firstsolution may be preferred. Once the frontier reaches its maximum size, the breadth-first crawlercan add only one unvisited URL from each new page crawled. If the frontier is implemented asa priority queue .A preferential crawler, which is also known as a best-first crawler is used. Thepriority queue may be a dynamic crawling the Web 157 array that is always kept sorted by the estimatedscore of unvisited URLs. At each step, the best URL is picked from the head of the queue.Once the corresponding page is fetched, the URLs are extracted from it and scored based on someheuristic. They are then added to the frontier in such a manner that the order of the priority queueis maintained. Duplicate URLs in the frontier can be avoided by keeping a separate hash-table forlookup. Once the frontiers maximum size (MAX) is exceeded, only the best MAX URLs are keptin the frontier.

Disadvantage of Existing System

If the crawler finds the frontier empty when it needs the next URL to crawl, the crawling processcomes to a halt. With a large value of MAX and several seed URLs the frontier will rarely reach theempty state. At times, a crawler may encounter a spider trap that leads it to a large number of differentURLs that refer to the same page. One way to alleviate this problem is by limiting the numberof pages that the crawler accesses from a given domain. The code associated with the frontier canmake sure that every consecutive sequence of URLs (say 10) URLs, picked by the crawler, containsonly one URL from a fully qualified host name (e.g., www.cnn.com). As side effects, the crawler is polite by not accessing the same Web site too often, and the crawled pages tend to be more diverse.

IV. PROPOSED SOLUTION

Site locating and in-site exploring, as shown in figure 1.2, the first site locating stage finds the mostrelevant site for a given topic, and then the second in-site exploring stage uncovers searchable formsfrom 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 intelligent Focus Crawlerto start crawling, which begins by followingURLs from chosen seed sites to explore other pages and other domains. When the number ofunvisited URLs in the database is less than a threshold during the crawling process, intelligent Focus Crawlerperforms reverse searching of known deep web sites for canter pages (highly ranked pages thathave many links to other domains) and feeds these pages back to the site database. Site Frontierfetches homepage URLs from the site database, which are ranked by Site Ranker to prioritize highlyrelevant sites. The Site Ranker is improved during crawling by an Adaptive Site Learner, whichadaptively learns from features of deep-web sites (web sites containing one or more searchableforms) 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 explorationfor 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 inCandidate Frontier, intelligent Focus Crawlerranks them with Link Ranker. Note that site locating stage andin-site exploring stage are mutually intertwined. When the crawler discovers a new site, the sites URL is inserted into the Site Database. The Link Ranker is adaptively improved by an AdaptiveLink Learner, which learns from the URL path leading to relevant forms.

