International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 22 (2018) pp. 15544-15555


# Deep Web Efficacy: A Knowledge Perspective

**Manpreet Singh Sehgal[#1], Jay Shankar Prasad[#2].**

[#1,2] *Department of Computer Science and Engineering, MVN University.*

**Abstract**

Since the start of the civilization knowledge extraction is fascinating and during the era of WWW, the knowledge has found strong root in HTML documents and electronic databases, sometimes in the proprietary hardware and also in the cloud storage. The means of acquiring knowledge is through accessing the information which utilizes the web crawlers. Contemporary crawlers cannot crawl and index the information stored in databases unless web search interface forms carry required data which need to be filled by information seeker to fetch the information on demand. The research on the abilities of web crawlers and the efforts to design crawlers to extract this data hidden in the electronic databases (by automatic filling search interfaces and by other mechanisms) has begun in the first decade of twenty first century. This paper presents a comparative study of such crawlers mentioning their merits and demerits so as to help readers to find the research gaps and unhide the deep web for knowledge perspective.

**Keywords:** Surface Web, Hidden Web, Deep Web, Deep Web Crawler, Data Extraction.

## I. INTRODUCTION

In the digitized world, whenever there is a need to know something, people look upon context sensitive experts (also called search engines) which have enough knowledge to quench the thirst of knowledge. In the earlier days of World Wide Web, when data was not big and the computing was not done on cloud, and the knowledge was weaved only into the HTML pages, these experts used to crawl this knowledge from one HTML page (retrieved from seed url) and extends its reach to other HTML pages which this HTML page refers to.

The problem arose when organizations started storing their contents in the databases as the data starts getting big and HTML designers started designing search interfaces to peek into this information. The crawlers of search engines were not trained to crawl and index such piece of information. Due to this the information stored inside databases which is away from the reach of crawlers started getting called hidden or deep information. The size of this information hidden behind search interfaces into the databases is many folds bigger in size, quality and relevance than what is present in HTML pages [1]. Popular examples of such collection of hidden knowledge are PubMed [5] which is a structured database of biomedical publications and US Patent and Trademark Office (USPTO) [6] a collection of US Patents and Trademarks. Consequently, a breeze of crawlers with the ability to fetch hidden data has started to flow in the beginning of twenty first century. The term used to refer such crawlers is hidden web crawlers [2], [3], [4].

## II. FEATURES AND MEASURE OF DEEP WEB

The deep web is a gateway to the large and fast evolving databases on the World Wide Web. At the beginning of efforts of accessing this deep web, the studies were conducted to estimate the size of the deep web. In 2001, Bergman[1] approximated its size to be about 500 times the size of HTML linked and weaved web (also called surface web), that spanned across 7500 terabytes. Chang et. al [7] in 2004 revealed that the nature of stored data in databases is structured. The approach used to figure this out was random IP Sampling. After three years in 2007 they found that major search engines like Yahoo!, Google and MSN do index only 37% of whatever is available in electronic form [8]. Hence, the salient features of Deep Web consists of high relevancy and good quality rich data, structured type, voluminous and focused than contents surface web documents.

## III. DIGGING THE HIDDEN WEB

In the absence of hidden web crawlers, the only way to access the hidden web is through query interfaces that the user must fill and submit to get the required result. The design of hidden web crawlers follow the automation process of filling the search query forms and submitting the results. For this process, there are two methods viz. surfacing and generic data integration. Surfacing is the job of the crawler that it does in the background to fetch relevant, interesting and quality information and hence updates the search engine's index. The job of a hidden web crawler is to process the query forms and submit them as well to retrieve (download) the result pages to be used later with the search engine's index structure as depicted in Figure 1.

