International Journal of Engineering & Technology, 7 (4) (2018) 4398-4404

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi: 15751
Research paper

UDDWE: Universal domain deep web exposer

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

[1] Research Scholar, Department of Computer Science & Engineering, MVN University, Palwal, India
[2] Department of Computer Science & Engineering, MVN University, Palwal, India
**Corresponding author E-mail: manpreet.sehgal@asu.apeejay.edu*

Abstract

Traditionally, the search engines were able to extract web data which is smaller in size as compared to much relevant and quality data (also called hidden web data) hidden behind search interfaces. A lot of research has been done to extract this web data to fetch its relevant and quality content. However, most of the methods are domain specific i.e. for multiple domains multiple tools are designed. In this paper, a novel method is proposed to present one universal tool for all the domains. The key point in this approach is the customization of the traditional search engine to receive the user query to process it for identifying the entry points (search interfaces) to the hidden web. After this filtering process, the entry points are presented for opening in a controlled programmed environment to ease the data extraction process.

Keywords: *Deep Web; Hidden Web; Information Retrieval; Search Interfaces; Universal; Domains.*

1. Introduction

The World Wide Web exists in two major categories, publically indexed Web (referred as PIW) and Deep Web (also called hidden web). These two parts differ in their sizes, where PIW takes up approximately 1% [1], [2], [3] of the entire web and Deep web takes the rest of the web which is rich in quality and relevance. Despite the fact of being the major part of the web, deep web is not easily accessible due to two major reasons, one being its presence behind search interfaces [9], [11] and other because of the inability of traditional search engines to index documents containing the hidden web pages. The current research in the area of hidden web crawling is focused on designing specialized tools for specific domains so as to deeply invade the domain.

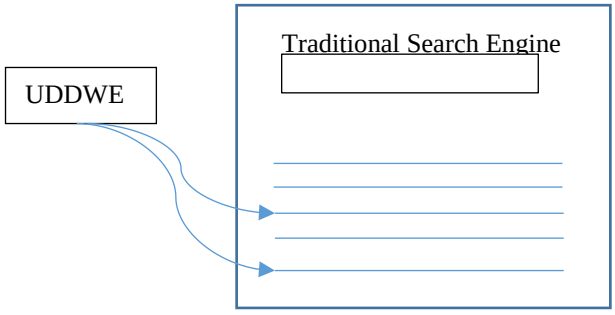


Fig. 1: Search Engine Links Used As Entry Points in UDDWE, UDDWE: Universal Domain Deep Web Exposer.

This kind of research contribution necessitates 'n' number of tools for 'n' number of domains, thereby causing enormous effort on the part of researchers and implementers. For example for domain 'car' there is one tool that extracts the hidden web data from the hidden websites of car domain after analyzing the structure and layout of the various car websites. With a different domain the different tool is required and hence designed, developed and gets deployed. Therefore there seems to be a gap in research efforts to create 'one' generic tool to address almost all possible domains.
This paper addresses this gap and suggests a novel method to extract hidden web data for the multiple domains. This method is dependent on the traditional search engine's output as it contains the links to the search interfaces in the mixed resultant index that it generates. This paper further proposes the technique to identify such links from the output of traditional search engine as depicted in Fig. 1. It also suggests the way to auto-fill and submit search interfaces to gather hidden web data as depicted in Fig. 2

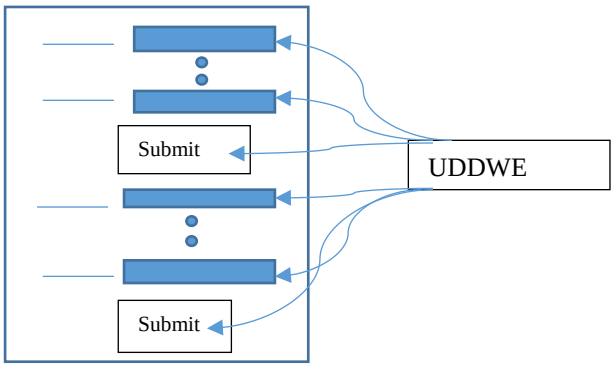


Fig. 2: Auto-Filling and Submitting the Identified Forms.

