International Journal of Applied Engineering and Technology ISSN: 2277-212X (Online)
An Open Access, Online International Journal Available at http://www.cibtech.org/jet.htm
2017 Vol. 7 (1) January-March, pp.41-47/Keer and Korabu
Review Article

A SURVEY ON WEB CRAWLER AND DEEP WEB INTERFACES

***Anika A. Keer and K.S. Korabu**
Department of Information Technology, Sinhgad College of Engineering, Pune, Maharashtra, India
**Author for Correspondence*

ABSTRACT
The number of web pages available on the Internet is growing vastly day to day. In this case searching relevant information on the Internet is a hard task. Also, most of the information on the internet is hidden behind the query forms that interface to unexplored databases containing high-quality structured data known as the deep web. This information is not accessible through standard search engines. The Web pages in the Deep Web are not indexed by the web crawler. Also, because Deep Web grows at very fast pace, there has been increased interest in techniques that help efficiently locate deep-web interfaces. Deep Web has a very large volume of web resources as well as it is dynamic in nature, so because of this achieving better result is a challenging issue. To solve this problem, there are different techniques defined. In this paper, a survey has been done on different techniques of harvesting deep web interfaces. The paper reviews the work related to different web crawlers and deep web interfaces.

Keywords: *Deep Web Interfaces*

INTRODUCTION
The meaning of the crawler is the one that crawls around the ground. A Web crawler is a computer program that browses the World Wide Web in a methodical, automated manner or in an orderly fashion. This process is called as web crawling. In web crawling, the crawler crawls around the web-pages, gathers and categorizes information on the World Wide Web. The web search engine basically contains three main parts that are, 1. Spider 2. Index and 3. Query Processing. First of all, the spider visits the web pages, then fetches the information and then follows the links to the other pages within a site. The spider returns to crawled site over a regular interval of time. The information that is found in the first stage is given to the next stage that is index stage. The indexing stage is also known as catalog stage. The index is the database which contains a copy of every web page that crawler finds. Whenever, the web page changes then the copy is updated with the new information in the database. The final stage is the query processing stage in which the program that examines millions of web pages recorded in the index finds matches to search and level them in order of what it believes as most relevant.
The deep web which is also known as an invisible web or hidden web is a part of World Wide Web whose contents are not indexed by standard search engines, including password-protected or dynamic pages and encrypted networks (Bergman, 2001). The term "Invisible Web" is actually a contradiction because the information is not invisible, it's just not indexed. The deep web makes up around 96% of all the content on the Internet, which is 500-550 times bigger than the surface web. The surface web is the information on the web that gets indexed by the standard search engines. The major search engines together index only 20% of the Web, thus, they miss 80% of the content. The deep web grows at a very fast pace, therefore, there has been increased interest in techniques that help efficiently locate deep-web interfaces. Besides this deep web pages are very challenging to locate as they are not registered with any search engines. The Deep Web contains a vast amount of valuable information and entities. The information is usually rarely distributed and keeps constantly changing. Therefore, to deal with this problem previous works has proposed two types of crawlers namely generic crawler and focused crawler (Zhao *et al.,* 2016).
Generic crawlers fetch all the searchable forms and do not focus on the specific topic whereas focused crawlers focus on the specific topic and give result which is relevant only to that specific topic. Focused crawler is divided into FFC (Form-Focused Crawler) and ACHE (Adaptive Crawler for Hidden web Entries). The main components of FFC are link, page, form classifiers and frontier manager for focused crawling of web-forms. ACHE extends the focused strategy of FFC with additional components as form

filtering and adaptive link learner. The link classifier helps to achieve higher efficiency for focused crawling (Freire and Barbosa, 2005; Jiang *et al.*, 2009).
Other than efficiency, quality and coverage on relevant deep web sources are also challenging. Crawlers must produce high-quality results. Source Rank ranks the result from the selected sources while FFC and ACHE prioritized the links that bring immediate return and delay the benefit links (Zhao *et al.*, 2016).
In this paper, the research has been done on the different type of web crawler. This survey discusses various web crawling techniques which are used for crawling the deep web. Also, various crawling algorithms are discussed.

