IFERP
connecting engineers... developing research

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 4, April 2018

A Framework Designing For RDF Smart Crawler for Extracting Semantic Information

[1] Gurdas Singh, [2] Brijesh Bakariya
[1] Research Scholar, IKGPTU, Kapurthala, Punjab,India
[2] Asst. Prof., IKGPTU Campus, Hoshiarpur, Punjab, India

Abstract: - **Currently Search engines only provide URL links for search queries. Crawling strategy adopted by most search engines only search on html keywords and index the pages, but semantic web retains most rich information in RDF files and crawlers don't index the RDF. In this work, we deal with problem and design a smart crawler which can retrieve semantic information for keyword queries. In addition to retrieving the information, the proposed solution also focus on ranking the semantic information. Ordinarily, a covetous framework is used to pick the terms that enlarge the new returns each cost unit. We comprehended that not each record is square with while selecting the request to cover them. Broad reports can be secured by various requests, paying little respect to how the inquiries are picked. In like manner the criticalness of a record is then again with respect to its size. Our further examination in this issue finds that the noteworthiness of the record depends not simply on the amount of the terms it contains, furthermore the sizes of those terms.**

***Keywords*: Crawling, Query analysis, RDF, semantic approach and smart crawler**

I. INTRODUCTION

The deep (or hidden) web refers to the contents lie behind searchable web interfaces that cannot be indexed by searching engines. Semantic web falls under the category of deep web as search engines don't index the semantic web contents. To locate deep web contents, earlier two types of crawler's generic crawler and focused crawler were proposed. Generic crawlers [1], [2], [3], [4], [5] fetch all searchable forms and cannot focus on a specific topic. Focused crawlers such as Form-Focused Crawler (FFC) [6] and Adaptive Crawler for Hidden-web Entries (ACHE) [7] 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 [8]. Both of these crawlers focused on extracting Forms and could not work for semantic web information. One another challenge is ranking the search results. If the semantic results can be ranked and displayed, it would reduce the time for retrieving most important information to the users. This we refer to as relevance. Web contents can be ranked based on hit count and relative word occurring frequency but semantic information ranking is a totally new area and no other previous works on it exist.
In this paper, we propose an adaptive smart crawling algorithm to extract semantic information from internet. Also the improvement issue, request assurance has similarly been shown as bolster learning issue. In this model, a crawler and an objective data source are considered as an administrators and nature independently. By then its assurance framework will be quickly adjusted by learning past addressing results and makes note of at most two-phase long two-stage remunerate.

II. LITERATUR REVIEW

Brin and Page's 1998 paper plotting the engineering of the original Google [9] framework contains a short depiction of their crawler. The first Google crawling framework comprised of a solitary URL server prepare that kept up the condition of the creep, and around four searching forms that downloaded pages. Both URL server and crawlers were executed in Python. The searching procedure utilized non-concurrent I/O and would commonly perform around 300 downloads in parallel. The pinnacle download rate was around 100 pages for every second, with a normal size of 6 KB for each page. Brin and Page distinguished social parts of searching (e.g., managing website admins' complaints) as a noteworthy test in working a crawling framework. Recently, Yan et al. described IRLbot [10], a single-process web crawler that is able to scale to extremely large web collections without performance degradation. IRLbot features a "seen-URL" data structure that uses only a fixed amount of main memory, and whose performance does not degrade as it grows. The paper

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 4, April 2018

describes a crawl that ran over two months and downloaded about 6.4 billion web pages. In addition, the authors address the issue of crawler traps, and propose ways to ameliorate the impact of such sites on the crawling process. Finally, there are various open-source crawlers, two of which merit extraordinary specify. Heritrix [11] is the crawler utilized by the Internet Archive. It is composed in Java and exceedingly componentized, and its outline is very like that of Mercator. Heritrix is multithreaded, however not circulated, and all things considered appropriate for directing reasonably measured creeps. The Nutch crawler [12] is composed in Java also. It underpins conveyed operation and ought to along these lines be appropriate for extensive crawlers; however as of the composition of [13] it has not been scaled past 100 million pages. Below is the explanation of existing model for information extraction from smart web crawler is shown fig. 1.

A. The crawler/Spider module
Web searchers use web crawler to gather data for ordering the pages; Crawlers are the robotized projects that take after the associations found on the webpage pages. The program i.e. Web Explorer, sends HTTP requests (hypertext transfer protocol), the most generally perceived protocol on the web which is used to recuperate the webpage pages and to download and uncover to them on the customer's service end.

B. The repository/database module
The repository or database has an unlimited amassing of data things. Each site page recouped by the crawler is pressed and after that set away in the storage facility with an intriguing ID associated with the URL and a note is taken of the length of each page [5].