This approach scores above the existing approaches as it can behave like a traditional search engine giving the user enough freedom and power to choose whatever domain it wishes for, whereas the existing deep web exposers let users only select the domains for which they are designed as illustrated in Fig. 3. The Section 2 discusses the relevant work in the field of hidden web extraction. Section 3 discusses the proposed algorithm. . In Section 4, the conceptual architecture is detailed out. Experiments and Results are discussed in Section 5. Section 6 concludes this paper with the future directions to the implementers of this algorithm followed by the list of references

Copyright © 2018 Manpreet Singh Sehgal, Jay Shankar Prasad. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

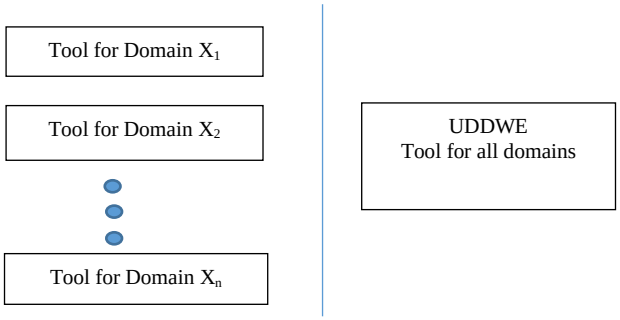


Fig. 3: Abstract Comparison of Existing Tools with the Proposed UDDWE Method. Three Colored Dots are for Continuation.

2. Related work

There have been many attempts to extract hidden web data [1, 2] using various techniques. However, these techniques have been classified as domain specific or vertical search strategies that target few specific domains whereas this paper is the implementation of the horizontal search strategy (covering maximum domains). This section briefly describes attempts to unhide the hidden web data of specific domains.

Sehgal and Anuradha [4] proposed the approach to extract page data using sponger and squeezer modules of their desktop tool (HWPDE) to scrape hidden web table bound data. This tool automatically creates a database and table after detecting data types of the table columns using automatic data type detector (ADTD) tool in it at the client side after harvesting and processing hidden data from a hidden web page. This method assumes the presence of hidden web page containing tabular formatted hidden web data as its input whereas UDDWE produces the hidden web pages of all formats.

Suggestions for mining data records from a web pages have also been made by Liu and Grossman [5] who proposed MDR technique for mining the data records from the Web page. MDR targeted Contiguous and non-contiguous data records and successfully mined both with a remarkable improvements. UDDWE. This method also assumes the presence of hidden web page containing advertisements and other irrelevant content surrounding the main content whereas UDDWE produces the hidden web pages of all formats.

[6] Discussed the method to extract the semantic structure in the web page by proposing a vision based segmentation technique. In this hierarchical semantic structure each node represents a block with the degree of coherence as a value. The UDDWE however generates the set of pages expected by [6].

[7] Delivers the method to identify templates of web page along with the tag structures of a document to extract structured data from deep web sources to present as the results using templates whenever query is fired. However this method is domain bound and does not go beyond one domain whereas UDDWE discussed in this paper covers wider range of domains.

Yalin Wang and Jainying Hu in [8] proposed a machine learning approach to identify tabular data in the web page. Methodology of [8] can be directly applied to the output of UDDWE to filter out the tabular structured webpages from other formats and hence it can serve as the input to HWPDE [4] which works only with the page containing tabular structures.

[9 - 11] proposed the use of dynamic ontologies for information extraction and representation. However the techniques mentioned in them are restricted and depends on the presence of ontologies. UDDWE does not exhibit any restrictions and does not require any ontologies presence.

Andrea et al [12], [16] proposed the way to eliminate noises from web pages to improve the accuracy and efficiency of the process of web content mining.

In [13] the methods to crawl hidden objects using kNN queries are discussed. This approach however does not address universal domain problem.

Researchers in [14], [16 - 19] have demonstrated the existing hidden web search engines and also suggested the approaches to handle various problems. None of them addressed multiple domains individually.

Sehgal and Prasad [15] have proposed algorithmic approach named "ADHWENT" to unhide the deep web using ontologies. All of the above approaches described above except [15] are domain restricted and specialized to uncover the selected domains. However [15] is only an algorithmic attempt using ontologies to overcome this restriction and is dependent on the presence of the specific ontology for the candidate domain. This paper is the result of implementation of the generic domain tool that can cater to wider ranges of domains unlike domain restricted approaches cited in references above.

3. Proposed algorithm

In this paper universal domain algorithm to fetch deep web (UDDWE) to expose hidden web links is proposed which helps in greatly reducing the time spent in knowledge discovery from the repository which is rooted deep into the www and is hidden behind the interfaces.

Following subsections represent the algorithm UDDWE along with the conceptual architecture and the details of its components. The algorithm is organized into three parts namely 'Input' (to fetch domain and statement of domain), 'Process' (for the creation of repositories against the domain name and analysis of statement for fetching the relevant hidden web pages after identifying the hidden web links among the mixed lot produced by Traditional Search Engine) and 'Output' (for storing the fetched pages in local directories and their links in the generated repositories)

3.1. Algorithmic details

The first task in the process of knowledge extraction has always been the mention of interest. In the case of UDDWE this interest mention is in the form of 'domain' and 'statement of domain'. In Algorithm 1 these two inputs are shared in D (domain) and SD (statement of domain) respectively as a result of ask_user function which is the process of user filling in these two details.