Related Work

Chakrabarti *et al.*, (1999) has proposed a new hypertext discovery system called Focused Crawler. The goal is to selectively seek out only those pages which are relevant to the given topic. This helps make the crawl more up-to-date. The focused crawler has three main components that are a classifier, distiller, and crawler. Classifier makes relevant judgments on pages crawled to decide on link expansion. For the classification purpose, Bayes rule is used which is defined as follows.

$$\Pr[c|d, \text{parent}(c)] = \frac{\Pr[c|parent(c)]\Pr\,[d|c]}{\sum_{c' parent(c')=parent(c)} \Pr\,[d|c']}$$

Distiller determines a measure of centrality of crawled pages to determine visit priorities. For focused crawling, distiller needs to control the edge weights carefully. This work has proposed two non-unit weights that are forward and backward edge weight. These forward and backward weights are denoted by E(F) and E(B) respectively. Here, it is proposed that the weight E(F) [u, v] of edge (u, v) be the probability that u is linked to v because v was relevant to the topic.
The final component of a focused crawler is a crawler with dynamically reconfigurable priority controls which is governed by the classifier and distiller. Focused crawler is used to selectively seek out pages that are relevant to a pre-defined set of topics.
Barbosa and Freire (2007) has proposed a new adaptive crawling strategy to efficiently locate the entry points to hidden-Web sources. It also presents a new framework where crawlers automatically learn patterns of promising links and adapt their focus as the crawl progresses. This greatly reduces the amount of required manual setup and tuning. Crawlers learn from new experiences. The crawlers that learn from scratch are able to obtain harvest rates that are similar to or sometimes even greater than manually configured crawlers. This framework can greatly reduce the effort to configure a crawler.
Jiang *et al.*, (2009) has presented a novel machine learning Deep Web crawling approach. The key to Deep Web crawling is to submit promising keywords as a query and retrieve Deep Web content efficiently. The keywords are encoded as a tuple by using its linguistic, statistic and HTML features so that a harvest rate evaluation model can be learned from the issued keywords for the un-issued in future. The structure of the tuple is given as (q, hr) for all issued queries where q is a query and hr is its harvest rate. For estimating the harvest rate of the keyword, an adaptive algorithm for Deep Web crawling is proposed. The basic idea of the algorithm is to estimate harvest rates for unseen keywords generalizing from those previously experienced queries. For this purpose, first the keyword with maximum harvest rate is selected from the candidate set, then the pages are downloaded and parsed and then finally the harvest rate is computed for the current query, likewise, the harvest rate for each keyword in candidate set is estimated. The proposed work needs to renew the training set and the candidate set. This method enables a crawler to utilize diversiform features of keywords and exploit the information to derive promising queries.
Dragut *et al.*, (2009) has described extraction technique that is based on a small set of general design rules which, together allow to extract schema trees with high accuracy. The extraction algorithm combines HTML tokens and the geometric layout of these tokens within a Web page. A token is an atomic visible element on the page. Tokens are classified into several classes amongst which the most important are text tokens and field tokens. A tree structure is derived for text tokens using their geometric layout. Each text token has a complex style attribute, which describes its layout properties: font-colour, background-colour,

font-size, font-style, font-weight, and font-family. The properties are retrieved from the rendering engine of a browser. The text tokens with the similar values for their style properties are clustered together. The first task is to cluster/classify the text tokens appearing on a query interface based on their text-style properties. Headings at the same depth in the heading hierarchy of a document have the same text-style. Also, the labels assigned to the groups at the same depth in the schema tree of query interface to have the same text-style. Second, for any two headings H and h, with h subheading of H, the content area covered by h is a subset of that of H. So, if we knew the area each label covered on the interface then, the hierarchical subordination relationship between the labels in a query interface could be determined by using the inclusion relationship. Another tree structure is derived for the field tokens. The hierarchical representation of a query interface is obtained by iteratively merging these two trees. Thus, the extraction problem is converted into an integration problem. The query interfaces in this method are automatically extracted into an appropriate model.

Dalvi *et al.,* (2011) has proposed an efficient method that uses the Voronoi partitioning of the space by the elements and a nearest-neighbour oracle. This method can be used in the hidden web framework where the goal is to estimate the number of certain objects of interest. Here, each object has a geographic location and the nearest neighbour oracle can be realized by applications such as maps, local, or store-locator APIs. The key technical contribution of this method lies in the algorithm Edge Chase to compute the area of a Voronoi cell of an object using the nearest neighbour oracle. Using this tool an aggregate can be estimated by sampling a random point, finding the nearest object and dividing the value of the function at that object by the area of the Voronoi cell of the same object.

