Designing A General Deep Web Harvester by Harvestability Factor

Mohamamdreza Khelghati, Maurice van Keulen, Djoerd Hiemstra

Databases Group, University of Twente, Netherlands
s.m.khelghati, m.vankeulen, d.hiemstra@utwente.nl

Abstract. To make deep web data accessible, harvesters have a crucial role. Targeting different domains and websites enhances the need of a general-purpose harvester which can be applied to different settings and situations. To develop such a harvester, a large number of issues should be addressed. To have all influential elements in one big picture, a new concept, called harvestability factor (HF), is introduced in this paper. The HF is defined as an attribute of a website (HF_{W}) or a harvester (HF_{H}) representing the extent to which the website can be harvested or the harvester can harvest. The comprising elements of these factors are different websites' or harvesters' features. These elements are gathered from literature or introduced through the authors' experiments. In addition to enabling designers of evaluating where they products stand from the harvesting perspective, the HF can act as a framework for designing harvesters. Designers can define the list of features and prioritize their implementations. To validate the effectiveness of HF in practice, it is shown how the HFs' elements can be applied in categorizing deep websites and how this is useful in designing a harvester. To validate the HF_{H} as an evaluation metric, it is shown how it can be calculated for the harvester implemented by the authors. The results show that the developed harvester works pretty well for the targeted test set by a score of 14.783 of 15.

Keywords: Deep Web, Deep Web Harvester, Harvester Performance Evaluation, Harvestability Factor, Harvester Design Framework

1 Introduction

Nowadays, in an information-thirsty environment, the *deep web* concept receives lots of attention. The content hidden behind web forms which is invisible or hidden to general search engines like Google or Yahoo is defined as deep web *[1] (*also known as *hidden* or *invisible web* [2]). Whether the goal of an access approach is indexing more representative content of a website (referred as *Surfacing approach* [3]) or extracting the whole content, harvesters have a crucial role. Covering different domains, and websites increases the need to have a general-purpose harvester which can be applied to different settings and situations. To develop such a harvester, a number of issues like business domain, targeted websites, and the harvesting goals should be considered. Different business domains

and goals could pose diverse characteristics on deep web access approaches. In some domains, a few number of big databases are the main sources of data and in others, data is scattered through a large number of websites. The latter makes it more desirable to have an approach with no need of extra configuration or at least minimal configuration effort for each website. The goal of the harvesting task is also important [1]. If the goal is to extract all the data and the harvester downloads it partially, the harvesting task is not considered successful. However, this might be a success story if the goal is just to obtain a representative set of data [1]. In addition to the domain and harvesting goal, features of deep websites could have great impacts on a deep web access approach. Different website features, from graphical interface to back-end designing and developing techniques could play an important role. If a website is Flash [4], an Applet, or a simple HTML page, it makes a big difference on the access approach design. Without a well-defined list of elements affecting harvesting tasks, having a general deep web access approach seems far from reach.

Contributions As the main contribution, a new concept, called Harvestability Factor (*HF*) is introduced. This concept enables websites' and harvesters' designers of evaluating where they products stand in the harvesting point of view. Through this concept, we also put all the important elements in harvesting deep websites in one big picture. This defines a framework for designing general deep web harvesters trying to cover all the features from different aspects in designing a harvester. Some of these factors are mentioned in the literature and the others are discovered through the experiments by the authors. Having all these important features in one big picture, we evaluate the influence of each of them on harvesting. This helps creating an implementation strategy for harvester. Defining the importance of each feature helps prioritizing features implementations.

Sections In Section 2, the harvestability factor (*HF*) is introduced. The Section 4 introduces the elements of HF_{W} which are also applied for categorizing deep websites. In this section, all the features of deep websites affecting harvesting process are introduced and deep websites are categorized accordingly. In Section 5, the *HF* elements for a harvester are defined. All the requirements for designing a general purpose deep web harvester from general requirements to detailed ones are also discussed. The different approaches applied in literature to meet these requirements are also explored. Having mentioned all the necessary requirements, in Section 6, as a sample of such a general deep web harvester, the designed harvester by the authors of this paper is introduced and both *HF* as a design framework and an evaluation metric are validated. Finally, in Section 7, the conclusions drawn from this work are discussed and future work is suggested.

2 Harvestability Factor

To formalize all important issues in accessing deep web data, a new factor is introduced in this paper as *Harvestability Factor* (*HF*). Although in a harvesting

process, the roles of both harvester and website are intertwined, separate definitions are required by website and harvester designers for better understanding of harvesting processes. Hence, the *HF* is defined as an attribute of a website or a harvester representing the extent to which the website can be harvested (HF_{W}) or the harvester can harvest (HF_{H}).