C. The adaptive link analysis module
The information is accessible in the database in sweeping aggregate so the information of site pages is to be secured in the most critical demand. It note worthily affects web look as indexer takes a gathering of data or reports and makes a searchable record. There could be different records in light of the substance of the pages so that the crawler can record the information required by the customer.

D. The retrieval/ranking module
The recovery means to find the records related with the request term. It determines the scores for the reports using a positioning figuring. This module is the inside fragment of any web searcher. Page positioning procedures are associated, which plan the reports out and out of their hugeness, essentialness and rank score orchestrate the site page [8]. Page rank estimation allots numerical weight to hyperlinked reports recorded by a web engine.

E. The user query interface
The customer enters a request related to the information required by the customer to the graphical customer interface
gave by the web list. Most web interfaces are amazingly essential; applications may use structures to make the customer display a query.
Issues related to motivations behind the proposed work are as follows:
- Current search engines only fetch relevant pages for search query, but we propose to search relevant RDF for the search query.
- As of now there is no real way to execute an inquiry and get semantic data connected with question. Prominent web crawlers just give website pages joins connected with question.
- There is no semantic crawler which record semantic data like RDF and OWL and after that gives it as result to search query.

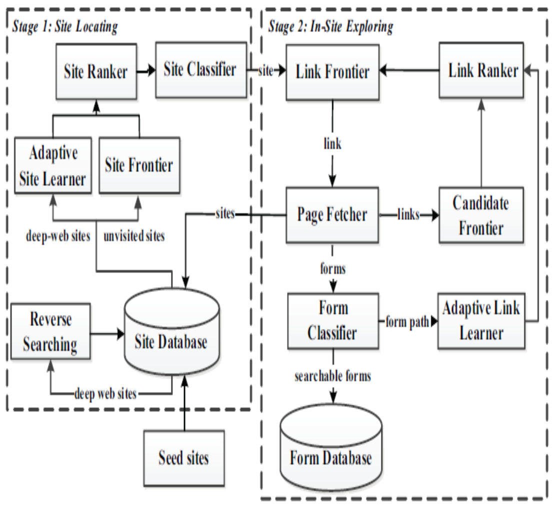


Fig.1. Architecture of existing Smart web crawler

III. PROBLEM STATEMENT

RDF store rich semantic information about contents and it is distributed on deep web interfaces. Given a search query, the RDF pertaining to that search query must be fetched so that in the search result the coverage and versatility of information must be good.
1) RDF: From the page got the savvy semantic web crawler includes semantic classifier that implies it brings the subjective pattern through RDF (Resource Description Framework) as rdf: sort. For instance in the event that we bring a search review identified with home then just links named with home are fetched.

2) OWNL: From the page got the savvy semantic web crawler includes semantic classifier that implies it gets the semantic outline for subjects through OWNL as patent sort to specific subject. For instance in the event that we bring a pursuit question identified with home then links named with home are brought as well as the semantic expressions of home related links are likewise fetched.
Following major problems are formulated from above discussion of related work section. The Semantic Web is useful as long as an application can access and merge any webpage due to following reasons:
• The data can be published anywhere, we cannot find all the data to answer a query
• People don't know the schema of each data source so that we cannot send a precise query to a specific RDF data source as we use SQL to query relational databases.
• The answer to the client queries should include not only the explicit information represented in RDF data but also the implicit information which can be got through data inference.

IV. PROPOSED ARCHITECTURE DESIGN

We propose a smart semantic crawler which crawls and index semantic information for search queries. The architecture of the proposed solution is given below. The solution consist of two parts Site Locating and In site Exploring.

A. Module 1: Site Locating

The main aim of Site Locating is find most relevant sites for given topic. Seed sites must be preconfigured and added to site database. Venerate Searching creeps seed locales in the website database and fined as much links with profound pages and adds all destinations found to webpage database.

a. Working of Reverse Searching

We randomly pick a known profound site or a seed site and utilize general web crawler's office to find focus pages and other significant locales, Such as Google's "link:" , Bing's "website:", Baidu's "space:". For example, [link: www.google.com] will list site pages that have joins