Ma *et al.,* (2012) has described the framework to extract the information from both the surface web and deep web. The work described by the author is suitable for treating AJAX pages and forms. To meet the needs of deep web search, this work has proposed a new structure of crawler, concerned mostly on extracting data from forms. The crawler proposed in this work makes use of some innovative parts like the mainframe extracting module. It also uses the algorithm to distinguish different websites with the same URL by using improved Bayesian classification, to expand the function to AJAX form dealing and so on. Also, Dom Tree is used to make simple and more visual analysis and treatment of downloaded web pages.

Radinsky and Bennett (2012) tackles the problem of predicting significant content change on the web. This work describes how and why the content on the web changes and how to predict such change. There are many techniques for modelling change which have focused simply on past change frequency. But this proposed work defines another technique by additionally studying the efficiency in predicting the change in the page's content, the degree and the relationship among the prediction of the page are observed changes and the relatedness to other pages and the similarity in the types of changes they undergo. This work presents numerous parallel metrics to recognize related pages and focus specifically on measures of temporal content similarity.

Dincturk *et al.,* (2014) has proposed a new methodology called “model-based crawling” that can be used as a basis to design efficient crawling strategies for RIAs (Rich Internet Applications). The framework of Model-based crawling is used to create an actual strategy from the chosen model. To build a model for a given URL, initially, we start with a single vertex representing the starting state of the URL which is reached by loading the URL. Then, the crawler identifies the enabled events in the initial state and explores one of the enabled events. After exploration of each event, the model is augmented by adding a new edge for the newly discovered transition. If a new state is discovered, a new vertex is added to the model. When from each discovered state every enabled event is explored, a complete model is obtained for the URL. In this work, the methodology called “model-based crawling” is introduced to design efficient strategies based on a chosen metamodel. A metamodel is defined by specifying the characteristics of the applications that constitute the instances of the metamodel. Different metamodels can be established using different sets of characteristics. Such a strategy uses the chosen metamodel as a guide for crawling. The strategy initially assumes that the application we are crawling is an instance of the chosen metamodel. A model-based strategy is basically designed in three steps. (1) A metamodel is chosen. (2) A strategy optimized for those applications that follow the metamodel is designed. (3) Steps to take are specified in case the application that is crawled deviates from the metamodel. Based on this chosen model, the created strategy is able to make anticipations about the behaviour of the application and thus discover the states in the application much faster.

Sharma and Khan (2015) has presented a creative framework that is ontology based and is also self-adaptive. It uses the ontology to discover the concept similarity between the known terms of the client inquiry or data need. To semantically expand the search, topic ontology is used for the pre-processing of the focused crawler and also to make search more effective. If the URLs have false tags or if the tags are invalid for the online pages then the outcomes could contradict and if any relevant word is not contained inside of the ontology which will build the page score then likewise the outcome could disagree. To overcome this issue the proposed work utilizes the hybrid methodology which consolidates two routines i.e. Semantic-based String Matching and Static based String Matching to discover the similarity between the semantically relevant words. The linguistics connection is calculated for the search topic and therefore, the website by staring at the linguistic distance between the topic of search and the web page's thought within the domain of the social bookmarking website. Therefore, the space of the search topic and website thought or semantically similar topic, within the ontology of each decides the connection with one another. If the space between the two is a lot, they're less relevant and the other way around. The linguistic distance is calculated by the following formula.

$$S_R(T_i, P_i) = \frac{1}{|dis(T_i, P_i)| + 1}$$

The proposed system is utilizing the shark-search strategy to distinguish the relevant and irrelevant pages. In shark search, the importance of the page is figured on the premise of probability and the score lies somewhere around 0 and 1. This is an unsupervised learning system in which the performance of the web crawler is improved by using the ontology-based learning. It is constantly being updated by dictionary based learning and related words of the named entities.