As it is shown in Formula 2.1, the *HF* of a given harvester (h) is defined by applying function *sum* of multiplying harvester performances for each one of websites' features discussed in Section 4 (percentage of harvester failure shown by Fa) by the importance of that feature. In this formula, Cr_f and Co_f represent the importance of a feature. n is the number of features and k represents the number of general features. The Cr_f represents how critical is the feature for harvesting and the Co_f represents how often this feature is used in the targeted domain. In this formula, general requirements are represented by GF_j. This is discussed more in Section 5.

$$HF_H(h) = \sum_{i=1}^{n}(1 - (h_{Fa_{f_i}} * Cr_{f_i} * Co_{f_i})) + \sum_{j=1}^{k}(h_{GF_j}) \tag{2.1}$$

In Formula 2.2, the HF_{W} is defined for a website by considering its features discussed in Section 4. In this formula, given a website, for each one of its features, the average performance of harvesters and the importance of the feature are multiplied. n is the number of features and m is the number of harvesters considered which can be also one. In this formula, $w_{p_{f_i}}$ represents the absence or presence of the feature in the website.

$$HF_W(w) = \sum_{i=1}^{n}(1 - (w_{p_{f_i}} * (1/m \sum_{j=1}^{m}(Fa_{f_j})) * Cr_{f_i})) \tag{2.2}$$

Assigning accurate values to the weights and features mentioned in these two formulas is beyond the scope of this paper and considered as a feature work. However, in Section (6), using simple approaches to assign values to these parameters, it is shown how these formulas can help in evaluating harvesters and websites. In this paper, it is tried to cover all aspects of the introduced *HF* elements; business domain, harvesting goal, harvester features and websites features to give a design harvester guideline.

3 Related Work

In this paper, two issues are targeted by introducing the *HF*; *HF* as a harvester design framework and *HF* as an evaluation metric for websites and harvesters. Since the introduction of deep web, there has been several attempts to give access to this part of web and improve the existing approaches [5,6,7,8,9,10,3]. In all these approaches, the focus is on harvesters rather than the websites. They try to improve harvester performance by applying new techniques. Although this is essential but it is not enough. In this paper, it is believed that improving efficiency, scalability, robustness, and other requirements of harvesters is not

possible without having all the effecting factors in one big picture. Introducing the *HF* is the first step in this direction. The *HF* helps to study not only the harvesters but also the targeted domain and the websites features in the design process.

Evaluate/Compare harvesters Small amount of work has been done in comparing and analyzing web harvesting tools. The related studies in literature such as the work in [11,12] focus mainly on a limited number of aspects such as capability in dealing with different data formats, capability to record the extracted data, user friendliness, price in market, export formats, ability to manage the anonymous scraping and multi-threading. However, in this study, in addition to these features, a more detailed set of features are introduced. Despite other works, this paper provides a mechanism to produce an evaluation number to each harvester considering a wide range of features from general requirements to detailed capabilities.

4 Elements of Website Harvestability Factor

In this section, different features of websites which are related to harvesting processes are studied. The roles of each feature as defining element of a website *HF* are also mentioned. Each of these features could be also applied for categorizing deep websites from the harvesting perspective. In the extended version of this paper [13], the more detailed descriptions of these elements could be found.

4.1 Web Development Techniques

A number of techniques applied in developing and designing web sites and web pages create challenges for harvesters. These techniques are usually applied to add interactivity to web pages as well as for improving site navigation. In following, there is a list of such techniques.

Embedded Scripting Languages in HTML Pages Embedded scripts in HTML pages can make content in layers either shown or hidden based on a user action, or change in a state. They can even build HTTP requests to fill out and submit a form dynamically. Managing HTML layers, performing redirections, dynamically generating navigations like pop-up menus and creating hidden anchors are some of the issues which could be caused by client-side scripts [4,14]. This prevents harvesters to have the page as it is shown to user.

Session Management Mechanisms In session management mechanism[1], server keeps track of transactions made with a client. Based on this history and information on client resources, server could provide different services to the client. For harvesters, in later access to documents or distributed crawling, this will create problems [4] as client environment changes or session expires.

[1] http://en.wikipedia.org/wiki/Session_(computer_science)

Complex URL Redirections For reasons such as similar or moved domains, manipulating search engines or visitors, or URL shortening, URLs are redirected. This means different responses are given to browser request which results in browser showing a different page. These redirections could happen automatically or manually. It is easier for harvesters to deal with redirections handled on server side unless it is a redirection loop which does not load any page at the end or it is a redirect chain which might take longer time to have the final page returned [4]. Handling the redirections initiated by scripts embedded in page content is a completely different story. Refresh meta tag in HTML, JavaScript redirections, and Frame redirections are examples of these redirections.