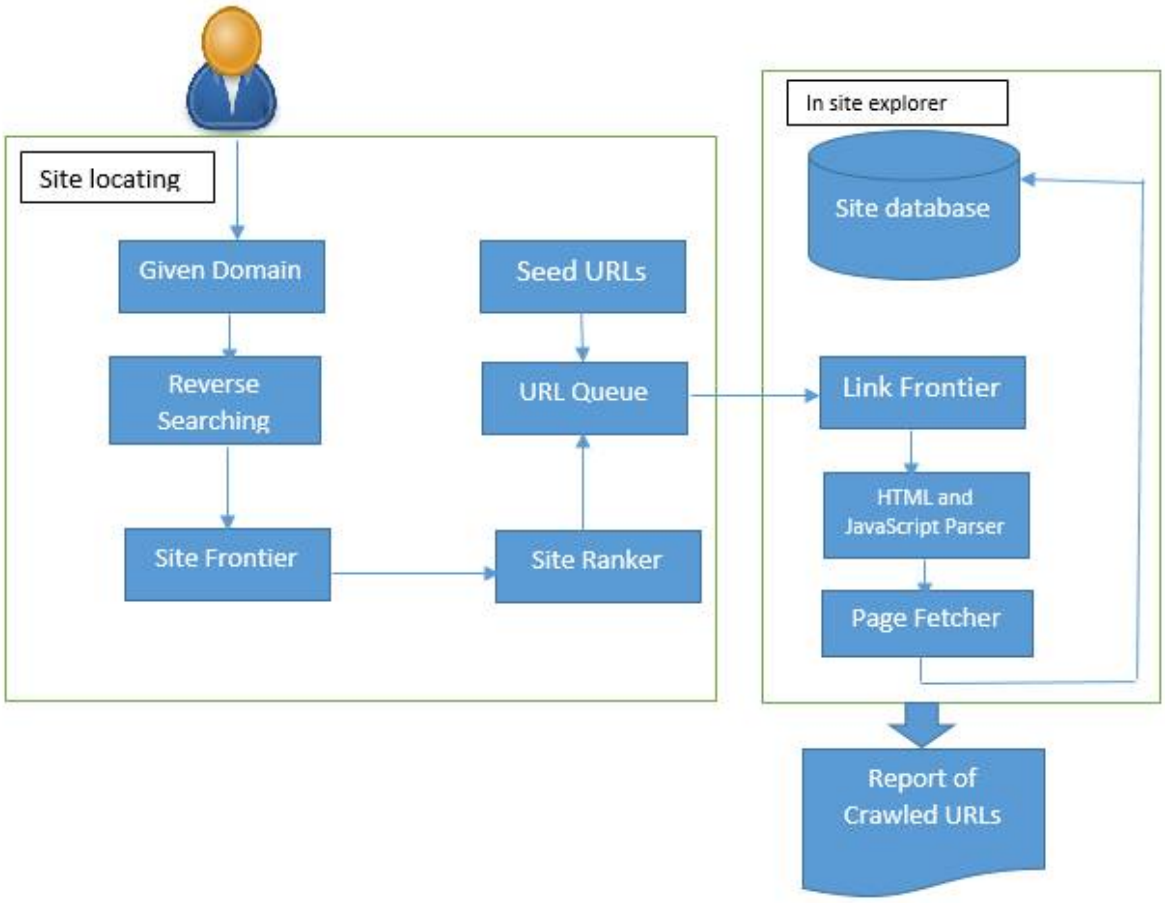
ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 4, Issue 12, December 2016



1.2 Proposed System Architecture

Site Locating

The site locating stage finds relevant sites for a given topic, consisting of site collecting, site ranking,and site classification. Site collectingthe traditional crawler follows all newly found links. In contrast, intelligent Focus Crawler strivesto minimize the number of visited URLs, and at the same time maximizes the number of deepwebsites. To achieve these goals, using the links in downloaded webpages is not enough. Thisis because a website usually contains a small number of links to other sites, even for some largesites. For instance, only 11 out of 259 links from webpages of aaronbooks.com pointing to other sites; amazon.com contains 54 such links out of a total of 500 links (many of them are differentlanguage versions, e.g., amazon.de). Thus, finding out-of-site links from visited webpages may notbe enough for the Site Frontier. In fact, the experiment in Section 5.3 shows that the size of SiteFrontier may decrease to zero for some sparse domains. To address the above problem, in givenproject, two crawling strategies, reverse searching and incremental two-level site prioritizing, to findmore sites.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 4, Issue 12, December 2016

Reverse searching

The idea is to exploit existing search engines, such as Google, Baidu, and Bing etc., to find center pagesof unvisited sites. This is possible because search engines rank webpages of a site and center pagestend to have high ranking values. Algorithm 1 describes the process of reverse searching. A reversesearch is triggered:
1. When the crawler bootstraps.
2. When the size of site frontier decreases to a predefined threshold.
For applying this strategy randomly pick a known deep website or a seed site and use general searchengines facility to find center pages and other relevant sites, Such as Google's link: , Bings site:,Baidus domain:. For instance, [link: www.google.com] will list web pages that have links pointingto the Google home page. In the proposed system, the result page from the search engine is firstparsed to extract links.