Algorithm 1: Ask for domain and Statement of Domain

```
Input (D, SD)
{
D=ask_user (Domain);
SD=ask_user (Statement_of_Domain);
}
```

Once the D and SD are filled in, the process of extracting hidden web starts. The first stage is to identify the entry points. Algorithm 2 identifies these entry points among the result returned by Customized Traditional Search Engine (CTSE). Algorithm 2 is named as Process with three parameters D, SD and k. where D and SD are for 'domain' and 'statement of domain' and k is number of links on the first page of the output of CTSE. TD is a database table and has a structure depicted in Table 1. CreateTable(D) is a function to create a table with the name D if Table with name of D does not exist. Once the TD is created, the process of filling in starts. CTSE (SD) causes customized traditional web crawler to generate various pages of web links corresponding to SD fetched from Algorithm 1. This output is filtered out for only k pages by first_k_entries (CTSE (SD)), and gets stored into the URL column of TD. After this pouring in of k entries is done. The process of identification of form tags from the k links starts. This is accomplished by a call to (Tag_Parser (TD [URL], form) which returns true if the page addressed by link contains form tag. Consequently the type_of_web column of TD is marked as PIW or EpHIW. This process is pseudo-coded in Algorithm 2.

Algorithm 2: Identify entry points

```
Process (D, SD, k)
{
//TD is a database table of domain D, the schema
//is mentioned in Table 1
if TD is Null // TD does not exist
        {
        TD=CreateTable (D);
        }
TD[URL]=first_k_entries (CTSE (SD))
for each tuple in TD
{
        if (Tag_Parser(TD[URL], form)==true)
        {
        TD[Type_of_Web] = "EpHIW"
        }
        else
        {
        TD [Type_of_Web] = "PIW"
        }
}
}
```

Algorithm 3 is named as 'Output' with 'D' and SD as parameters. According to this algorithm once user types the domain in 'D' and selects the SD, UDDWE presents a message to the user that like following are the entry points for the Domain D and the Statement of domain SD.

After this the process of displaying the relevant Entry points to Hidden Web (EpHIW) for the said Domain D on HTML Page (HTML_page) starts. Each such entry is accompanied by the option to 'download and customize the page' (mentioned in Algorithm 4) as well as to open the page in a controlled environment (mentioned in Algorithm 5) so that the auto filling of a form can be performed. The format of the output view is in Table 2 where Action1 and Action2 in the last two columns are the representations of these two options.

Algorithm 3: Presenting the output

```
Output (D, SD)
{
//display the message like following are the entry
//points for the Domain and Statement of domain
for each tuple in TD
{
if (TD [Type_of_Web]=="EpHIW") and (TD[Query]==SD)
{
HTML_Page+= TD [URL];
//provide option for downloading and
//customizing the URL page and opening the link in
//a controlled environment as in Table 2
}
}
```

Algorithm 4 downloads the HTML contents of the URL in the local environment of the client with new nomenclature. File naming conventions are written in Table 3. It takes click event on the 'download and customize' button on the HTML_Page as the input. The need to do such a treatment to the file is important due to the policy of same origin applied to all the web pages of World Wide Web, which states that if the web pages are in the same origin than they can be made to communicate without any restrictions. So to apply the same origin policy to the search interface we had to change the nomenclature of the files referenced and save them in a local file directory.

Algorithm 4: Download and Customize

Input: HTML_Page hyperlink (download and customize) click event

Output: Hidden webpage entry point (pointing to targeted page) downloaded and customized

1) Save the contents of the targeted page and all JavaScript files referenced in the targeted page in a local file directory with the nomenclature mentioned in Table 3.
2) Following Table 3, Change references of js files in the targeted page.

Algorithm 5 is only possible as a result of "same origin policy" [17]. It takes the click event and the local file retuned by algorithm 4. It opens up the local file in iFrame and at the top it displays the button called "fire query". Once the fire query button is clicked then "search input boxes" are looked for in the local file and query (statement of domain) is filled in and submit button is searched for and clicked to generate a hidden web page.

Algorithm 5: Open in Controlled Environment

Input: Click (Open in a controlled Environment) event, Local file returned by Algorithm 4

Output: Hidden Web Page

1) Open the local file in iFrame
2) Display the "fire query" button.
3) Wait for Click event
4) If clicked, search for input text boxes, paste the analyzed and parsed Statement of Domain in these text boxes and auto click the submit button.
5) The hidden webpage is ready

Table 1 represents the structure of this three column table (T_D). <fetched_URL> is generic term used to represent the extracted URL from the web extractor. PIW/HIW is the mention of the type of web that <fetched_URL> represents (PIW: Publically indexed web, EpHIW: Entry Point of Hidden Web). The term <Statement of Domain> is the user query to UDDWE.