Applets or Flash code If Flash or Applet is used for designing whole website, it is almost impossible for harvesters to access its content without running expensive analysis over each item. Nowadays, websites designers avoid these practices in order to make sure their sites are on good terms with crawlers. If only the welcoming page is designed by Flash or Applet, it becomes easier for harvesters. If they are used for advertisements, they can be ignored.

Frames There are also some issues such as frames which can create difficulties for harvesting processes. Detecting the right frame which contains the page content in a multi-frame page is one of the problems created by such issues.

HTML Coding Practices For harvesters which rely on tags, attributes, and also presentation features, HTML code practices become highly important. Having bad-written HTML code (like not closed tags) might cause problems in analyzing page HTML tree and therefore incapability of harvester to extract data. Lacking well-defined ID, class, and other explanatory attributes for items could also make difficulties for harvesters and make them prone to mistakes. Being consistent in coding practices for all pages and data items is also important. For example, if there is IDs for items, it should be the case for all of them or at least a defined set of items (like different categories). In some cases, data from the same category, even with the same presentation template have small differences in the HTML code behind them. This might mislead harvesters.

4.2 Website Policies

Search Policies

Query Interfaces There are a number of different web interfaces classified as keyword-based, form-like, browsing, and a combination of them [15]. Each one of these interfaces creates a different set of requirements for harvester. For example, in a form-like search interface, information on attribute-value bindings or accessing predefined lists of values for attributes could be of great help for harvesters to decide on which queries to send to the search engine. Detecting interfaces and recognizing different features of web forms could be help harvesters.

Knowing that query interface provides different search options like searching by keyword, industry domain, region, or time helps harvester to act efficiently.

Indexing Policies In case of having search feature in a website, it becomes important to know about the indexing policies. For example, with stop-words indexed in a web site, sending a stop-word query is one of the most reliable options to have a response. Also, if there is no limitation on browsing through search results, sending only one stop-word results in a high coverage. In addition to indexing policies regarding stop words, it is important to know which parts of data are indexed. For example, having only titles indexed makes great difference in defining next queries with having whole text of detailed pages indexed. This is the case in generating queries based on most frequent words in visited pages.

Search Queries and Algorithms In response to a query posed to a search engine, websites do not necessarily follow the same principles. In some cases, stop-words are removed from search queries, query phrases are treated in different ways (considered as AND phrase or OR phrase), or number of returned results shown to user is limited. There might be even differences on additional information provided in reply to a query, such as statistics on search results and number of found related answers. There are also websites which put a limitation on the number of queries a client can send.

Navigation In most of websites, a query is sent, search results are displayed and by following each one of those returned results, a detailed page is represented. However, there are situations in which this is not the case. In some websites, in return to a query, a list of categories related to that query are displayed. Following each one of those categories might end up in another subcategory. This makes it difficult for harvester to realize which returned page is a category or actually a detailed page.

Security, Privacy and Legal Policies Answering this question should be one of the first steps in harvesting process: "is it legal to access data, store it and present it to users?". It is also important to note that if login is required by website to access data. Considering website's terms of service to follow the privacy policies is also important. In some websites, the *Robots Exclusion Protocol* is applied which gives instructions about the site to web robots in a file named *Robots.txt*. In case of the existence of such a file and depending on how strict it is asked to be followed, necessary concerns should be considered. Not all the websites welcome bots (harvesters, or crawlers) with open arms. Having recognized bots through traffic monitoring, bot identity declaration, or real person declaration techniques like a CAPTCHA, websites can use various measures to stop or slow them down. Blocking an IP address, disabling web service API, commercial anti-bot services, or using application firewalls are some of these measures. It is also important to note other privacy policies of the website like policy on disclosing aggregate information for analytical purposes by owners of website.

4.3 Data and Content

Type and Format of Residing Data in Data Sources The content of a deep website could be categorized in two different groups [1]: structured data found in almost all shopping websites (products as entities), movie sites, job listings, and etc, and unstructured data like articles and papers. Each of these mentioned data types have different features which could be helpful in harvesters performances. For example, in a website representing structured data, using the features of a data item like company name could help in defining next queries resulting in a more efficient crawl. It is also of a great importance for harvesters to know about different data file formats for pdf, image, or video files. Different data formats need different handlers to download them.