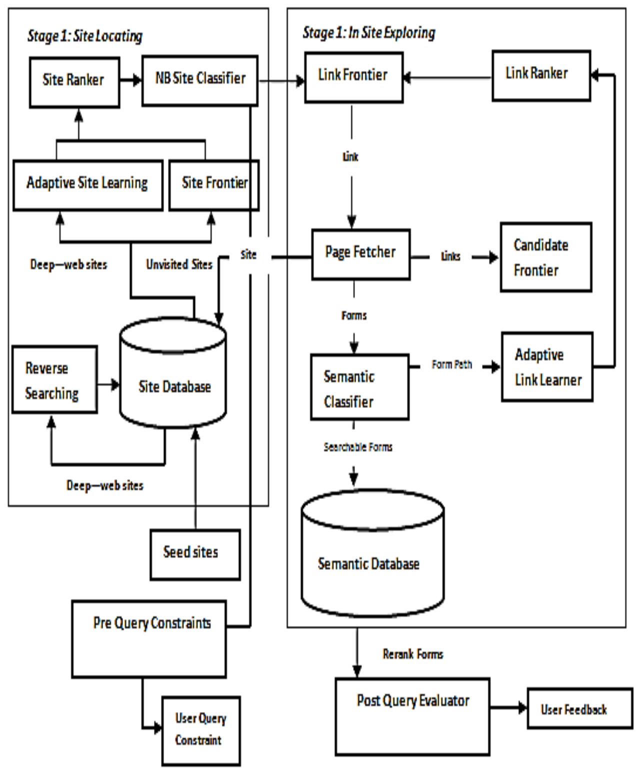


Fig. 2. Architecture of proposed solution

indicating the Google landing page. In our framework, the outcome page from the internet searcher is first parsed to concentrate joins. At that point these pages are downloaded and dissected to choose whether the links are applicable or not utilizing the accompanying heuristic rules: –
• If the page contains related searchable structures, it is significant.
• If the quantity of seed locales or brought profound sites in the page is bigger than a client defined limit, the page is important.
• Finally, the discovered significant links are added to site database.

a. Working of Site Frontier

Site Frontier peruses the links from the site database and gives to Site Ranker module. In the second stage, Smart Crawler accomplishes quick in-site looking by uncovering most important links with a versatile link positioning. To kill inclination on going to some exceedingly pertinent links in concealed web registries, we outline a link tree information structure to accomplish more extensive scope for a site

b. Working of Site Ranker

Site Ranker gets all url from Site Frontier positions site URLs to organize potential profound destinations of a given theme. To this end, two elements, site similitude and site recurrence, are considered for positioning. Webpage likeness measures the theme comparability between another website and known profound sites. Site recurrence is the recurrence of a site to show up in different destinations, which demonstrates the prevalence and power of the site — a high recurrence site is possibly more vital. Versatile Site learner discovers site similitude and site recurrence. In view of this site ranker figures the score of every sites.

c. Working of Site Classifier

Site Classifier decides the topical significance of a site in view of the substance of its landing page. At the point when another site originates from site ranker, the landing page substance of the site is separated and parsed by expelling stop words and stemming. At that point we develop a component vector for the site and the subsequent vector is nourished into a Naıve Bayes classifier to figure out whether the page is theme significant or not.

A. Module 2: Insight Exploring

The main aim of Insight Exploring is to find searchable forms. During the knowledge investigating stage, important links are organized for quick in-site looking. We have played out a broad execution assessment of Smart Crawler over genuine web information in 1representativedomains and contrasted and ACHE and a webpage based crawler.

a. Working of Link Frontier

Link Frontier gets each one of the associations from the Site Classifier and goes to Page fetcher. The link classifiers in these crawlers assume a vital part in accomplishing higher searching productivity than the best-first crawler However, these link classifiers are utilized to anticipate the separation to the page containing searchable structures, which is hard to appraise, particularly for the postponed advantage joins (interfaces in the end prompt pages with structures). Therefore, the crawler can be wastefully prompted pages without focused structures.

b. Working of Page Fetcher

Page fetcher read the website page from the url connect given by Link Fronties. We utilize Http to download the website pages. For links that only differ in the query string part, we consider them as similar URL. Since links are regularly appropriated unevenly in server catalogs, organizing joins by the importance can conceivably inclination toward a few indexes. For example, the links under books may be allocated a high need, since "book" is an essential element word in the URL.

c. Working of Semantic Classifier

From the page fetched, Semantic classifier discovers RDF and OWNL and groups them to important and immaterial Forms. HIFI system is received by Semantic Classifier. Semantic Classifier judges whether a shape is point significant or not in light of the content element of the RDF that comprises of space related terms. The methodology of parceling the element space permits choice of more successful learning calculations for every feature subset. The basic web crawling calculation is straightforward: Given an arrangement of seed Uniform Resource Locators (URLs), a crawler downloads all the pages tended to by the URLs, separates the hyperlinks contained in the pages, and iteratively downloads the website pages tended to by these hyperlinks.

V. PROPOSED ALGORITHM

A. Site Locating

When a search query is issued, Google search is invoked with the keyword of query to get the relevant sites. The relevant sites are stored in Sites Database. Reverse Search modules takes the sites from sites database and does reverse search on Google to get the pages where the sites are referred and populates those pages also into Site Database. Site Frontier extracts each link from Site Database and provides to Site Ranker for lookup on RDF content in it and ranks the Site based on the number of RDF located in the Site. The links on the Site where RDF are present is provided to the Link Frontier.