Table 1: Format of the T_D as a URL Store

URL	Type_of_Web	Query
<fetched URL>	<PIW/EpHIW>	<Statement of Domain>

4. Conceptual architecture

Figure 4 shows the conceptual diagram of the UDDWE. Following sections describe the components in detail

4.1. Domain and statement of domain

The entry point of UDDWE is the mention of the domain and the statement of Domain by the knowledge seeker. For example if one needs to know something about some book than the domain is "book" and the statement of domain is a sentence describing the intention for example "A book for Advanced Programming in Python" can be considered as a statement of domain. This type of action is the similar of a kind that a user performs while looking for some information in the traditional search engine like Google / Bing / yahoo etc. The only difference is that here user needs to mention the domain as well so that a repository against the domain can be generated for future searches.of domain is a sentence describing the intention for example "A book for Advanced Programming in Python" can be considered as a statement of domain. This type of action is the similar of a kind that a user performs while looking for some information in the traditional search engine like google / bing / yahoo etc. The only difference is that here user needs to mention the domain as well so that a repository against the domain can be generated for future searches.

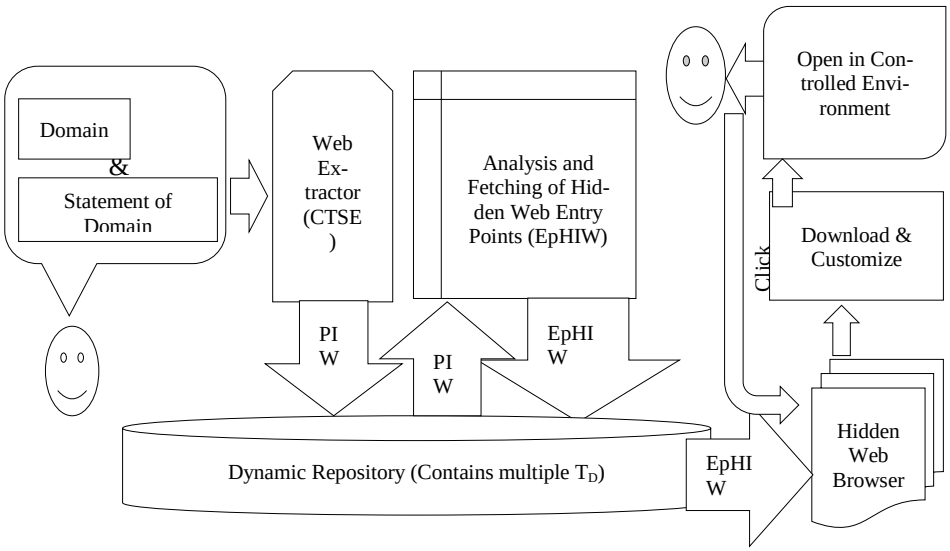


Fig. 4: Conceptual Diagram of UDDWE. CTSE: Customized Traditional Search Engine, PIW: Publically Indexed Web, TD: Table of Domain, EpHIW: Entry Point of Hidden Web

4.2. Web extractor (CTSE)

The mention of domain and statement of domain is fed to the phase where the available Web is searched for the statement of domain using the traditional search engine algorithm. Proposed algorithm makes use of a Traditional search engine customized to our need, we will call it CTSE (Customized Traditional Search Engine) for extracting the indexed pages for the query written in the form of statement of Domain.

4.3. Dynamic repository (contains multiple TD)

The data fetched from the web extractor is searched for URLs and these URLS are stored in the relational database table, against the mention of type and query (statement of domain). Table 1 represents the structure of this three column table (T_D). <fetched URL> is generic term used to represent the extracted URL from the web extractor. PIW/HIW is the mention of the type of web that <fetched_URL> represents (PIW: Publically indexed web, EpHIW: Entry Point of Hidden Web). The term <Statement of Domain> is the user query to UDDWE.

Table 1: Format of the T_D as A URL Store

URL	Type_of_Web	Query
<fetched URL>	<PIW/EpHIW>	<Statement of Domain>

4.4. Analysing and fetching of hidden web entry points (EpHIW)

This phase retrieves the contents of the dynamic repository and do the detailed analysis of each URL stored against the query. This analysis involves the presence of search interface on the page that distinguishes a publically indexed website from the entry point of the hidden website. The presence of such search interfaces can be detected by the presence of a <form> tag with action attribute pointing to the server mechanism to fetch data rooted in the databases (thereby called hidden web databases). After such search form detection, it updates the corresponding table to reflect this information.

4.5. Hidden web browser

This phase brings the aggregated hidden website entry points for the statement of domain asked in the first phase. The resultant format is depicted in Table 2. This phase selects all the hidden web page links for the various queries in the same domain and presents to the user in a tabular format. Where it provides two actions naming download and customize and open in a controlled environment against each query. The table 2 shows the format for displaying k queries with n URLs ($Q_1.URL_n$ being the nth URL for query 1) for query 1 (Q_1) and Last query (Q_k) with the m URLs.