Data Layout How data is represented in web pages affects the harvesters relying on presentation features of data. Different data types in a website could be presented in different ways. Even data items of a same category could be presented differently based on their features. If these differences in presentation are not known to harvester, it will use the same algorithm to extract all data. This might result in extracting none or undesired information. Structural variations on data presentation must be also tolerated by harvesters and treated accordingly. If the data is represented in a structured way like lists or tables or it is represented in text or a combination of both, harvester should treat them differently. It is also important if a data item has fields represented as nested data on pages; for example, "comments" or "scores information". This poses different requirements on extracting and storage of information.

Data Type Formats Including ontologies and text-patterns in the process of extracting data from detailed pages makes it important to investigate how they can affect the harvesting process. Committing to one ontology and following same patterns for same concepts like "dd-mm-yyyy" format for all dates mentioned on the website could affect the configuration and design of the harvester. Also, for example, if the mentioned address format on the website is not the same for all addresses mentioned in the website, it can have a great effect on the harvester configuration.

Information of a Data Item is Scattered in Different Pages Usually, the queries are sent to search engine, returned results are followed and data about desired items is extracted. However, this is not always the case. In some cases data of a interesting data item is scattered in website. In a more common way, general data is presented in the page navigated through search results. However, more detailed information is provided in some other links which is accessible (only) through this detailed page (you need to go to the detailed page and then browse through the tabs or links to access the information you want). Finding these links and extracting information from them could be a challenge for harvesters.

Providing Semantic Annotations (Meta data) The pages may include meta data or semantic markups and annotations. The annotations might be embedded in the pages or organized into a semantic layer [16] stored and managed separately from the web pages. Data schema and instructions from this layer can be retrieved by harvesters before scraping the pages.

Website Content Language Dealing with the language of the targeted website is one of the abilities that the harvesters should have. Some of the approaches applied in harvesters are based on parsing the content of web pages like data patterns. Having this in mind, it should be noted that dealing with Chinese language needs different configurations than English or the Farsi languages. Having different languages in the targeted websites will cause difficulties for these harvesters.

5 Elements of Harvester Harvestability Factor

As mentioned in the Introduction Section, designing a deep web access approach is highly affected by business domains, websites, and the harvesting goals. In the previous section, the features of a website affecting the *HF* is mentioned. In defining the harvestability factor for a harvester, in addition to the ability of the harvester in dealing with each one of the mentioned website features in the previous section, there are also a number of general requirements which should be met by the harvester. These two set of features help us in defining the *HF* for a harvester. Knowing about the techniques and methods applied in each harvester helps in deciding about the harvester performs for each one of the elements. Therefore, in this section, a subsection is dedicated to defining these methods and techniques.

5.1 High Level Requirements

Every general purpose harvester, despite the differences among its goal, and the domain and websites it is targeting, should meet a set of requirements important in all harvesting processes. Being automatic or running with minimal configuration is one of these requirements. Being scalable (applicable to a large number of websites), independent (of business domain, technology, and etc), efficient (with the least possible number of queries, harvests the most possible amount of data), easy to use (configuration and run settings should be easy for users to perform), and resilient to changes both on website content and presentation are other general requirements which should be followed by a harvester. With these features, a harvester should be able to fill in forms efficiently and automatically, extract data/entities from the returned results pages, and store the extracted data[2] [15].

[2] In [15], two more steps are also considered for a harvester; discovering deep web sources of interest, and presenting extracted data to users and providing them with posing query mechanisms.

For all these steps considered in a deep web harvesting process, having an automatic error/change detection helps to improve harvesting process. This enables the harvester of doing an uninterrupted harvest as it becomes capable of detecting and resolving issues like IP based blocking, website failures, and etc. The harvester should be capable of providing firm guaranties about the exhaustive coverage of the harvested part of the Web. Size estimation of deep websites [?] and also defining a stop condition for harvesting process could help in reaching this goal. In monitoring entities on the Web, it becomes highly important if the harvester could be able to keep the data up-to-date. This needs harvesters being capable of detecting new and deleted entities on the Web. While fulfilling these high level requirements, the harvester should be also capable of harvesting the different categories of websites mentioned in Section 4. There are different approaches to meet these requirements in literature. In the following section, these approaches are introduced.

5.2 Harvesting Techniques

To access data behind web forms, a wide range of harvesters are suggested in literature [5,6,7,8,9,10,3]. The differences among these harvesters root from different sources; from applied techniques in each step of harvesting process to the main goal behind the harvester design. In this paper, the focus in categorizing harvesters is on the techniques and tools applied by harvesters to meet the requirements introduced in Subsection 5.1 and Section 4. This categorization helps reasoning why a harvester could work well for a website and not for the other. It also helps to judge about harvester performance on a website with known features before even applying it in practice. If known that harvesters from a category have problems with website with a specific feature, with the features of the website and harvester at hand, it is possible to predict the outcome of harvesting process. In the following, this classification is represented [8] [3].