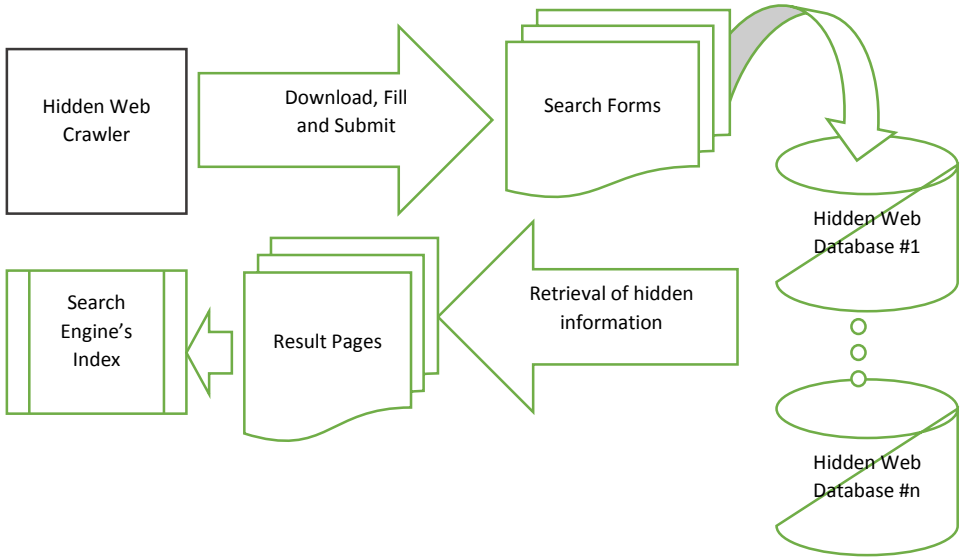


**Figure 1:** Surfacing the Hidden Web

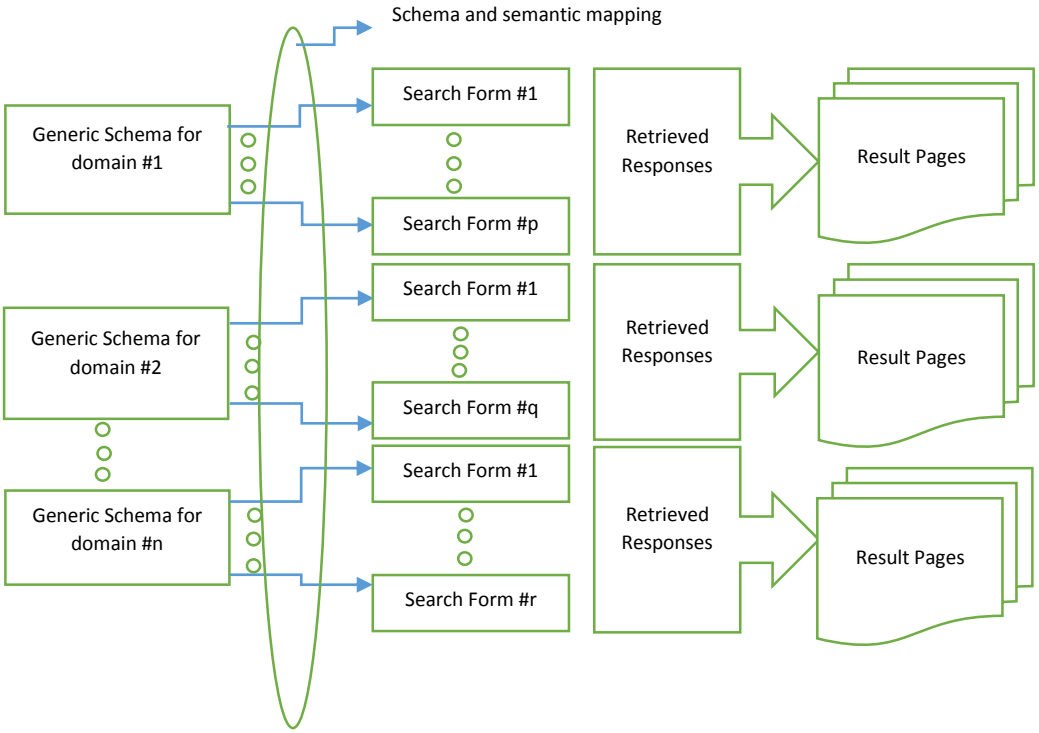


**Figure 2.** Generic Data Integration

Although finding out the best filling pattern for form submissions is a complex job yet it can be executed passively by an active crawler without affecting the assets of hidden web making surfacing a direct approach. Generic data integration refers to creating a generic search form for a specific domain of hidden website. The filling schema of the generic search form is mapped with the actual form fields to ensure correct submission. This attempt is more optimized than surfacing as it tries to fill multiple search forms with only one generic form. However, this is a slower approach than surfacing as lots of external API's need to be called by the search engines and hence the speed is limited by the performance of deep web resources. The process is depicted in Figure 2. Generic Schemas for 'n' domains are feeding search forms from their domains to retrieve responses that are used by API's to fetch result pages. In the diagram, domain #1, #2 and #n has #p, #q and #r search forms associated with generic schema. Designing of schema and semantic mapping is the tough proposition. This

requires careful analysis of the search forms of the associated domain so that the keywords in the queries can be easily migrated to the fields in the search forms. Moreover, the number of domains (#n) is infinitely large hence it makes virtually impossible to design infinitely large schema and semantic mappings.

Despite the research efforts on generic data integration, the difficulties involved have inspired us to opt for the former approach of surfacing the hidden web and discussed hereafter.

## III. HIDDEN WEB SURFACING APPROACHES

The hidden web crawler should imitate the human activities in the search of hidden web content. It should be able to fill the search form and submit that to fetch the result pages. The authors in [4] proposed the algorithm shown in Figure 3 to surface the hidden web. Resource discovery and content extraction are the two main objectives of the hidden web crawling. Resource discovery deals with the process of automatically finding out the relevant websites that contain a search form interface. Sehgal and Prasad in [9] have suggested ontological approach for the recourse discovery problem. The content extraction is about extracting the information itched in the databases. Sehgal and Anuradha in [10] have proposed HWPDE approach for the content extraction and in the process they introduced Sponger and Squeezer to extract and pour the extracted data in the automatically created database by their proposed algorithm. Due to inherent limitations imposed on the crawlers to uncover the hidden web along with the ever growing size of this repository the two kinds of crawling is reported in literature viz. Wide Crawling and Deep Crawling.

```
Algorithm Hidden_web_crawl()
Input:
Output:
 While (Resources Available) do{
                                        qi=SelectTerm()
          //Select term to send to the site.
                                R(qi)= QueryWebSite(qi)
// Where qi is the selected query and R(qi) is the result page for the query qi.
                                Download (R(qi))
                                }
 End
```

**Figure 3.** Algorithm to Crawl Hidden Web Site

Wide Crawling: This is also called breadth oriented crawling. The emphasis of such crawling is to cover the maximum domains, rather than going deep into a particular domain and exhausting all its resources. The main challenge is to design a model to identify the sources of hidden web information that can also analyze the returned results. Authors in reference [9] presented a model using ontologies to identify the sources of information that can be further analyzed by independent tools.

1. Deep Crawling: This is also called depth oriented crawling. The emphasis of such crawling is to cover one specific domain, rather than covering wider domains. All websites of one particular domain are crawled and indexed using deep crawling. The goal is to collect most of the data using minimum number of queries. The query generation or selection problem is required to be an automatic process.

The above types of crawling also takes into consideration the type of information stored in the hidden web data sources. Two categories of information impact the process of crawling, they are unstructured and structured. Unstructured information lies in the plain text documents and is searched using keywords provided in the search boxes and the results are presented in the result pages. Figure 4 presents the generic form of such interface and the connection to connect unstructured collection from the interface. On the other hand structured databases are crawled using multiple attribute search forms, where many query fields are present to gather different features of the information. Figure 5 shows the generic form of such an interface. K attributes are asked from the user and upon submitting the structured database is issued the query formulated by fitting in all the attributes entered by user. The tables in the structures databases are looking for the required results which are presented to the user in the result pages.