B. In Site Exploring

Link frontier traverses the link and extracts the pages from internet for that link. It then orders the Links based on the RDF content relevant to the search query and highly relevant links are provided Semantic Classifier. Semantic Classifier classifies the RDF present in the pages to three levels of High, Medium and Low relevance to the search query. The classified RDF is stored in the Semantic database. In the Post Query, users can give extra preference on the search query and based on the preferences the RDF is fetched from the Semantic database and the result is provided to the user.

The algorithm steps for Semantic crawling is below

Input: Keyword

Output: RDF pages

- Step1: Using the keyword, Google search is done and the output links are got.
- Step2: Each of the output links is taken and deep crawling on site is done to extract any RDF or OWL links.
- Step3: Conceptually similar RDF is grouped by using clustering algorithm (k means clustering is used).

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 4, April 2018

Each of clusters is displayed to user and user must select the cluster which is most relevant to his search.
<Clustering of RDF by using K means is a separate algorithm>

- Step4: From RDF cluster selected by user all subjects and objects are extracted. Using the dictionary, a concept keyword table is created by learning all synonyms for subjects and objects.
- Step5: By taking most occurring words from the concept keyword table, a new keyword is created and Google search is done with new keyword and from the resulting links, deep crawling is done to extract pages with RDF.
- Step6: The RDF concept distance similarity to selected RDF concept by user is measured, if the distance is less than a threshold i.e. 1000 taken in this project, then those RDF page is selected.

RDF similarity measurement is done by a separate algorithm named as DSFC (Domain Specific form classifier). This algorithm classifies the forms structure on the basis of topic domains.

- Step7: All the selected RDF pages are then displayed to user.

VI. IMPLEMENTATION PROCESS

The proposed semantic crawler was implemented in JAVA. Parameters to be used to implement and generate the outcomes of proposed solution are as follows:

1) Accuracy: Accuracy is measured as the number of relevant out of the total RDF fetched. Relevant is judged by human user after fetching of RDF.

Accuracy= total relevant pages/total RDF fetched pages

2) Information depth: Information depth is measured in terms of non-duplicated information present in RDF out of total RDF extracted.

Information depth= non-duplicated information in RDF/ Total RDF extracted

3) Harvest rate: Harvest rate is defined as number of RDF searched as per number of pages.

Harvest rate= Total RDF searched pages/Total number of pages

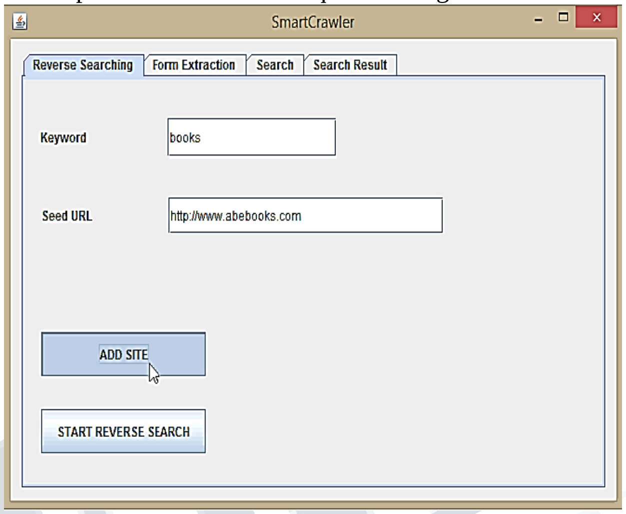
4) Running time: Running time is defined total time taken in running process to search as per number of RDFs.

Running time= total time of search process/ Total number of RDFs

5) Preference impact: Preference impact is defined as number of filtered forms as per number of matched forms

Preference impact= Total number of filtered forms/ total number of matched forms

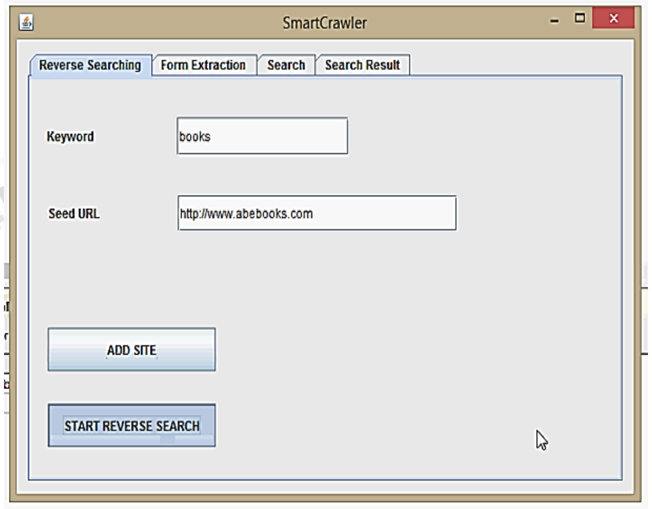
We tested for different keywords and the snapshots for whole process for one of the queries are given below:



Keyword: books
Seed URL: http://www.abebooks.com

Fig 3: GUI proposed solution

Fig.3 shows the GUI of proposed solution. The seed site for the query is added and after adding reverse search is launched**.**