1. HTML-based harvesters
   In HTML-based harvesters, the harvester relies on a set of different features of document HTML code [17]. To analyze the HTML structure of the pages, the document is translated into a parsing tree. This could be done by using browser controls like Internet Explorer to parse webpages into Data Object Model (DOM) trees. Then, by running a number of pre-defined extraction rules on the tree, the data is extracted.
2. Harvesters based on Natural Language Processing (NLP) techniques
   In these harvesters [18,19], NLP techniques such as filtering, part-of-speech tagging, and lexical semantic tagging are applied to build relationships between phrases and sentences. From these extracted relationships, a number of extraction rules can be derived. These rules are based on syntactic and semantic constraints and help to identify the relevant information within a document.

[3] This categorization is introduced in [8] except number 6 and 7 which are added by authors of this paper.

3. Machine learning based harvesters
   These harvesters [20] rely on a given set of training examples to derive a number of extraction rules. In these techniques, rather than relying on linguistic constraints found in the page, rules are based on features of the structure of the pieces of data.
4. Modeling-based harvesters
   In modeling-based harvesters [21,22], a data model is defined. In this data model, a number of objects, their properties and relationships are defined. Based on this data model and its modeling primitives, points of interest are located in Web pages.
5. Ontology-based harvesters
   In these harvesters [23], the extraction process is based on the data and not the presentation structure. These harvesters need a specific domain ontology. Through domain ontologies, concepts relevant to a particular topic or area of interest are defined and available for harvesters. The ontology-based harvesters use these ontologies to locate ontology's constants present in the page and to construct objects associated with them.
6. Computer vision based harvester
   These harvesters use computer vision techniques in addition to techniques from machine learning to analyze webpages. In these harvesters, the main goal is to identify and extract information from web pages by interpreting them visually as a human being does. Some of these approaches use also the visual features on the deep Web pages [24].
7. Harvesters based on a combination of previous categories. For example, in a harvester based on HTML structure, applying machine learning techniques could help in having more precise extraction results.

6 Harvestability Factor Validation

As mentioned in Section 1, the *HF* can be used for evaluating a harvester in harvesting and a website in being harvested. It was also discussed that this factor can work as a design framework. To validate these claims, a collection of deep websites is studied considering the *HF* elements. The developed harvester by authors of this paper, as an example effort for developing a general purpose harvester, is applied on the test set.

6.1 Test Set

To create the test set for illustrating how deep websites can be categorized based on the *HF* elements and how this can be used in designing a general purpose harvester, a set of websites from the list of top-100 job vacancy websites by [25]. In the selection of websites from this list, the ones including job boards are considered. To extend this test set, a set of Dutch job vacancy websites are also considered. For each of these websites, all the elements of *HF* are studied. To examine the harvester performance on each one of the categories, the harvester is applied on the websites.

6.2 Developed Harvester

The developed harvester is a HTML-based harvester which automates loading of pages in a browser. These features help to resolve the challenges caused by some of the websites' features mentioned in Section 4. For example, to enable the harvester of implementing embedded scripts in HTML pages, the techniques for automating browsers are applied. Also, for selecting the points of interests, HTML-based techniques are considered. These features also help the harvester to meet the general requirements mentioned in Subsection 5.1 like automation, scalability, independency, efficiency, and being easy to use. For efficiency purposes, different query generation mechanism could be applied to have the most amount of data harvested with the least possible number of posed queries. The configuration is limited to entering the template, and XPaths for points of interests. There is also no need to enter a data model for data storage. Given these configurations for each website, high scalability level can be achieved. Domain-independency is also highly achieved through using only HTML-based techniques which also makes it language-independent.

6.3 HF Validation as a Framework

HF as a Design Framework Through this study, it is shown how these websites are categorized by applying the *HF* elements and how this can guide the design and implementation of a harvester. Having studied the set of deep websites and prioritizing the features of deep websites, By applying this harvester on this set of websites, it is shown how these features are effective on harvesting processes in practice.

Results Studying this set of websites from the domain of job vacancies brings a number of facts into light. If we assume this set of websites represents the job vacancies domain, the results can guide the design process by emphasizing on the elements of *HF* faced more frequently. As it can be seen in Table 6.1, embedded scripts, query interfaces, data layouts, and in-persistent data patterns need further attention during the harvester design process.