Crawling of web has been existing since the beginning of web whereas the crawling of hidden web has started with the work of Raghvan and Molina [2] in 2001 in which they focused on the design of extracting data from the electronic databases. This marked the beginning of exploration of hidden web using deep crawling. Due to the enormous effort and challenges involved in the wide crawling, the number of efforts were limited and hence less discussed in the literature. This paper is an attempt to bridge this gap by elaborating on the area of wide crawling along with the discussion of deep crawling. In the next sections, the deep crawling for structured and unstructured databases followed by the detailed efforts in the field of wide crawling of structured and unstructured databases has been discussed.

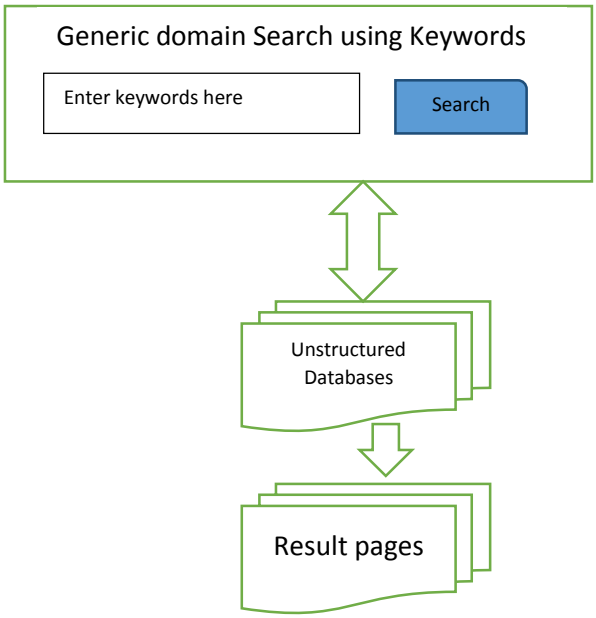


**Figure 4.** User Interface to Unstructured databases

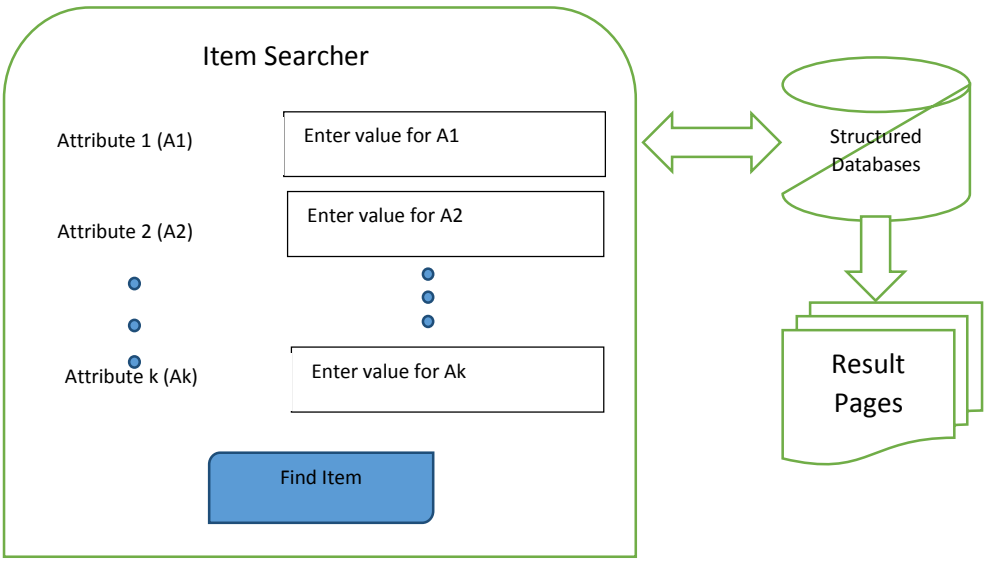


**Figure 5.** User Interface to Structured Data Source

### A. Deep Crawling for Structured Databases

The operational model for deep crawling proposed by Raghvan and Molina [2] is presented in Figure 6, illustrates the interaction between the crawler and the search form [2]. The prototype hidden web crawler HiWE (Hidden Web Exposer) followed a model. The outline of HiWE is depicted in Figure 7. The methodology proposed by [2] is to auto fill the 'search forms' from the manual queries or through the queries generated from the query interfaces. The 'search form' is depicted as "Form Page" in figure 7 and response page is what user gets in response to submitting a form. The Form page consists of form elements like input box, radio buttons, checkboxes etc. Each element is filled with values from its domain after inspecting the descriptive text (label) associated with them. The selection of values for the element depends on the closest location of the labels from the four nearby label locations. The probable assignments to the form are calculated from the LVS table values which are Label (L)-Value (V) tuples. 'V' here is a graded/fuzzy set of values associated to the label. The measure selected by the authors[2] is the fraction of non – error pages returned. After the submission of the candidate assignment the result pages are stored in the database for future user query support. However to deal with the elements with infinite domain was a challenging task.