Table 2: Result Format of Hidden Web Page Browser

Query	Hidden web entry point URL	Action 1	Action 2
Q_1	$Q_1.URL_1$ $Q_1.URL_2$ $Q_1.URL_n$	Download and customize	Open in a controlled Environment
Q_k	$Q_k.URL_1$ $Q_k.URL_2$ $Q_k.URL_m$	Download and customize	Open in a controlled Environment

4.6. Download and customize

This phase does save the pages containing search interfaces and the JavaScript files associated with them (if any) in the local domain to avail the facilities of features available to the documents under "same origin policy" [17] thereby introducing the need to change the action attribute of the form tag from relative to absolute so as to avoid typical "file not found" errors due to incorrect mapping of relative to absolute URL.

4.7. Open in controlled environment

This component allows a hidden web page entry point to get opened in iframe embedded in HTML page. It is to be noted here that the source location of the hidden web entry point webpage is the same as that of the UDDWE because "Download and customize" phase has downloaded and made changes in the html to change the relative URLS into absolute URLS (pointing to the local drive and directories). Being able to control the DOM (Document Object Model) elements through PHP scripts is one of the reason of it being called a controlled environment. The input search box and other input boxes are traced out through DOM Explorer. Once having access to these search boxes the same query (after being parsed and analyzed) is fired automatically to fetch the hidden web page.

Copyright © 2018 Manpreet Singh Sehgal, Jay Shankar Prasad. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

5. Experiments and results

Extensive experiments were performed over many domain and their statements to evaluate effectiveness and accuracy of the proposed algorithm. For this purpose we have identified algorithmic complexities of the individual components and in the end the cumulative time complexity of the whole UDDWE. As a point of comparing UDDWE with the existing hidden web crawlers and mechanisms, UDDWE scores higher on the front of covering wider range of domains (almost infinite) but scores lesser on the front of time taken to parse the HTML pages to scrape off URLs and Looking for <form> tags. However this trade-off between time and coverage is justified due to unavailability of the wide domain hidden web explorer. At the time of this writing, the research work in the process of improving this time is being chalked upon. For the sake of completeness, we in the following subsections try to mention the components of this time complexities of UDDWE.

5.1. T (D & SD)

The first component of the proposed work is a combination of two input components (Domain and Statement of Domain). Hence the time complexity is a constant function as the job of storing the values in variables associated with the two input components does happen in a constant time (say c). Hence Eq. (1) refers to this time complexity using the Big Oh notation.

$$T(D \& SD) = O(c) \quad (1)$$

Where 'T' denotes the time complexity, 'D & SD' represents Domain and Statement of Domain' and 'c' is the constant time and 'O' (Big Oh) is the order of function.

5.2. T (CTSE)

This phase calls up the traditional search algorithm (CTSE) for the links which is a mixed collection of documents that might have the presence of search interfaces which are assumed to be the shelters of hidden web pages. The time complexity of the web extractor (indicated as "CTSE" in following equations) is the average time taken by any traditional search engine (Let us call it Activity 1) added to the time taken to scrap web links from the first page (Activity 2). The combined activities take time as represented in Eq. (2) below

$$T(CTSE) = T(\text{Activity 1}) + T(\text{Activity 2}) \quad (2)$$

For 'Activity 1', let's consider $'\mu n'$ as the representation of the average of the time units taken to fetch results for n queries and |P| represents the number of pages (showing publically indexed web links) fetched by CTSE. The Web Extractor in the proposed work scraps the web links presented in the first page retrieved by CTSE. So the time taken by this second activity is "μn / |P|". Hence the calculated time taken by WE is as shown in Eq. (3) below.

$$T(WE) = \mu n + \mu n / |P| \quad (3)$$

5.3. T (analysis & fetching of hidden web entry points)

This component filters the links fetched by Web Extractor, separating the publically indexed web pages which are non- entry points to hidden web pages from the pages which contain the search interfaces as the gateways to the hidden web world. The identification of hidden web entry points is based on the presence of a <form> tag in the html content. This operation is a linear lookup function that is easily accomplished using any standard lookup function of server side technology (e.g., strops () in PHP) in the loaded html contents of the fetched URL. So for all the URLs corresponding to the current query (statement of domain), the pages are opened up and the lookup operation is performed. For n pages for the current query, the lookup operation takes linear time proportional to n. The time complexity for this operation is as mentioned in Eq. (4).

$$T(\text{Analysis of stored urls}) = O(n) \quad (4)$$

The secondary task is to update the table entries to reflect the changed category against the URLs which are identified as hidden web (HIW) entry points. Suppose k out of n total fetched URLs are identified as hidden web entry points, then the time complexity of this task is as given in Eq. (5)

$$T(\text{Fetching of HIW Entry points}) = O(k) \quad (5)$$

Where $k \leq n$. Hence the total complexity is O (n) + O (k).