Being based on browsers enables our harvester to overcome some of the challenges caused by embedded scripting languages in HTML pages. This is perfectly valid when there is no change of content based on user interaction with the page. However, presence of scripts changing the content based on user interaction or change of browser or time makes it more difficult for the harvester. Simulating user actions or changes in the page environment and comparing the generated result page with the previous version of the page should be performed to be capable of harvesting the page presented to users. This part is not included in our current version of harvester. However, it is worth mentioning that this type of scripts was not faced in our test collection. So, it was reasonable to postpone the implementation of this feature.

The second most common *HF* element in the test set is detecting query interfaces. In all the cases, our harvester could detect the template and query the

Table 6.1. *Categorizing Websites from the Test Set based on the HF Elements*

Harvestability Factor's Element	**Percentage of Sample Websites Having the Element** $(Common_{feature})$ *and* $Critical_{feature}$	**The Harvester Performance** $HarvestFailure_{feature}$ *=1-success_rate*
Embedded Script in HTML 4.1	100 percent , Very Critical (75/100)	Harvester was successful in dealing with this element for all the cases. *-> 0*
Applet / Flash 4.1	0 percent , Highly Critical (100/100)	This feature is not included. In case of facing this element, harvester fails. *->1*
Data Layout (different layouts) 4.3	26 percent , Critical (50/100)	Harvester needs pre-configuration for different page templates. *-> 0.7*
Navigation (not straight-forward) 4.2	2 percent , Very Critical (75/100)	Successful (can differentiate only BTW search result pages and detailed pages) *-> 0.5*
Muli-page data source 4.3	2 percent , Critical (50/100)	Harvester needs pre-configuration. *-> 0.8*
Search Policies (limited search results) 4.2	14 percent , Critical (50/100)	Using different query generation mechanisms resolves this situation *-> 0.1*
Indexing Policies (not stopwords) 4.2	10 percent , Critical (50/100)	Harvester detects if stopwords are indexed or not and sends next queries accordingly *-> 0.1*
HTML Coding Practices (not persistent) 4.1	0 percent (all sample websites are persistent in coding) , Critical (50/100)	(problem as it is HTML-based) *-> 0.8*
Security / Privacy / Legal Policies 4.2	0 percent (no websites with username, pass, or limitation for bots), Very Critical (75/100)	*1*
URL Redirection 4.1	14 percent , Critical (50/100)	*0*
Residing Data (text, no structure) 4.3	10 percent , Critical (50/100)	*0*
Session Management 4.1	2 percent , Very Critical (75/100)	*dealing with cookies -> 0*
Query Interface Type 4.2	100 percent (all have text search or browsing features) , Very Critical (75/100)	*Successful -> 0*
Persistent Data Patterns (not persistent) 4.3	24 percent , Very Critical (75/100)	*Successful if the data layout is defined -> 0.2*
Multi-frames 4.1	0 percent (no website with framing issues) , Very Critical (75/100)	*0.8*

search engines. The other common faced feature is having different data layouts. This is resolved in our harvester by making it possible to define different page templates for each website. However, this might pose a lot of time and effort during configuration phase if there are a large number of page templates used for showing data. In the HTML-based approaches, data can be extracted also based on the content. Therefore, if the data patterns are consistent, a high quality data extraction is still possible through the techniques applied by our harvester. Among the websites in the test set, 15 percent of the websites have limitation on browsing the number of viewed search results. To resolve this problem, different query generation mechanisms are applied which allow efficient harvesting of deep website. The harvester can also detect if stopwords are indexed or not and send the next queries accordingly. These meet two other common *HF* elements mentioned in Table 6.1.

Among the samples, it was observed that users are asked to enable the cookies for the website. This technique is becoming more frequently used by web developers. Therefore, harvesters should accordingly be able to recognize and resolve it. To resolve other session management techniques, keeping the session information and tracking the navigation path to the page are useful. In a not straight-forward search navigation website, which results in more steps than going through search, browsing results page, and viewing the detailed page, the developed harvester could work successfully. This was provided that there are only two types of page templates; search results page, and detailed page templates. The harvester can distinguish only these two types.

As it can be seen in Table 6.1, for some of the *HF* elements, no websites in the test set were found. This might be due to the specifications of the test domain. For example, the application of techniques like Applet or Flash could be seen more frequently in domains like Graphics or Music industries and not so often in job vacancy domain. The same applies to requiring credentials to view job vacancies which is unacceptable in business models of these companies. It is also worth mentioning that defining some of these elements in *HF* for a website is time-consuming and sometimes hard. Persistent coding practices is one of those elements. It is time-consuming to study a website if it follows a persistent coding paradigm unless you face an exception.