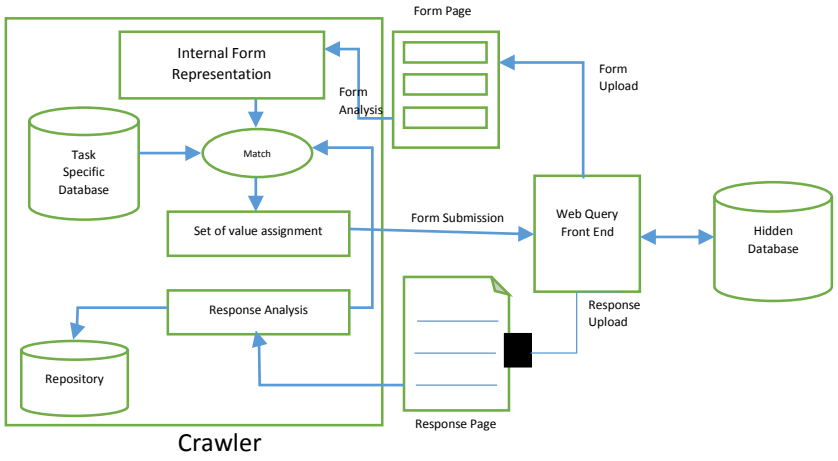


**Figure 6.** Interaction between crawler and the search Form

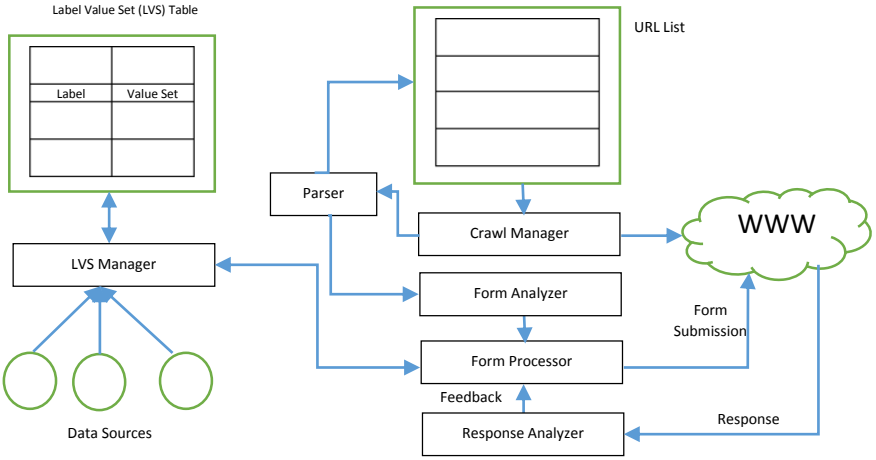


**Figure 7.** Architecture of HiWE.

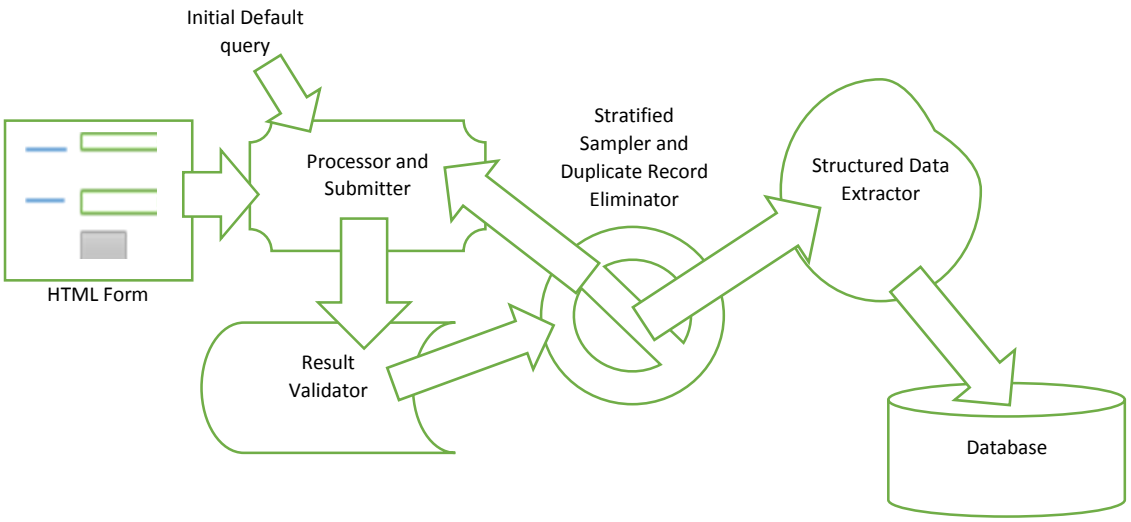


**Figure 8.** Approach of Liddele

Liddle et al. in 2002 introduced approach to detect form elements and using default values [11] for each field to generate the submission request in the form of HTTP GET request. This approach is not automatic like HiWE and takes user intervention as well. The role of this approach is to acquire all or major percentage of the data before submitting the label value assignments (virtualization of queries). Figure 8 depicts the approach in the block diagram. HTML form is fed with the default query for the first time. The fetched results are validated and duplicate records are eliminated. The semi structured data fetched, forms the basis of the new and processed queries using stratified sampling method so that in the next result new information that is not fetched so far can be fetched. The basic idea of such sampling is to treat all form elements with equal probability. The process of exhaustively issuing new processed queries continues until a specific threshold is attained. This semi structured data needs filtering which structured data extractor performs and extract the structured data for populating the database.

Wu et al.[12] in 2006 devised a mechanism to crawl a structured web source efficiently using graph traversal problem where the links in the structured databases are the edges in the graph and pages are the vertices. The structured database was the part of one large relational table as shown below in Table 1.