Incremental site prioritizing

To make crawling process resumable and achieve broad coverage on websites, an incremental siteprioritizing strategy is proposed. The idea is to record learned patterns of deep web sites and formpaths for incremental crawling. First, the prior knowledge (information obtained during past crawling, such as deep websites, links with searchable forms, etc.) is used for initializing Site Ranker andLink Ranker. Then, unvisited sites are assigned to Site Frontier and are prioritized by Site Ranker,and visited sites are added to fetched site list. The detailed incremental site prioritizing processis described in Algorithm 2. While crawling, intelligent Focus Crawlerfollows the out-offsite links of relevantsites. To accurately classify out of- site links, Site Frontier utilizes two queues to save unvisited sites.The high priority queue is for Fig. out-of-site links that are classified as relevant by Site Classifierand are judged by Form Classifier to contain searchable forms. The low priority queue is for out-offsitelinks that only judged as relevant by Site Classifier. For each level, Site Ranker assigns relevantscores for prioritizing sites. The low priority queue is used to provide more candidate sites. Oncethe high priority queue is empty, sites in the low priority queue are pushed into it progressively.

V. PSEUDO CODE

Reverse Searching From

```
input : seed sites and Domain specific topic
output : relevant sites while of candidate sites less than a threshold do
// pick a deep website;
site getDeepWebSite(siteDatabase, seedSites);
resultPage reverseSearch(site);
links extractLinks(resultPage);
for each link in links do
page downloadPage(link);
relevant classify(page);
if relevant then
relevantSites extractUnvisitedSite(page)
Output relevantSites;
end
end
```

In site exploration

```
input : siteFrontier
output : searchable forms and out-of-site links
BFS (topic, starting-urls)
foreach link (starting-urls)
enqueue(frontier, link, 1);
End For
```

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 4, Issue 12, December 2016

while (visited ¡ MAX-PAGES)
link := dequeuetop □ link(frontier);
doc := fetch(link);
score := sim(topic; doc);
enqueue(frontier; extract □ links(doc); score);
if (frontier > MAX BUFFER)Then
dequeue ->bottom -> links(frontier);
endif
endwhile

HTML parsing

After retrieving HTML code, it has to be parsed. LibCURL use cstring to store data, to not wastetime to copy it into a string, the program use cstring functions directly on the Crawler structure: Thealgorithm search every occurrences of href tags (href=) in the cstring, then chars are stored until thefirst non URL char: ; ‘ ; ¡ ; ¿ and space. The ? and chars are added to this list to not store anchorsand to ignore cookies set for a same page. To be the most efficient, a regular expression could havebeen used., but no function provide regex searches in string nor in string.h, so another library is beenused. With my way, links who are not included in href tags of which are using a non-strict syntax(like with simple quotations, of with spaces before or after the equal) will just not be crawled.

URL parsing

The parsed URL are stored in C strings. They are parsed in turn, removing http:// part, useless tostore, and eventually the “www.” part. URL parts, bounded by / chars, are then stored in the tree. For that purpose string manipulation functions used.URL parts are stored at the same time than URL isparsed, using a vector of strings
Storing parsed URLsthe program uses a vector iterator to browse the different parts stored in the parsed URL. Each partis the parent of the deeper one, using the tree function To avoid the problem of duplicated elements,a temporary tree for each URL is been chosen, and then to use the merge function, with the optionduplicate leaves to false. Using this pre-made function was the simplest and the most optimized way.

Recursive browsing

The URL are parsed and stored according to the process of HTML parsing. To be strictly recursive,it would have been required to retrieve and parse HTML data at every new URL found in a page, butit would also have caused memory problems, because of HTML data accumulation in memory. Mysolution is to make a first iteration to store URLs in the URL the user entered, and after to browsethe tree, reconstructing URLs and crawling those one. It’s working because the new elements storedin the tree are added at the end (like in a list), so the tree iterator will point new URLs at the end.The sort function is used only when the crawl is finished, to be able to display the tree or searchresults in alphabetical sorting. Every treatments are so included.