5.4. T (hidden web browser)

This component takes the linear time proportional to the number of hidden web entry points. So for k hidden web URLs the time taken by this phase is in the order of k as shown in Eq. (6)

$$T(\text{Hidden Web Browser}) = O(k) \quad (6)$$

5.5. T (downloads & customize)

The motive of this component of UDDWE is to bring the page hidden web entry point under the same domain as that of the UDDWE, so that the principle of same origin policy [17] gets applied to the entry point. This action enables automatic filling up of the forms from the keywords specified in the statement of domain [20]. This task involves saving of the URL page with the name. Table 3 presents the naming convention for saving html file and JavaScript files referred in the head and body section of HTML file for the URL.

Table 3: File Naming Conventions

Type of File	Naming Convention
HTML	<domain>_<statement_of_ domain>_<URL index>.html
JavaScript in head	<domain>_<statement_of_domain>_<URL index>_head_<JavaScript file index>.js
JavaScript in body	<domain>_<statement_of_domain>_<URL index>_body_<JavaScript file index>.js

As an example, if the domain is "car" and the statement of domain "get me a new car", then names for saving the html file will be "car_get_me_a_new_car_1.html" for first URL identified as a first hidden web entry point. In addition to this all the JavaScript files found in the body and head needs to be saved in the local file system with the names like "car_get_me_a_new_car_1_body_3.js" and "car_get_me_a_new_car_1_head_2.js for third JavaScript file in the body of html and second JavaScript file in the head of html respectively. DOM tree of the HTML is searched for the action attribute and relative address is changed to absolute address so as to avoid typical file not found error. The various sections of JavaScript files are also searched for the presence of relative URLs (e.g., open methods of AJAX requests) and this phase gets them converted into absolute URLs. The time consumed in this process is larger compared to the other components due to repetitive nature of tasks. For computing the time complexity of this phase some notations are tabulated in Table 4 as follows.

Table 4: Notations for Time Complexity

Sr. No	Notation	Meaning
1	N_{HJS}	Average number of Java Script links in the head tag of Hidden web entry point.
2	N_{BJS}	Average number of Java Script links in the body tag of Hidden web entry point.
3	N_H	Average number of Hidden Web Entry points

After analyzing experiments for various query domain and their statements, the time complexity of this phase is understood to be as mentioned in Eq. (7) below

$$T(D \& C) = \boldsymbol{\theta}\ (N_H X \{N_{HJS} + N_{BJS}\}) \quad (7)$$

Where 'D & C' stands for Download and Customize

6. Experience

To evaluate the effectiveness of our approach, we tested it on various domains with different domain statements. In this section we are illustrating our experience of ten domains with two domain statements as shown in Table 5. We asked common people to type the domain of their choice and write the statements in the style they prefer. We fired these domain and their statements on UDDWE and recorded the resulting hidden web page links in dynamically generated MySQL tables with the name same as that of Domain.
In the process of generating the resulting hidden web links we used customized traditional search engine (CTSE) tailored to our needs to work in the background. CTSE crawled the links related to the domain and its statement in the background and stored first k links into local repository (depicted in Fig. 4 as Dynamic Repository and mentioned in Algorithms as T_D). The value for k is the number of links displayed on the first URL filled HTML page among many retrieved by CTSE (Customized Traditional Search Engine.

Table 5: HIW Entry Points (H) Out of First K Entries

Sr No	Domain	Statement of Domain	K	H
1	Car	Maruti latest car in cng	19	6
		brand new Audi in Gurugram	45	15
2	Hospital	Best Dental Hospital in India	49	14
		Hospital for TB Patients	29	17
3	Care	Cancer Care in North India	39	9
		Day care for toddlers in NCR	26	3
4	Food	Ethnic food in Indore, MP	24	14
		Best Non veg outlets in Sadar bazar Gurugram	34	4 (Failure)
5	Dance	Evening Classical dance classes in Moga	16	8
		Hip hop dance tutorials	17	8
6	Engineering	B.Tech (CSE)	32	8
		M.Tech (IT)	24	9
7	Furniture	Wooden Furnishings in 2BHK	30	14
		Furniture on Sale in NCR	49	8
8	Books	Availability of Class 6, NCERT Mathematic Book.	18	4
		Publishing Houses for publishing engineering books in India	55	18
9	Flights	Cheap flights for kualalumpur from IGI, New Delhi in the month of June, 2018	39	12
		Luggage norms for international and national flights	41	9
10	Cricket	IPL, 2018	32	11
		Indian Squad for the World Cup, 2019	28	12

Table 5 shows the values of k for ten domains that user has chosen along with their statements (two each). This table also shows the resultant H page links identified by the UDDWE as Hidden web page entry points. In the Following subsection 6.1 we elaborate upon the format of the results obtained by our proposed algorithm UDDWE.