6.4 Validation of Harvestability Factor as a Website/Harvester Evaluator

In this part, the ability of the *HF* in evaluating the harvestability of a website or a harvester is discussed. As mentioned in Section 2, assigning values to weights and features in the HF_{W} Formula is beyond the scope of this paper. However, to show how this could be beneficial, we use a simple method to assign these numbers. All the values are assigned with probabilities. We assign the percentage of feature occurrence in the test set as the Co_f. The Fa_f values are calculated through $Fa_f = 1 - success_rate_f$ formula. The $success_rate_f$ are assigned by the authors of this paper based on the capability of the developed harvester in resolving the problems caused by each feature. Introducing a more accurate method for these

values will be studied as future work. The (Cr_f) values are assigned based on the authors experience and the results observed during experiments. They represent how influential is the corresponding feature in the whole harvesting process. We assign them with four values; *Highly Critical (1)*, *Very Critical (0.75)*, *Critical (0.5)*, *Effective not Critical (0.25)*, and *no effect (0)*. Having assigned all the present parameters in the formula with values, the HF_H can be calculated. The best score is the result of having all the features equal to 1 which results in 15. Our developed harvester score is 14.783. In the following line, it is shown how this number is calculated. This high number tells us that the harvester works well for this targeted domain. Having nine features absent in the websites tested in this domain gives a high advantage to this harvester. Of course, by having average numbers for each parameter of the *HF* formula, the harvester performance can be tested generally. In this part, we only show the *HF* of the harvester for a limited test set.

$$HF_H(DevelopedHarvester) = 9+(1-(\tfrac{26}{100}\times\tfrac{50}{100}\times\tfrac{70}{100}))+(1-(\tfrac{2}{100}\times\tfrac{75}{100}\times\tfrac{50}{100}))+(1-(\tfrac{2}{100}\times\tfrac{50}{100}\times\tfrac{80}{100}))+(1-(\tfrac{14}{100}\times\tfrac{50}{100}\times\tfrac{10}{100}))+(1-(\tfrac{10}{100}\times\tfrac{50}{100}\times\tfrac{10}{100}))+(1-(\tfrac{24}{100}\times\tfrac{75}{100}\times\tfrac{20}{100}))=\mathbf{14.783}$$

7 Conclusions and Future Work

As discussed in Section 6, the elements of the introduced harvestability factor can categorize deep websites based on their features which are important in harvesting process. This enables the owners of deep websites and website designers of evaluating where their products stand from harvesting point of view. This helps them to decide about which measures to take in order to follow their policies whether it is increasing access or limiting it. For harvester designers, the harvestabiltiy factor acts not only as an evaluation metric of how well the harvester can behave in practice dealing with different websites, it also behaves as a framework for designing deep web harvesters. The *HF* provides designer with a thorough list of requirements they should meet and also helps to prioritize the features to be addressed and included in the harvester. Categorizing deep websites based on their harvestabiltiy factors makes it feasible to understand the importance of different websites' features. This helps to prioritize the features to be addressed and included in the harvester.

Having the *HF* as a comparison metric for different deep web harvesters is another advantage of this introduced concept. To show how this can be applied, we tested the formula for our own developed harvester on a predefined set of job vacancy websites. To enable the measuring, we applied simple methods in assigning values. The importance of each element was judged in a combination of author experience and expertise with the frequency of usage of that element among the test set websites. Having more than half of the elements absent among the websites gave an advantage to the harvester to get a high score. Of course, this shows that harvester would work very well for this set. However, judging its performance for a bigger or different domain needs new values for each parameter in the formula. This is realized if the average values for each parameter in the formula are assigned. This enables developers to decide what to include in a

harvester and predict the outcome in accurate numbers even before running the harvester on the target websites.

For the next step, we will study how to assign more accurate values automatically. This means having websites classified based on the introduced elements and judging about the importance of each element in a more automatic approach. Also, having average values to calculate the performance of a harvester in more general domains is in the list of our future work. As another future work, we aim at using the *HF* in guiding us in developing a more general deep web harvester. Using the studies performed in this paper and extending them to a bigger test set will help us in deciding on the features our deep web harvester should include and prioritizing their developments.

8 Acknowledgement

We thank the WCC Company for hosting the first author and Jan de Vos, Eliska Went, and Marko Smiljanić for their support, discussions, and valuable input. This publication is supported by the Dutch national program COMMIT.