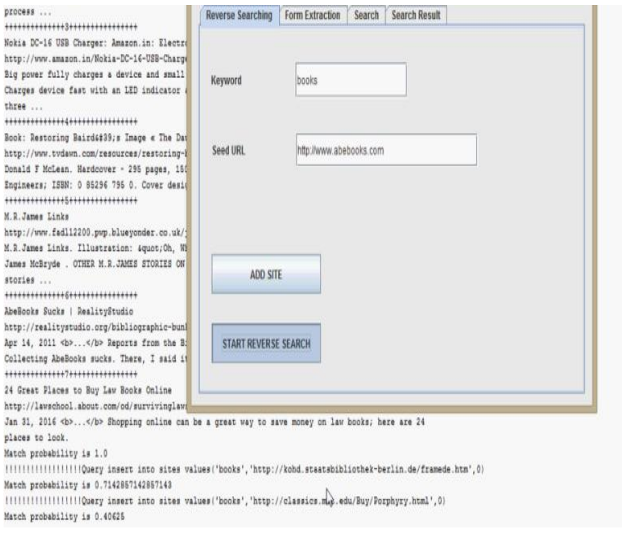


Keyword: books
Seed URL: http://www.abebooks.com

Fig 4: Reverse search process in proposed solution

Fig.4 shows the reverse search process that fetches more sites relevant to the keyword and the result is shown below in fig.4. Once reverse search is complete, crawling is done to fetch the Semantic pages that is presenting through fig.5**.**

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 4, April 2018

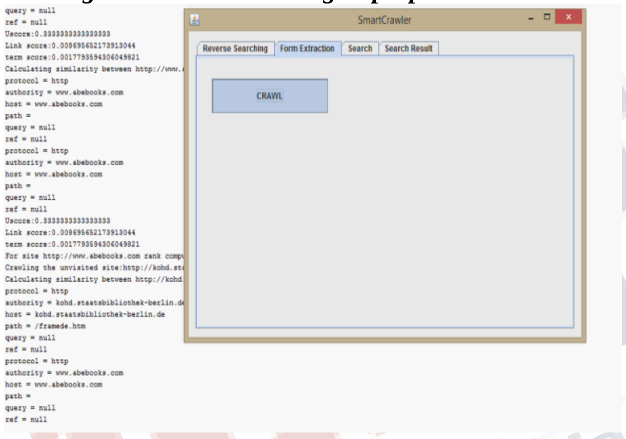


```
process ...
+++++++++++++3+++++++++++++++
Nokia DC-16 USB Charger: Amazon.in: Electr
http://www.amazon.in/Nokia-DC-16-USB-Charg
Big power fully charges a device and small
Charges device fast with an LED indicator
three ...
+++++++++++++4+++++++++++++++
Book: Restoring Baird&#39;s Image « The Da
http://www.tvdawn.com/resources/restoring-
Donald F McLean. Hardcover - 295 pages, 15
Engineers; ISBN: 0 85296 795 0. Cover desi
+++++++++++++5+++++++++++++++
M.R.James Links
http://www.fadl12200.pwp.blueyonder.co.uk/
M.R.James Links. Illustration: &quot;Oh, W
James McBryde . OTHER M.R.JAMES STORIES ON
stories ...
+++++++++++++6+++++++++++++++
AbeBooks Sucks | RealityStudio
http://realitystudio.org/bibliographic-bun
Apr 14, 2011 <b>...</b> Reports from the B
Collecting AbeBooks sucks. There, I said i
+++++++++++++7+++++++++++++++
24 Great Places to Buy Law Books Online
http://lawschool.about.com/od/survivinglaw
Jan 31, 2016 <b>...</b> Shopping online can be a great way to save money on law books; here are 24
places to look.
Match probability is 1.0
!!!!!!!!!!!!!!!!!!Query insert into sites values('books','http://kohd.staatsbibliothek-berlin.de/framede.htm',0)
Match probability is 0.7142857142857143
!!!!!!!!!!!!!!!!!!Query insert into sites values('books','http://classics.mit.edu/Buy/Porphyry.html',0)
Match probability is 0.40625
```