6.1. Results

After carefully analyzing the statistics of Table 4 obtained through UDDWE, It was turned out that K is not 100% relevant source as the algorithm which scrapes the links of CTSE outcome also scrapes the advertisement links which are not related to the domain and its corresponding statement. However this error is a fixed constant and for the larger results it is treated as insignificant and hence is easily ignored. But for the smaller collection, this error component (say E) can considerably slow down the whole process. Let us consider that K` is the new K without the error component as defined in Eq. (8)

$$K = K - E \quad (8)$$

Fig. 5 and Fig. 6 represent the relationships between K` and H for the Table 4 datasets. These charts assume the value of E as 3. In the charts obtained in Figure 5 and 6 the term DiQj represents Domain i and Query j from the Table 4 dataset

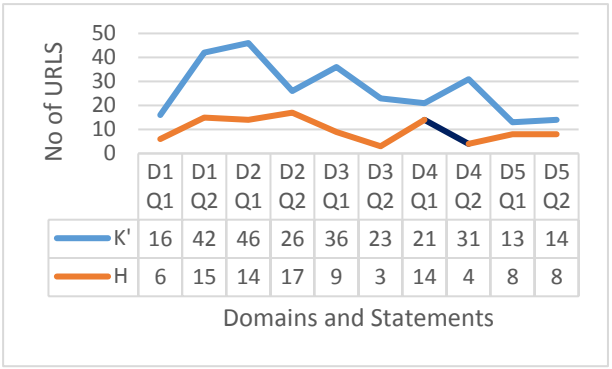


Fig. 5: K' and H Relationship among First Five Domains of Table 4.

From the curves obtained in Figure 5, it appears that the variations in H curve are smoother relative to variations in K'. The black patch leading to D4Q2 is the error in measurement which is caused due to set time limits.

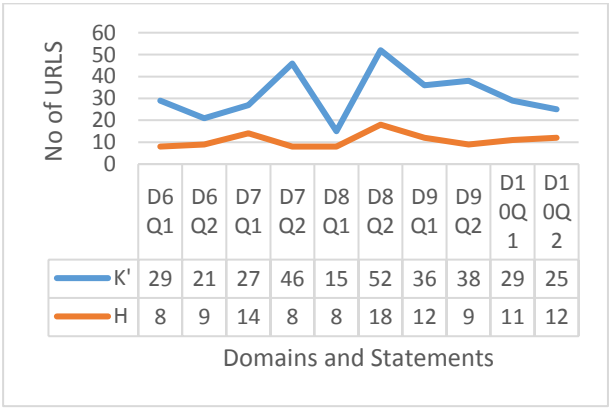


Fig. 6: Depicts the Analysis of Next Set of Five Domains and Their Queries Executed on UDDWE.

A comparison was also conducted with other web search engines. Following Table 6 depicts the resultant analysis of UDDWE with Google, Bing, Yahoo and hidden web extractors.

Table 5: Comparative Analysis of UDDWE with Other Search Engines. PIW: Publically Indexed Web, HIW: Hidden Web

Search Engine	Domain	Web	Coverage	Time
Google	Universal	PIW	Millions	Millisecond
Bing	Universal	PIW	Millions	Millisecond
Yahoo	Universal	PIW	Millions	Millisecond
UDDWE	Universal	HIW	First k Entries (can be extend-ed)	Seconds to Minutes
Other Hidden web search engines	Different Tools for Different Domains	HIW	A few to tens of entries	Seconds

From the above analysis it was clear that UDDWE scores well over other hidden web search engines due to its universal domain feature but it lacks in the time against PIW search engines. At the time of this writing authors are working on this issue. However considering the vastness of hidden web and the efforts expended in searching the quality content is justifying the time taken. The coverage of UDDWE is also scalable provided it is published as a web service.

7. Conclusion and future scope

The experiments proved that UDDWE efficiently unearths the maximum hidden web entry points. UDDWE proposed in this paper can be applied at the front end of [4] to generate the deep web page so as to apply its sponger and squeezer to pour the data into their database. But for this the hidden web page generated by UDDWE must have tabular structure as [4] works only on the tabular structured data. This filtering of tabular structured data pages can be done by using approach suggested by [8]. This interaction is depicted in Fig. 4 where brown colored pages are the tabular structured pages. D and SD are for 'domain' and 'statement of domain'.

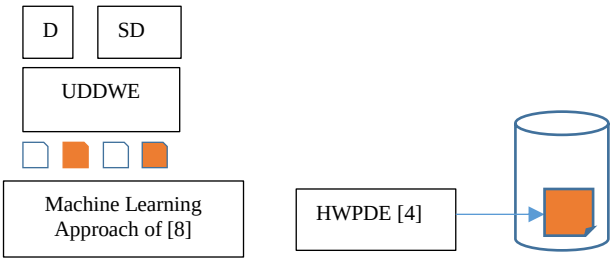


Fig. 7: UDDWE Applied as A Front End to HWPDE [4], D: Domain, SD: Statement of Domain, Colored Pages are Tabular Pages.