**Table 1.** Structured Web Source of Wu et al.[ ]

| | | | |
|---|---|---|---|
| $t_1$ | $a_1$ | … | $a_m$ |
| $t_2$ | $a_1$ | … | $a_m$ |
| $t_3$ | $a_1$ | … | $a_m$ |
| … | $a_1$ | … | $a_m$ |
| $t_n$ | $a_1$ | … | $a_m$ |

The data records are in the form of $\{t_1, t_2, \ldots, t_n\}$ over the $m$ attributes $\{a_1, a_2, \ldots a_n\}$. The distinct attribute values (DAV) are stored in DAV Set. Depending on the web source, the attribute-value undirected graph (AVG) is constructed where each vertex $v_i \epsilon V$ is a value ( $av_i$ )from the DAV set, and each edge $(v_i: v_j)$ represents the coexistence of the values for $a_i$ and $a_j$ in the record $t_k$. This edge relationship corresponding to Table 1 is shown in Figure 9a and 9b

| | | | | |
|---|---|---|---|---|
| $t_k$ | $a_i$ | $a_j$ | … | $a_m$ |

**Figure 9 (a).** Structured web source entry


**Figure 9(b).** Corresponding AVG

The crawler starts the traversal of the graph from the set of seed vertices and iterates the AVG graph. After each iteration, the last seen node is selected to visit with its new attributes. Hence, the entire graph nodes along with their records and different attributes is read and stored in the repository for future visits and to maintain sensible reportage.

The Google's approach to fill the forms was studied by Madhavan et al.[ ] in 2009. HTML form presents more than one input so cartesian product generates a large search space. Madhavan et al.[ ] proposed algorithm to select appropriate subset of this cartesian product to include only sensible combinations in the index. The algorithm is a three step approach wherein the first step is the formativeness test where all query templates are probed to test the ability of returning adequately distinct documents. In the second step, the query search space is traversed to figure out which of the templates are appropriate for surfacing. The last step selects the appropriate values for the various form fields (e.g. Prices, dates, pin code) for better crawling.

In [13], a domain specific hidden web crawler, DSHWC to target multi-input form was developed. This crawler works in three phases. In the first phase, the search interfaces are downloaded. In the next phase the semantic relationships among attributes of different search interface elements is found out. The last phase is the integrator phase that clubs all the search forms and generically forms a Unified Search Interface (USI). The process of automatic filling the USI and submitting it is followed by the process of storing and indexing the result pages into the repository maintained by this crawler. This phased process is called the AKSHAR Approach. The pictorial representation of the phases is depicted in Figure 10.

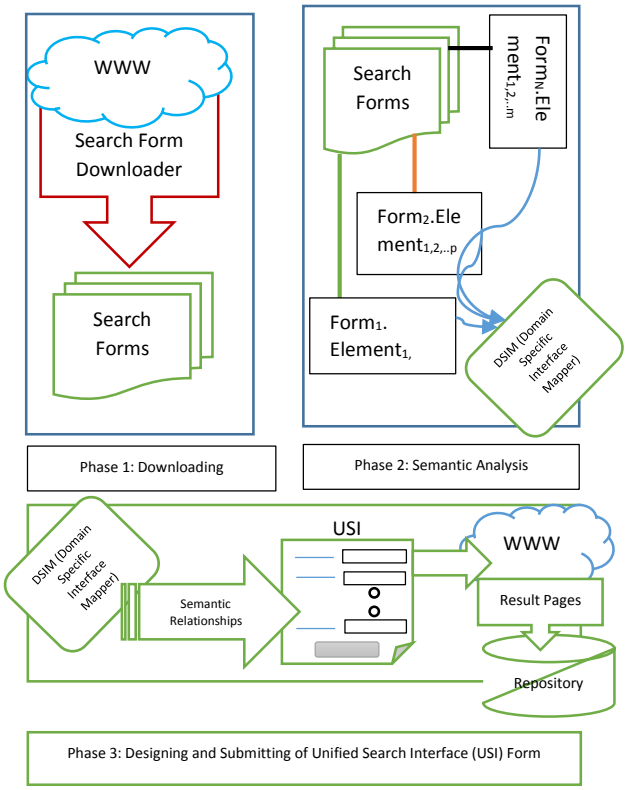


**Figure 10.** The AKSHAR Approach

### B Deep Crawling for Unstructured Databases

Immense strides have been taken to uncover the data hidden behind interfaces based on keywords. This section describes some of them.

Gravano et.al. in 2003 proposed a technique [14] for automating the data extraction from textual repositories by taking a sample of documents with some bias. These documents are extracted after adaptively penetrating the database with topically motivated queries. These queries are automatically derived with the help of a classifier on a Yahoo! like directory of topics. In their work they also evaluated the results and exploited the statistical properties of text so as to calculate the estimation of word frequency in extracted documents.

Barabosa et.al in 2004 proposed a two phase algorithm [3] to formulate text based queries. The first phase of the algorithm forms a data sample from the website and then discovers the set of keywords (associated weights) from the formed sample. In the second phase of the algorithm, these keywords are arranged to formulate queries to siphon the data from the web. This process used greedy approach so as to expend minimum queries to extract maximum content. However, assumed that queries based on non-text based controls like radio buttons, check boxes etc. are easier to formulate as text based controls like input search box expose wide domain of possibilities for the query builders.

In 2005, A. Ntoulas et.al proposed theoretical framework [4] to analyze the single attribute query generation process guided by feedback of generated results. This feedback policy is termed as adaptive policy as the process adapts itself according to the generated result to get new result every time. They compared this approach with two other policies, random policy and generic policy where in random policy ensures the random pickup of attributes (queries) from dictionary and generic policy is based on the keyword frequencies in any generic document corpus. The experimental results exhibited the positive trends with the adaptive approach.

### C. Wide Crawling

The first ever effort in the direction of Wide Crawling was made in 2003 by Bergholz et. al [15] to identify the hidden web entry points from the Surface web. The strategy adopted was highly domain dependent and experimental results were excellent.

Two years later in 2005, Barbosa and Freire [16] suggested a Form focused crawler (FFC) that locates topic based web forms. The organization or components of FFC are shown in Figure 11. The page and link classifiers are provided training data so that the crawler can focus its crawl on a selected topic after analyzing the page contents and patterns in and around the hyperlinks directed to a webpage. The backward search strategy was employed first to figure out the suitable links that might lead to the searchable forms in some positive steps. The link frontier provides the feedback to fine tune the crawling process. Form classifier is used to filter out the searchable forms and to store them in the form database.