Keyword: books

Seed URL: http://www.abebooks.com

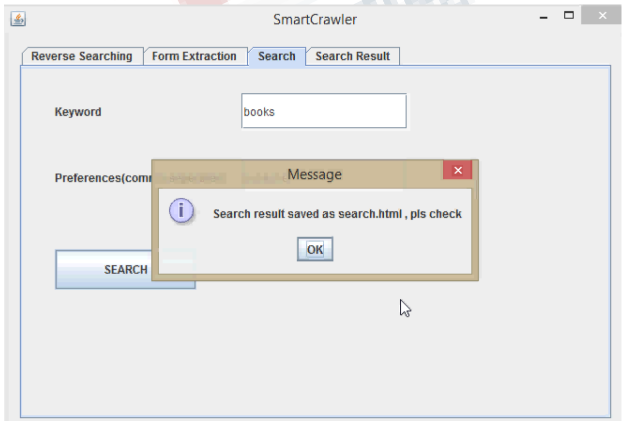
Fig 5: Semantic crawling in proposed solution



```
query = null
ref = null
Uscore:0.3333333333333333
Link score:0.008695652173913044
term score:0.0017793594306049821
Calculating similarity between http://www.
protocol = http
authority = www.abebooks.com
host = www.abebooks.com
path =
query = null
ref = null
protocol = http
authority = www.abebooks.com
host = www.abebooks.com
path =
query = null
ref = null
Uscore:0.3333333333333333
Link score:0.008695652173913044
term score:0.0017793594306049821
For site http://www.abebooks.com rank comp
Crawling the unvisited site:http://kohd.st
Calculating similarity between http://kohd
protocol = http
authority = kohd.staatsbibliothek-berlin.d
host = kohd.staatsbibliothek-berlin.de
path = /framede.htm
query = null
ref = null
protocol = http
authority = www.abebooks.com
host = www.abebooks.com
path =
query = null
ref = null
```

Fig 6: RDF extraction process in proposed solution

Fig.6 shows the complete crawling process using RDF extraction process. Crawling completes when threshold limit of RDF are extracted



Keyword: books

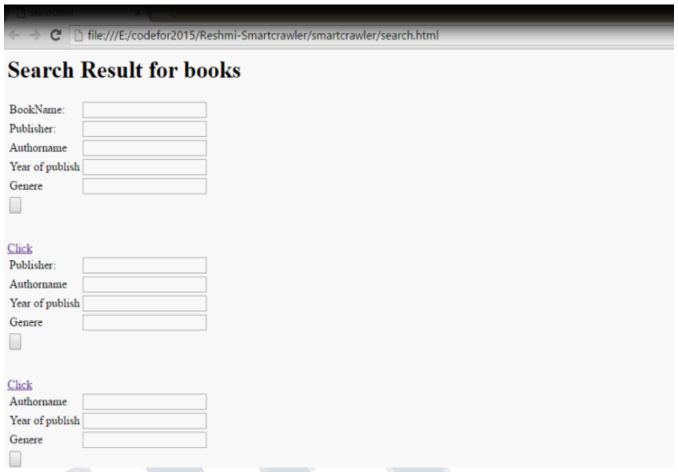
Preferences(com

Message

Search result saved as search.html , pls check

Fig 7: Completion of Search process

Fig.7 shows the results of searching process. The RDF results can be viewed using the form shown in fig.8.



file:///E:/codefor2015/Reshmi-Smartcrawler/smartcrawler/search.html

Search Result for books

BookName:
Publisher:
Authorname
Year of publish
Genere

Click
Publisher:
Authorname
Year of publish
Genere

Click
Authorname
Year of publish
Genere

Fig 8: Search results (RDF) for books

The solution was tested for accuracy and information depth of the proposed solution with Google Search.

VII. SRESULTS AND DISCUSSION

The results from the proposed semantic crawler for different parameters are shown below:

Accuracy: The accuracy is measured for different length of keywords and the result is below. From the result shown in fig. 9 we see that accuracy of Semantic crawler is better than Google Search.

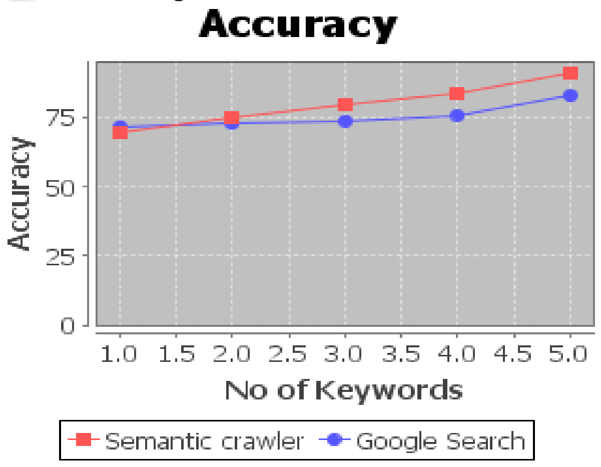


Fig 9: Comparative results of accuracy for proposed semantic crawler and Google search

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 4, April 2018

Table 1: Comparative results of accuracy for proposed semantic crawler and Google search

No. of keywords (in thousands)	Accuracy in percentage	
	Proposed semantic crawler	Google search
1	68	70
2	75	72
3	80	75
4	85	79
5	95	85

Information depth: Information depth is measured for different length of keyword and the result is shown in fig. 10 we see that the information depth ratio is higher in Semantic crawler when compared to Google Search.