Furthermore, UDDWE has a lot of scaling scope if uploaded on the web space provided by big cloud providers, where results and queries will get stored in the cloud database with almost no restriction and thereby knowledge seeker community can greatly benefit from this application. In the cloud computing environment, the single page theory of this proposed work can also be extended to multiple pages being scrapped thereby reaching the maximum potential that UDDWE has in store.

Acknowledgement

Authors would like to thank the entire department of Computer Science and Engineering of MVN University, Palwal and Apeejay Stya University, Gurugram for extending their resources and their contributions to ideas in accomplishing this research.

References

[1] The Deep Web: Surfacing Hidden Value. http://www.completeplanet.com/Tutorials/DeepWeb/.

[2] S. Lawrence and C. L. Giles. Searching the World Wide Web. Science, 280(5360):98, 1998. https://doi.org/10.1126/science.280.5360.98.

[3] S. Lawrence and C. L. Giles. Accessibility of information on the web. Nature, 400:107{109, 1999}.

[4] Manpreet Singh Sehgal and Anuradha. "HWPDE: Novel Approach for Data Extraction from Structured Web Pages." Published in International Journal of Computer Applications (0975-8887) Volume 50 – No. 8. July 2012 pages 22-27.

[5] Bing Liu, Robert Grossman, and Yanhong Zhai. Mining data records in web pages. In KDD ″03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 601–606, New York, NY, USA, 2003.ACM Press). https://doi.org/10.1145/956750.956826.

[6] Cai, D., Yu, S., Wen, J.-R., and Ma, W.-Y. 2003. VIPS: a Vision-based Page Segmentation Algorithm. Tech. Rep. MSR-TR-2003-79, Microsoft Technical Report.

[7] Anuradha, A.K Sharma. "A Novel Technique for data extraction From Hidden Web Databases Published in International Journal of Computer Applications (09758887) Volume 15-No. 4 February 2011 pages 45-48.

[8] YalinWang and Jianying Hu. A machine learning based approach for table detection on the web. In WWW ″02: Proceedings of the 11th international conference on World Wide Web, pages.

[9] Michael Benedikt, Georg Gottlob, and Pierre Senellart. 2011. Determining relevance of accesses at runtime. In Proc. of PODS. 211–222. https://doi.org/10.1145/1989284.1989309.

[10] Andrea Cal`ı and Davide Martinenghi. 2008. Querying Data under Access Limitations. In Proc. of ICDE. 50–59.

[11] Andrea Cal`ı, Davide Martinenghi, and Riccardo Torlone. 2016. Keyword Queries over the Deep Web. In Proc. of ER 2016. 260–268.

[12] Andrea Cal`ı and Umberto Straccia. 2015. A Framework for Conjunctive Query Answering over Distributed Deep Web Information Resources. In Proc. of SEBD. 358–365.

[13] Kevin Chen-Chuan Chang, Bin He, and Zhen Zhang. 2005. Toward Large Scale Integration: Building a MetaQuerier over Databases on the Web. In Proc. of CIDR. 44–55.

[14] Tim Furche, Georg Gottlob, Giovanni Grasso, Xiaonan Guo, Giorgio Orsi, and Christian Schallhart. 2013. The ontological key: automatically understanding and integrating forms to access the deep Web. VLDB J. 22, 5 (2013), 615–640. https://doi.org/10.1007/s00778-013-0323-0.

[15] M.S., Prasad J.S. (2019) All Domain Hidden Web Exposer Ontologies: A Unified Approach for Excavating the Web to Unhide Deep Web. In: Tiwari S., Trivedi M., Mishra K., Misra A., Kumar K. (eds) Smart Innovations in Communication and Computational Sciences. Advances in Intelligent Systems and Computing, vol 851. Springer, Singapore.

[16] Andrea Calí. 2017. Querying and searching the deep web. In Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics (WIMS '17). ACM, New York, NY, USA, Article 3, 1 pages. https://doi.org/10.1145/3102254.3102257.

[17] Gollmann D. (2011) Problems with Same Origin Policy. In: Christianson B., Malcolm J.A., Matyas V., Roe M. (eds) Security Protocols XVI. Security Protocols 2008. Lecture Notes in Computer Science, vol 6615. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-22137-8_11.

[18] D. Song, Y. Luo and J. Heflin, "Linking Heterogeneous Data in the Semantic Web Using Scalable and Domain-Independent Candidate Selection," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 143-156, Jan. 1 2017. https://doi.org/10.1109/TKDE.2016.2606399.

[19] Langley, P., Pearce, C., Barley, M. et al. Mind Soc (2014) 13: 83. https://doi.org/10.1007/s11299-014-0143-y.

[20] Liakos, P., Ntoulas, A., Labrinidis, A. et al. World Wide Web (2016) 19: 605. https://doi.org/10.1007/s11280-015-0349-x.