Quoc *et al.*, (2015) has proposed a new crawling system for multiple geographically distributed sites called as UniCrawl. As the information available on the web keeps growing, to harvest this massive amount of data has become a major challenge. UniCrawl arranges (plan or coordinate the elements to produce desired effects) several geographically distributed sites. Each site operates an independent crawler. They use well-established techniques for fetching and parsing the content of the web. UniCrawl splits the crawled domain space across the sites and merges their storage and computing resources while minimizing the inter-site communication cost. UniCrawl makes use of the map-reduce paradigm. Map-reduce is a paradigm for processing large data sets in parallel on a cluster of workers. Unicrawl works in four phases: Generate phase in which a set of pages are selected to process during the round. The map step computes the subset of pages in the crawl database which were not previously fetched during the previous rounds. These pages are grouped by the host in the reduce phase, and each reducer outputs the k top ranked pages. Then, during the fetch phase, the map step first groups by the host the pages that were generated in the previous phase. In the reduce step, each worker receives a set of hosts together with the list of web pages it needs to retrieve from those hosts. Once the pages are fetched, they are analyzed during the parse phase. This phase consists solely of a map step. During this step, the mapper extracts from the fetched pages stored locally the set of outlinks it contains and adds them to the crawl database. And finally, the update phase. The goal of the update phase is to refresh the scores of pages that belong to the frontier in order to prioritize them. The Unicrawl has implemented in the controlled environment using the ClueWeb12 database.

Sarhan *et al.*, (2015) has proposed an algorithm for increasing the accuracy of feature extraction of keywords. The general-purpose search engines often return so many irrelevant results when users are searching for specific information on a given topic. In this study, two steps for Web Crawling are used to reduce the difficulties while searching on the web. The first step is the feature selection for the datasets used and the second step is web page classification step. A proposed algorithm for feature selection, which uses the Document Frequency technique for the term in the category, is presented. The proposed

DF algorithm gets the weights of chosen keywords in the tested datasets. After determining the datasets it prepares the input by extracting the content on the web page from the dataset, then removes the stop words, stems the keywords and finally determines the term frequency. Then, the algorithm calculates the DF by using two formulas as follows.

$$DF_i^{in} = \frac{n(doc(term_k)_{all}, cat_i)}{n(doc_{all}, cat_i)}$$

$$DF_i^{out} = \frac{n(doc(term_k)_{all}, collection - cat_i)}{n(doc_{all}, collection - cat_i)}$$

Then, the method computes the resulted DF of the required terms by subtracting both the equations. The resulted values represent the weights of required keywords. The keywords are stored with their weights in a database, as a vector, to be classified.
Then, the threshold of the keyword is calculated and the keyword is categorized as positive or negative keyword based on the heuristic that if the weight of the keyword is greater than 1 then it is term as positive else negative.
Then, for page classification, the Support Vector Machine (SVM) technique is used. The difference of the document frequency for positive and negative increased the accuracy of classification stage. The algorithm proposed in this work uses Document Frequency technique and reduces the redundancy during feature selection. It also increases the accuracy of Web page classification.

Comparison between Different Crawling Strategies

Sr.	Method	Concept	Advantages	Limitations
1.	Crawling the large sites first	Crawling starts with the sites with the large number of pending pages, i.e. web pages for crawling.	Large websites are crawled first	When important pages exist in short web site, then this is crawled later.
2.	Breadth First Search Algorithm	Starts at the root URL and searches all the neighbouring URLs at the same level.	Well suited for situations where the objective is found on the upper or shallower parts in a deeper tree.	It will not perform so well in problems like game tree where there are so many branches and all the path lead to same objective with the same length of path.
3.	Depth First Search Algorithm	Starts at the root URL and traverse depth through the child URL.	Well suited for search problems.	When the branches are large then this algorithm might end up in an infinite loop.
4.	Page Rank Algorithm	Download the web pages on the basis of page rank. Page Rank is the measure of relevance of that page determined by counting the number of citations and back links to that page.	In the very limited time important pages are downloaded.	Page rank favours old pages, a new page even though very good one will not have many links unless it is part of an existing site.