References

1. Y. He, D. Xin, V. Ganti, S. Rajaraman, and N. Shah, "Crawling deep web entity pages," in *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, (New York, NY, USA), pp. 355–364, ACM, 2013.
2. M. Cafarella, "Extracting and Querying a Comprehensive Web Database," in *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, 2009.
3. J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Halevy, "Google's Deep Web crawl," *Proc. VLDB Endow.*, vol. 1, pp. 1241–1252, Aug. 2008.
4. M. Alvarez, A. Pan, J. Raposo, and A. Vina, "Client-side deep web data extraction," in *Proceedings of the E-Commerce Technology for Dynamic E-Business, IEEE International Conference*, CEC-EAST '04, (Washington, DC, USA), pp. 158–161, IEEE Computer Society, 2004.
5. L. Barbosa and J. Freire, "Siphoning hidden-web data through keyword-based interfaces," in *SBBD*, pp. 309–321, 2004.
6. A. Ntoulas, P. Zerfos, and J. Cho, "Downloading textual hidden web content through keyword queries," in *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '05, (New York, NY, USA), pp. 100–109, ACM, 2005.
7. S. Raghavan and H. Garcia-Molina, "Crawling the hidden web," in *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, (San Francisco, CA, USA), pp. 129–138, Morgan Kaufmann Publishers Inc., 2001.
8. A. H. F. Laender, B. A. Ribeiro-Neto, A. S. da Silva, and J. S. Teixeira, "A brief survey of web data extraction tools," *SIGMOD Rec.*, vol. 31, pp. 84–93, June 2002.
9. G. Weikum and M. Theobald, "From information to knowledge: harvesting entities and relationships from web sources," in *PODS* (J. Paredaens and D. V. Gucht, eds.), pp. 65–76, ACM, 2010.

10. J. L. Hong, "Deep web data extraction," in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pp. 3420–3427, 2010.
11. A. Herrouz, C. Khentout, and M. Djoudi, "Overview of web content mining tools," *CoRR*, vol. abs/1307.1024, 2013.
12. F. Johnson and S. K. Gupta, "Article: Web content mining techniques: A survey," *International Journal of Computer Applications*, vol. 47, pp. 44–50, June 2012. Full text available.
13. S. M. Khelghati, M. van Keulen, and D. Hiemstra, "Designing a general deep web access approach based on a newly introduced factor; harvestability factor (hf)," Technical Report TR-CTIT-14-08, Centre for Telematics and Information Technology, University of Twente, Enschede, June 2014.
14. "Dynamic web page." `http://en.wikipedia.org/wiki/Dynamic_web_page`, 2013.
15. N. Zhang and G. Das, "Exploration of deep web repositories," *PVLDB*, vol. 4, no. 12, pp. 1506–1507, 2011.
16. "What is freeformat." `http://www.gooseeker.com/en/node/knowledgebase/freeformat`, 2013.
17. V. Crescenzi, G. Mecca, and P. Merialdo, "Roadrunner: Towards automatic data extraction from large web sites," in *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, (San Francisco, CA, USA), pp. 109–118, Morgan Kaufmann Publishers Inc., 2001.
18. D. Freitag, "Machine learning for information extraction in informal domains," *Mach. Learn.*, vol. 39, pp. 169–202, May 2000.
19. I. Muslea, S. Minton, and C. A. Knoblock, "Hierarchical wrapper induction for semistructured information sources," *Autonomous Agents and Multi-Agent Systems*, vol. 4, pp. 93–114, Mar. 2001.
20. N. Kushmerick, "Wrapper induction: Efficiency and expressiveness," *Artif. Intell.*, vol. 118, pp. 15–68, Apr. 2000.
21. B. Adelberg, "Nodose - a tool for semi-automatically extracting structured and semistructured data from text documents.," in *SIGMOD Record*, pp. 283–294, 1998.
22. B. Ribeiro-Neto, A. H. F. Laender, and A. S. da Silva, "Extracting semi-structured data through examples," in *Proceedings of the Eighth International Conference on Information and Knowledge Management*, CIKM '99, (New York, NY, USA), pp. 94–101, ACM, 1999.
23. D. W. Embley, D. M. Campbell, Y. S. Jiang, S. W. Liddle, D. W. Lonsdale, Y.-K. Ng, and R. D. Smith, "Conceptual-model-based data extraction from multiple-record web pages," *Data Knowl. Eng.*, vol. 31, pp. 227–251, Nov. 1999.
24. W. Liu, X. Meng, and W. Meng, "Vide: A vision-based approach for deep web data extraction," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 3, pp. 447–460, 2010.
25. "The top 100 websites for your career." `http://www.forbes.com/sites/jacquelynsmith/2013/09/18/the-top-100-websites-for-your-career/`, 2013.