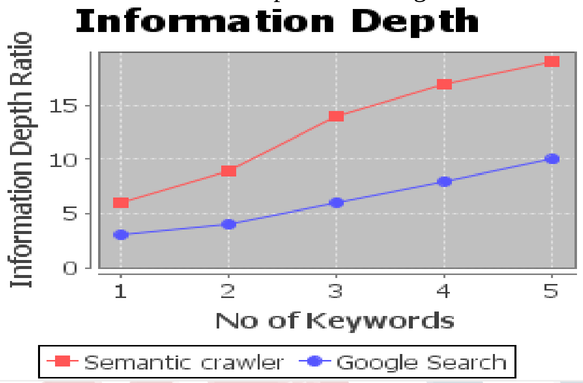


Fig 10: Comparative results of information depth for proposed semantic crawler and Google search

Table 2: Comparative results of information depth ratio for proposed semantic crawler and Google search

No. of keywords (in thousands)	Information depth ratio	
	Proposed semantic crawler	Google search
1	6	3
2	9	4
3	14	6
4	16	8
5	18	10

Harvest rate: Harvest rate can be measured by dividing total number of RDF searched pages by total number of pages and the result is shown from fig. 11. From the result shown in fig. 11 we see that the harvest rate is higher in Semantic crawler when compared to Google Search.

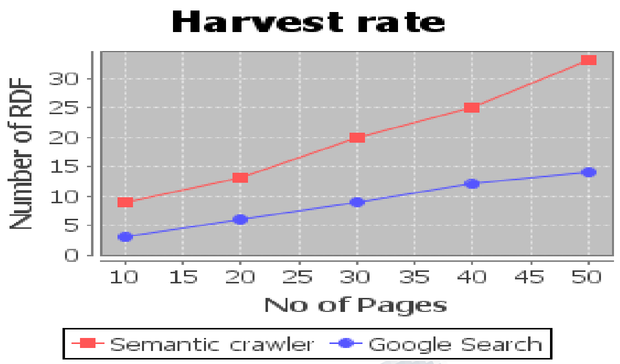


Fig 11: Comparative results of harvest rate for proposed semantic crawler and Google search

Table 3: Comparative results of harvest rate for proposed semantic crawler and Google search

No. of pages (in hundreds)	Harvest rate	
	Proposed semantic crawler	Google search
10	0.9	0.4
20	0.7	0.3
30	0.667	0.3
40	0.625	0.3
50	0.68	0.28

Running time: Running time can be measured by dividing total time taken in searching process by total number of RDFs and the result is shown from fig. 12. From the result shown in fig. 8 we see that the running time is higher in Google Search when compared to Semantic crawler.

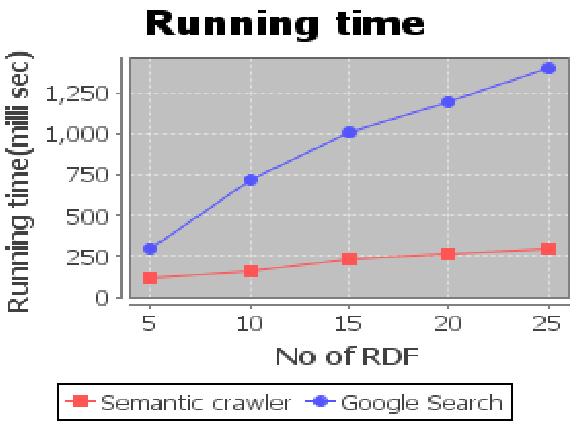


Fig 12: Comparative results of running time for proposed semantic crawler and Google search

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 4, April 2018

Table 4: Comparative results of running time for proposed semantic crawler and Google search

Total number of RDFs	Running time (in milli seconds)	
	Proposed semantic crawler	Google search
5	125	260
10	175	750
15	240	1000
20	250	1240
25	260	1450

Preference impact: Preference impact can be measured by dividing total number of filtered forms by total number of matched forms and the result is shown from fig. 12. From the result shown in fig. 13 we see that the preference impact is higher in Semantic crawler when compared to Google Search.

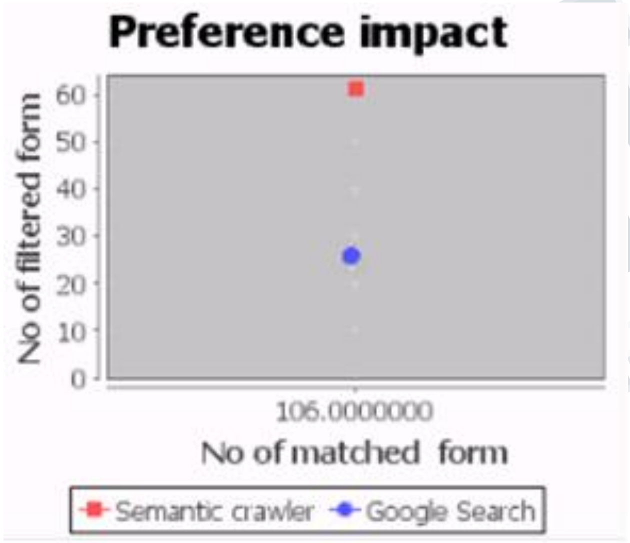


Fig 13: Comparative results of preference impact for proposed semantic crawler and Google search