5.	Path Ascending Crawling Algorithm	This algorithm crawls each path from the home to the last file of that URL. This nature of the crawler helps to get more information from that site.	Path-ascending crawler is very effective in finding isolated resources, or resources for which no inbound link would have been found in regular crawling.	The main problem in focused crawling is that in the situation of a Web crawler, we would like to be able to predict the similarity of the text of a given page to the query before actually downloading the page.
6.	Online Page Importance Calculation Algorithm	The crawler will download web pages with higher cash in each stage and cash will be distributed between the pages it points when a page is downloaded. The cash of a node records the recent information discovered about the page.	The cash value is calculated in one step and very short duration of time.	Each page will be downloaded many times, that will increase crawling time.
7.	Focused crawling algorithm	In this approach, we can intend web crawler to download pages that are similar to each other, thus it would be called focused crawler or topical crawler.	Attempts to bias the crawler towards pages in some categories in which user is interested.	It is slow as compared to traditional search engines.
8.	Generic crawling algorithm	Genetic algorithm is based on biological evolution whereby the fittest offspring is obtained by crossing over of the selection of some best individuals in the population by means of fitness function.	Best suited when the user has very less or no time at all to spend in searching a huge database and is also very efficient in multimedia results.	This algorithm gives topic relevant as well as irrelevant results unlike focused crawling.
8.	By HTTP Get Request and Dynamic Web Page	It is a query based approach and crawler just download updated web pages after the last visit by using HTTP Get Request.	Web crawler downloads only latest updated web pages.	Both web crawler and web site should follow rules which are stated by the algorithm.

Conclusion

The Deep Web contains a vast amount of valuable information and entities. Harvesting deep web interfaces is the most challenging issue today because they are not registered with any search engines, and are usually sparsely distributed, and keep constantly changing. This paper has discussed different web

Review Article

crawling techniques which are used to harvest deep web interfaces. Also, different algorithms used for crawling are covered in this survey.

REFERENCES

Barbosa L and Freire J (2007). An Adaptive Crawler for Locating Hidden Web Entry Points, *WWW '07 Proceedings of the 16th International Conference on World Wide Web*.

Bergman MK (2001). White paper: The deep web: Surfacing hidden value. *Journal of Electronic Publishing* **7**(1).

Chakrabarti S, van den Berg M and Dom B (1999). Focused crawling: a new approach to topic-specific Web resource discovery. *Journal Computer Networks: The International Journal of Computer and Telecommunications Networking Archive* **31**(11-16) 1623-1640.

Dalvi N, Kumar R, Machanavajjhala A and Rastogi V (2011). Sampling Hidden Objects using Nearest-Neighbor Oracles. In *KDD '11 The 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Diego, CA, USA* ACM 2011 978-1-4503-0813-7/11/08.

Dincturk ME, Vincent Jourdan G, Bochmann GV and Onut IV (2014). A model-based approach for crawling rich internet applications. *ACM Transactions on the Web* **8**(3).

Dragut EC, Kabisch T, Yu C and Leser U (2009). A hierarchical approach to model web query interfaces for web source integration. *Proceedings of the VLDB Endowment VLDB Endowment Hompage Archive* **2**(1).

Freire J and Barbosa L (2005). Searching for Hidden Web Databases, *Eighth International Workshop on the Web and Databases (WebDB 2005).*

Jiang L, Wu Z, Zheng Q and Liu J (2009). Learning Deep Web Crawling with Diverse Features. *IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies 2009*.

Kumar R, Jain A and Agrawal C (2014). Survey of Web Crawling Algorithms. *Advances in Vision Computing: An International Journal (AVC)* **1**(2/3).

Ma W, Chen X and Shang W (2012). Advanced deep web crawler based on Dom. *Fifth International Joint Conference on Computational Sciences and Optimization*.

Quoc DL, Fetzer C, Felber P and Rivi`ere E (2015). Unicrawl: A Practical Geographically Distributed Web Crawler. *2015 IEEE 8th International Conference on Cloud Computing*.

Radinsky K and Bennett PN (2013). Predicting Content Change on the Web. *Proceeding WSDM '13 Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*.

Sarhan AM, Hamissa GM and Elbehiry HE (2015). Proposed Document Frequency Technique for Minimizing Dataset in Web Crawler. *Computer Engineering & Systems (ICCES), 2015 Tenth International Conference*.

Sharma DK and Khan MA (2015). SAFSB: A Self-Adaptive Focused Crawler. *1st International Conference on Next Generation Computing Technologies (NGCT-2015) Dehradun, India*.

Singh AV, Vikas and Mishra A (2014). A Review of Web Crawler Algorithms. *International Journal of Computer Science and Information Technologies* **5**(5) 6689-6691.

Zhao F, Zhou J, Nie C, Huang H and Jin H (2016). Smart Crawler: A Two-Stage Crawler for Efficiently Harvesting Deep-Web Interfaces. *IEEE Transactions on Services Computing* **9**(4).