VI. CONCLUSION

Proposed System had demonstrated that goal-directed crawling is a powerful means for topicalresource discovery. The focused crawler is a system that learns the specialization from examples,and then explores the Web, guided by a relevance and popularity rating mechanism. Proposedsystem selects work very carefully from the crawl frontier. A consequence of the resulting efficiencyis that it is feasible to crawl to a greater depth (along link chains) than would otherwise be possible.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 4, Issue 12, December 2016

VII. FUTURE WORK

The future work is going to be carried out regarding to the scale problem of this allocation algorithm. Since cloud is although the initial results are encouraging, there is still a lot of work to do for improving the crawling efficiency. A major open issue for future work is to do more extensive test with large volume of web pages. Future work also includes code optimization and url queue optimization, because crawler efficiency is not only depends to retrieve maximum number of relevant pages but also to finish the operation as soon as possible.

VIII. ACKNOWLEDGMENT

I take this opportunity to express my heartfelt gratitudeto my guide Prof. Pradnya Kasture and head of department, Prof.Vina M Lomte, Department of Computer Engineering, RMDSSOE, SavitribaiPhule Pune University, for her kind cooperation, constant Encouragement and suggestions and capable guidance duringthe research, without which it would have been difficult to proceed with.

REFERENCES

[1] Feng Zhao, Jingyu Zhou, Chang Nie, Heqing Huang, Hai Jin.Smart Crawler: A Two-stageCrawler for Efficiently Harvesting Deep-Web Interfaces, IEEE 2016.
[2] Y. He, D. Xin, V. Ganti, S. Rajaraman, and N. Shah, Crawling deep web entity pages, in Proc.6th ACM Int. Conf. Web Search Data Mining, 2013, pp. 355364.
[3] (2009). Clustys searchable database dirctory [Online]. Available: http://www.clusty.com/
[4] S. Denis, On building a search interface discovery system, in Proc. 2nd Int. Conf. Resource
Discovery, 2010, pp. 8193.
[5] L. Barbosa and J. Freire, An adaptive crawler for locating hiddenweb entry points, in Proc.16th Int. Conf. World Wide Web, 2007, pp. 441450.
[6] S. Chakrabarti, M. V. den Berg, and B. Dom, Focused crawling: A new approach totopic-specific web resource discovery, Comput. Netw., vol. 31, no. 11, pp. 16231640, 1999.
[7] M. E. Dincturk, G. vincent Jourdan, G. V. Bochmann, and I. V. Onut, A model-based approachfor crawling rich internet applications, ACM Trans. Web, vol. 8, no. 3, pp. Article 19, 139,
[8] J. Cope, N. Craswell, and D. Hawking, Automated discovery of search interfaces on the web,in Proc. 14th Australasian Database Conf.-Volume 17, 2003, pp. 181189.
[9] D. Susan and C. Hao, Hierarchical classification ofWeb content, in Proc. 23rd Annu. Int.ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2000, pp. 256263.
[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, The weka datamining software: An update, SIGKDD Explorations Newsletter, vol. 11, no. 1, pp. 1018, Nov.2009.
[11] (2013). Open directory project [Online]. Available: http://www. dmoz.org/
[12] Hai Dong, Member, IEEE, and Farookh Khadeer Hussain. Self-:Adaptive Semantic FocusedCrawler for Mining Services Information Discovery IEEE, VOL. 10, NO. 2, MAY 2014
[13] Joel Acevedo, Giselle Agosto, Maria Ortiz de Zuniga, Professor Morley Mao.Web Crawler ,University of Michigan EECS 489.
[14] Domain-Specific Web Site Identification:The CROSSMARC Focused Web Crawler.
[15] Md. Faizan Farooqui, Dr. Md. Rizwan Beg and Dr. Md. Qasim Rafiq. An Extended ModelFor Effective Migrating Parallel Web Crawling With Domain Specific And IncrementalCrawling., IJWSC, Vol.3, No.3, September 2012.

BIOGRAPHY

Revati Rajaneis a Student in the Computer Engineering Department, RMD Sinhgad School of Engineering,Warje, Pune University. She is pursuing Master of Computer engineering degree in. Her research interests are Information Retrieval, Data mining, Algorithms, etc.