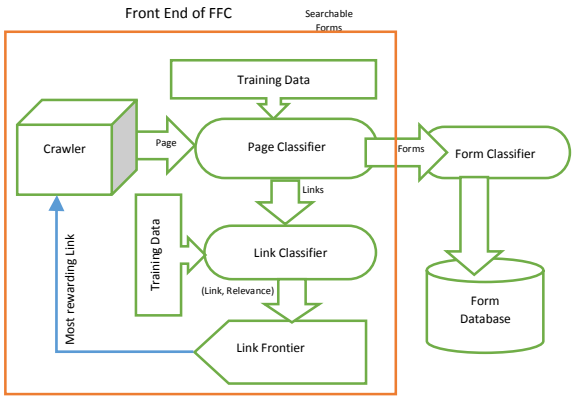


**Figure 11.** Form Focused Crawler

Barbosa and Firere [17] addressed the limitations of their project and presented a framework called ACHE (Adaptive Crawler for Hidden-Web Entries). This architecture is a two phase addition to the existing FFC.

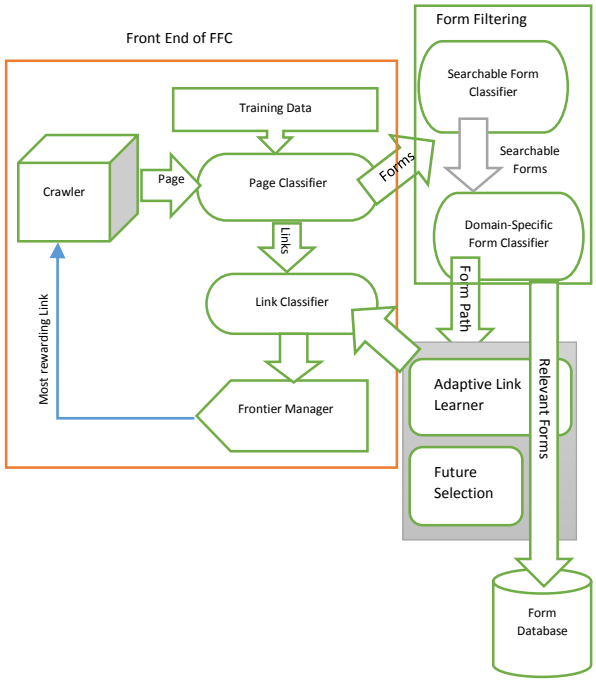


**Figure 12.** ACHE Organization

Form Filtering has taken over form classifier. Form Filtering comprises of two classifiers. Searchable form classifier (SFC) is to filter out searchable forms from the mixed collection of forms. Domain-Specific Form Classifier (DSFC) validates the applicability of the searchable form onto the domain. The two module package containing Adaptive Link Learner and Future Selection replaced the training data to link classifier. This replacement paves the way to dynamic learning of the features which are automatically extracted from 'Form paths'. This learning is later updated to the link classifier so as to fine tune the crawling process through Link Frontier that provides the most rewarding link to the crawler.

In 2018, Sehgal and Prasad [9] suggested ontological approach to penetrate the web on the assumptions of the presence of the

master ontology which binds all the ontologies of all domains together. In case if some domain does not find its relevant ontology then the need to build the corresponding ontology gets notified to the assumed component named as Ontology-Builder. The domain independent knowledge seeker does not need to get bound for a specific domain to search for information.

In the next section a brief comparison of the efforts made in the web excavation to unhide the deep web both in vertical (deep crawling) and horizontal (wide crawling) dimensions is made.

## VI. COMPARISON OF PREVIOUS RELATED WORK

The attempts made to fetch hidden web data are classified into crawling and page data extraction. Hidden web crawlers are trained to fill the forms and submit in anticipation of returning valuable and relevant data. The techniques followed are deep and wide crawling techniques discussed in this paper. Detailed comparative study of approaches including that of crawlers addressed in this research is presented in Table 2. The orientation of the research is deep and wide corresponding to the approach used for hidden web crawling. The table discusses the positives and perils of the researched efforts to provide the reader enough impetus to get guided for the future research.

**Table 2.** Comparison among different crawlers and Extractors

| Researcher | Year | Orientation | Database Type | Technique | Positives | Perils |
|---|---|---|---|---|---|---|
| Raghavan et al. [2] | 2001 | Deep Crawling | Structured | 1. Text Similarity for matching fields and attributes of domain.<br>2. Form elements identification using Partial Page layout & visual Adjacency.<br>3. Page's Visually Imported parts are hashed for error detection. | 1. Significant contribution to the process of label matching.<br>2. Feedback mechanism to fine tune the crawling process. | 1. Minimum threshold of 3 form elements for consideration.<br>2. Significant human involvement<br>3. Not scalable |
| Liddle et al. [11] | 2002 | Deep Crawling | Structured | 1. Stratified Sampling.<br>2. No auto filling of finite text elements<br>3. Pages concatenated if linked. | 1. Domain independent<br>2. Duplicate results are accounted for. | 1. Forms inside results are not considered<br>2. Huge resource requirement. |
| Garvano et al. [14] | 2002 | Deep Crawling | Unstructured | 1. Topically focused queries<br>2. Adaptive query probing | 1. Meta search engine design facilitated.<br>2. Hidden Web databases categorized. | 1. Directory type query selection.<br>2. No flat classification. |
| Bergholz et al. [15] | 2003 | Wide Crawling | Unstructured | 1. Domain Specific Crawling.<br>2. Recognition and assessment of usefulness of hidden web resource by query prober | 1. Combination of query prober and syntactic HTML elements caused efficient crawling | 1. Only full text search forms are considered.<br>2. It is initialized with relevant keywords and pre classified documents. |
| Barbosa et al. [3] | 2004 | Deep Crawling | Unstructured | 1. Checks the frequency of appearance of a query in each round | 1. Full automated and simple approach<br>2. High coverage. | 1. No guarantee of retrieving new pages.<br>2. Not usable for static search |