Table 5: Comparative results of preference impact for proposed semantic crawler and Google search

Total number matched form	Preference impact	
	Proposed semantic crawler	Google search
106	0.566	0.235

Overall performance evaluation is presented by comparing the results of number of relevant forms harvested by Google search and proposed semantic crawler for multiple keywords as shown by fig. 14.

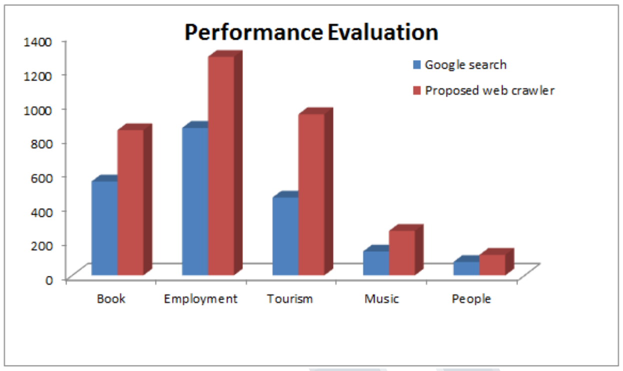


Fig 14: Comparative results of number of relevant forms harvested by Google search and proposed semantic crawler for multiple keywords

VIII. CONCLUSION

The proposed semantic crawler was able to crawled deep web and mines the RDF relevant to the search query. Through our tests, we have proved the coverage and quality of RDF is good. As noteworthy web makes at an expedient pace, there has been augmented vitality for methodologies that assist competently with finding huge web interfaces. Regardless, because of the expansive volume of web assets and the dynamic strategy for huge web, completing wide degree and high effectiveness is an attempting issue. We propose a two-stage structure, particularly Smart Crawler, for fruitful get-together noteworthy web interfaces. As a future work, we will consider how to extend the crawler to query on OWL data on internet.

IX. ACKNOWLEDGMENT

We acknowledge the IKGPTU, Kapurthala for given us an opportunity for the research work.

REFERENCES

1. Kevin Chen-Chuan Chang, Bin He, and Zhen Zhang. “Toward large scale integration: Building a metaquerier over databases on the web,” In CIDR, 2005, pages 44–55.

2. Denis Shestakov. “Databases on the web: national web domain survey,” Proc. of the 15th Symposium on International Database

Engineering & Applications, ACM 2011, pages179–184.

3. Denis Shestakov and Tapio Salakoski. “Host-IP clustering technique for deep web characterization,” Proc. of the 12th International Asia-Pacific Web Conference (APWEB), IEEE 2010, pages 378–380.

4. Denis Shestakov and Tapio Salakoski. “On estimating the scale of national deep web,” Database and Expert Systems Applications, Springer 2007, pages 780–789.

5. Shestakov Denis. “On building a search interface discovery system,” Proc. of the 2nd international conference on Resource discovery, Lyon France, 2010 Springer, pages 81–93.
6. Luciano Barbosa and Juliana Freire. “Searching for hidden-web databases,” WebDB, 2005, pages 1–6.

7. Luciano Barbosa and Juliana Freire. “An adaptive crawler for locating hidden-web entry points,” Proc. of the 16th international conference on World Wide Web, ACM 2007, pages 441–450.

8. Soumen Chakrabarti, Martin Van den Berg, and Byron Dom. “Focused crawling: a new approach to topic-specific web resource discovery,” Computer Networks, 31(11): 1999, 1623–1640.

9. S. Brin and L. Page, “The anatomy of a large-scale hypertextual web search engine,” Proc. of the 7th International World Wide Web Conference, 1998.

10. Yahoo! Research Barcelona, “Datasets for web spam detection,” http://www.yr-bcn.es/webspam/datasets

11. H.-T. Lee, D. Leonard, X. Wang, and D. Loguinov, “IRLbot: Scaling to 6 billion pages and beyond,” Proc. of the 17th International World Wide Web Conference, 2008.

12. Internet Archive, “Heritrix home page,” http://crawler.archive.org/.

13. G. Mohr, M. Stack, I. Ranitovic, D. Avery, and M. Kimpton, “An introduction to Heritrix, an open source archival quality web crawler,” Proc. of the 4th International Web Archiving Workshop, 2004.

14. R. Khare, D. Cutting, K. Sitakar, and A. Rifkin, “Nutch: A flexible and scalable open-source web search engine,” Technical Report, Commerce Net Labs, 2004.