| Researcher | Year | Orientation | Database Type | Technique | Positives | Perils |
|---|---|---|---|---|---|---|
| | | | | to decide its candidature. | 3. Self-creation of document description to make it usable in other discovery systems. | interfaces with fixed output rate.<br>3. Simplicity and Security Tradeoff. |
| Ntoulas et al. [4] | 2005 | Deep Crawling | Unstructured | 1. Method of Incremental adaption<br>2. Frequency estimated from already downloaded documents<br>3. Attempts are made to maximize gain using greedy approach. | 1. Goodness of random generic and adaptive approaches to choose appropriate queries.<br>2. Zipf;'s law based and independent frequency estimators. | 1. Huge memory requirements for calculating potential gain.<br>2. Difference in assumed constant cost and real situation cost.<br>3. No assurance of query distribution adaption with database attributes. |
| Barbosa et al. [16] | 2005 | Wide Crawling | Structured & Unstructured | 1. Topic specificity in search.<br>2. Useless searches are stopped using stop criteria. | 1. Highly efficient in retrieving topic specific search forms. | 1. Difficult to select training set manually.<br>2. Creating a link classifier is time consuming. |
| Alvarez et al. [18] | 2006 | | | 1. There are domain definition sets which describe Data collection task.<br>2. Heuristics are used to automatically identify query forms that are relevant. | 1. Extendable system for finding the relevant resources.<br>2. Both client Side and Server Side hidden web taken care of.<br>3. Effective data collection proved experimentally | 1. Absence of threshold definition for mapping of form elements to attributes.<br>2. Imaginary Assumption of association of all form elements with label which is not true for bounded form elements (combo boxes). |
| Ping Wu et al. [12] | 2006 | Deep Crawling | Structured | 1. Structured database modeled as a distinct attribute – value graph.<br>2. Problem of crawling is reduced to set covering problem. | 1. Overcomes greedy method limitations.<br>2. Domain knowledge tunes meaningful | 1. Expensive approach as in each round the graph size increases due to addition of query results. |

| Researcher | Year | Orientation | Database Type | Technique | Positives | Perils |
|---|---|---|---|---|---|---|
| | | | | | queries to be issued. | |
| Barbosa et al. [17] | 2007 | Wide Crawling | Unstructured | 1. Weighted keywords in the retrieved data derives greedy algorithm.<br>2. Error page detection using dummy word queries. | 1. Improvement of harvest rate with the progression of crawl.<br>2. Fetches similar set of forms.<br>3. Adaptive and automated approach eliminating any bias in the process of learning. | 1. More efforts expended to start the crawler than that of expended to start manual crawlers.<br>2. Recognizes only single keyword queries. |
| Madhvan et al. [ ] | 2008 | Deep Crawling | Structured | 1. query template evaluation by in formativeness test definition | 1. Possible input combinations are efficiently navigated | 1. Hidden web crawling efficiency is ignored. |
| Komal Bhatia et al. [ ] | 2010 | Deep Crawling | Structured | 1. Unified interface for the domain.<br>2. Probability of Changes occurring in webpage decides the frequency of revisit. | 1. To avoid repetition to minimize mapping effort, the mapping knowledgebase is used.<br>2. Pave the way to develop the specialized search engine. | 1. Only crawling performance is defined. Other factors were ignored.<br>2. Storage was not indexed. |
| Sehgal et al. [10] | 2012 | Page Data Extraction | Unstructured to Structured | 1. Table tag identification.<br>2. Data type detection | 1. Automatic Sponging of tabular data from HTML page.<br>2. Automatic Squeezing of Sponged data into the repository. | 1. Only workable for table tag.<br>2. Only a Desktop solution and is not available as a web service.<br>3. Requires human effort |
| Sonali Gupta et al. [ ] | 2013 | Deep Crawling | Unstructured | 1. Builds up a domain representation which is stored in the repository specific to that domain.<br>2. Query term is identified using a classification hierarchy that is domain specific. | 1. Maximum gain with minimal input.<br>2. Domain specific approach that can be used for other domains. | 1. Domain Specific<br>2. Requires human effort at the beginning. |

| Researcher | Year | Orientation | Database Type | Technique | Positives | Perils |
|---|---|---|---|---|---|---|
| Aysha Banu et al. [19] | 2014 | Page Data Extraction | Unstructured to Structured. | 1. Coherence based page ranking algorithm.<br>2. WordNet approach to extract data and check similarity. | 1. Reduction in search space.<br>2. Higher precision, recall and filter accuracy | 1. Comparison made only with ViDRE [20] |
| L.Saoudi et al. [21] | 2015 | Deep Crawling | Structured | 1. SQL Injection approach to find the relevant keywords. | 1. Recall in form domain association reached 1.<br>2. Precision in form field and attributes mapping exceeded 0.92 with the recall 0.9 | 1. Failed for one source Blackwell's Bookshop.<br>2. The research does not handle JavaScript powered forms. |
| B.Liu et al. [22] | 2016 | Deep Crawling | Structured | 1. Meta Information Extraction from query interfaces.<br>2. Method is based on user defined rules and visual features | 1. It solved the Problem of maintaining and managing source information.<br>2. The research defined rules to extract data from hidden web query interfaces. | 1. For small number of source information it generates errors.<br>2. The research is not able to handle constant HTML version changes and different coding styles of web developers. |
| Umamageswari et al. [23] | 2017 | Deep Crawling | Structured | 1. Heuristic Algorithm based on DOM Tree and XPATH for navigating automatically and extraction of information | 1. It does not require the entire DOM Tree to be processed.<br>2. Linear time Complexity.<br>3. If domain keywords and synonyms are known than the technique can be applied to other domains. | 1. It is not tuned for journal pages structured in a different manner but from the same publishers' web site.<br>2. There is no handling of attribute ordering<br>3. There is no adhoc accommodating of new templates. |
| Sehgal et al. [9] | 2018 | Wide Crawling | Structured and Unstructured | 1. Universal algorithm based on ontological knowledgebase to crawl all domains. | Feasible to implement provided ontologies are provided. | It does not address automatic CAPTCHA filling. |

## CONCLUSIONS

The role of hidden web crawlers is to index, analyze and mine the hidden web content for knowledge extraction. The hidden web databases can further be classified by the extracted content. The paper lets the reader travel in the journey of several hidden web crawlers developed for exposing hidden web along with the stepping stones of hidden web page data extractors. The paper also compared the various techniques in the direction of unhiding the deep web for knowledge extraction. The approaches have been differentiated on the basis of the underlying techniques. Each of the approach has its own limitation and strengths. For tuning this research effort for better retrieval and more quality, more inputs from future researchers are needed so as to reap the benefits of the triggers presented in the paper.

## REFERENCES

[1] M. Bergman. "The deep web: Surfacing hidden value." In the journal of Electronic Publishing 7(1) (2001).

[2] S. Raghvan. H. Gracia-Molina. Crawling the Hidden Web. In the proceedings of the 27th International Conference on Very large databases VLDB'01, Morgan Kaufmann Publishers. Inc., San Francisco, CA, p.p. 129-138.

[3] L. Barbosa, J. Freire: Siphoning hidden-web data through keyword based interfaces. In: SBBD, 2004, Brasilia, Brazil, pp. 309-321.

[4] A. Ntoulas, P. Zerfos, J. Cho. Downloading Textual Hidden Web Content Through Keyword Queries. In 5th ACM/IEEE joint conference on Digital Libraries (Denver, USA. Jun 2005) JCDL05, pp 100-109.

[5] Available at: http://www.ncbi.nlm.nih.gov/pubmed/. Accessed October 10, 2018.

[6] Available at: http://patft.uspto.gov/. Accessed October 10, 2018.

[7] K.C. Chang. B. He, M. Patel, Z, and Zhang: Structured databases on the web: Observations and Implications. SIGMOD Record, 33(3). 2004.

[8] B. He, M. Patel, Z, Zhang, and K.C. Chang: Accessing the Deep Web: A survey. Communications of the ACM, 50(5): 95-101, 2007.

[9] Manpreet Singh Sehgal, Jai Shankar Prasad, "All Domain Hidden Web Exposer Ontologies: A Unified Approach for excavating the web to unhide deep web" accepted for publishing in the proceedings of 2nd International Conference on Smart Innovations in Communications and Computational Sciences ICSICCS-2018

[10] Manpreet Singh Sehgal and Anuradha and. Article: HWPDE: Novel Approach for Data Extraction from Structured Web Pages. International Journal et.al Computer Applications 50(8):22-27, July 2012

[11] S. W. Liddle. D. W. Embley, D.T. Scott. S.H. Yau. Extracting data behind Web Forms. In: 28th VLDB Conference 2002, Hong Kong, China.

[12] Ping Wu. J-R. Wen. H. Liu, and W.-Y. Ma. Query Selection Techniques for efficient crawling of Structured Web Sources. In ICDE, 2006

[13] Komal Kumar Bhatia. A.K Sharma, Rosy Madan: AKSHAR: A Novel Framework for a domain specific Hidden web crawler. In Proceedings of the first International Conference on Parallel, Distributed and Grid Computing, 2010.

[14] P.Ipeirotis and L. Gravano, "Distributed Search over the hidden web: Hierarchical database sampling and." in VLDB, 2002.

[15] A Bergholz, B. Chidlovski. Crawling for domain-specific hidden web resources. In Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE`03). pp 125-133 IEEE Press, 2003

[16] L. Barbosa and J. Freire. Searching for Hidden-Web Databases. In proceedings of WebDB, pages 1-6, 2005

[17] L. Barbosa and J. Freire, "An Adaptive Crawler for locating Hidden Web entry points," in Proc. of WWW, 2007, pp 441-450.

[18] Manuel Álvarez, Juan Raposo, Alberto Pan, Fidel Cacheda, Fernando Bellas, Víctor Carneiro: Crawling the Content Hidden Behind Web Forms. In Proceedings of the 2007 International conference on Computational Science and its applications, Published by SpringerVerlag Berlin, Heidelberg, 2007.

[19] Aysha Banu, M. Chitra, "DWDE-IR: An Efficient Deep Web Data Extraction for Information Retrieval on Web Mining", Journal of Emerging Technologies in Web Intelligence, vol. 6, 2014.

[20] X. Meng, W. Liu and W. Meng, "ViDE: A Vision-Based Approach for Deep Web Data Extraction," in IEEE Transactions on Knowledge & Data Engineering, vol. 22, no. , pp. 447-460, 2009. doi:10.1109/TKDE.2009.109

[21] L.Saoudi , A.Boukerram and S.Mhamedi, "A New Hidden Web Crawling Approach" International Journal of Advanced Computer Science and Applications(IJACSA), 6(10), 2015. http://dx.doi.org/10.14569/IJACSA.2015.061039

[22] B. Liu and J. Xiang, "Extraction and management of meta information on the domain-oriented Deep Web," 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, 2016, pp. 787-790. doi: 10.1109/ICSESS.2016.7883185

[23] Umamageswari Kumaresan and Prof. Kalpana Ramanujam, "Web Data Extraction from Scientific Publishers' Website Using Heuristic Algorithm," I.J. Intelligent Systems and Applications, 2017, 10, 31-39 Published Online October 2017 in MECS (http://www.mecs-press.org/)