Articulating the Construction of a Web Scraper for Massive Data Extraction

Shreya Upadhyay[1], Vishal Pant[2], Shivansh Bhasin[3]
Department of Computer Science and Engineering
Graphic Era Deemed University
Dehradun, India
Email: shreya123upadhyay@gmail.com,
vishalpant.pant65@gmail.com,
theturion.bhasin16@gmail.com

Mahantesh K Pattanshetti[4*]
Department of Computer Science and Engineering
Graphic Era Hill University
Dehradun, India
Email: mahant.india@gmail.com

*Author for correspondence

***Abstract*- Massive volumes of data are generated by various users, entities, applications and disseminated online. This copious volume of *big data* is distributed across millions of websites and is available for various applications. Search engines do provide a simple mechanism to access this data. Accessing this data using search engines requires a user to spend time and resources to manually click and download. Clearly, such a manual approach is not scalable for a vast majority of real life applications at the enterprise and organization level. There exist a number of automated approaches to data extraction from the web. Most of these approaches are ad-hoc and domain specific. Therefore, the need for a robust, automated, easy to use framework for extracting content from the web with a minimal human effort across domains appears enticing. The architecture proposed by the authors for a *web scraper* addresses this gap to harvest data from the web. The proposed web scraping framework offers an easy and feasible approach for parsing and extracting data on a large scale from multiple websites with minimal human intervention. This paper provides an insight into issues relevant to constructing a web scraper and concludes by describing the implementation of a web scraper for harvesting learning objects for an eLearning application.**

Keywords - web scraping; web data extraction; web information extraction; knowledge-based systems; web data analysis

I. INTRODUCTION

Slowly but surely the Internet has become a major source for various data needs. It would not be out of order to consider the Internet as a *big data* source. Normally a user submits a query to a search engine and then typically examines the top three to four links to satisfy his/her information requirements. But, consider the scenarios where a researcher in Psychology would like to analyze a number of social media posts to study teenage habits or a company is interested in mining the web for feedback pertaining to the company products or services using the vast contents on the web. Clearly, manually submitting queries and collating the data is cumbersome and an automated solution would be more welcome. The automated solution that comes handy is the development of a web scraper to fetch and download targeted web resources for further processing [1, 2]. A web scraper is an automated software application that is used to extract data from targeted web sources. At a fundamental level, it can be viewed as a robot mimicking the functions of a human by interacting with websites and extracting data stored in it [3, 4].

This paper is organized as follows. In section II, applications, motivation and research challenges involved in automatic web data extraction are discussed. This is followed by a state-of-the-art review of current approaches for the implementation of a web scraper. Section IV discusses the legal and ethical aspects to be taken into account prior to the implementation of any automated web extraction tool. Section V provides an overview of the implementation of a web scraper followed by concluding remarks.

II. APPLICATIONS, MOTIVATION & RESEARCH CHALLENGES

Data extraction from the web can be considered to belong to the class of *big data* problems [5]. From a conceptual viewpoint, two components are employed for web scraping namely a *crawler* and a *scraper*. The task of a crawler is to download the specified web data from targeted sites and the job of the scraper is to extract meaningful content from the crawled data by the elimination of needless content like advertisements, frivolous links, and irrelevant data [2].

A. Applications

In a data-driven world, web scraping offers a novel approach to extract data from the web and utilize it for a large number of data science applications. Information on the web is generally structured using HTML pages. This structuring reveals a good amount of the author's semantic intent that can be used for data analysis applications. Traditionally a research assistant would spend thousands of hours of manual effort to collect data and collate it for analysis. Automated extraction has major cost-time and manpower advantages. In addition, a massive volume of data on the web affords real-time behavioral analysis of human dynamics which may have never been possible before. Primarily web extraction applications can be found in *enterprises* and in the *social* web spheres [4]. A few illustrative applications broadly defining the potential are mentioned below [1, 2].

- A vast amount of data in the form of feedback, reviews, and complaints can be used to monitor sentiments, mine opinions towards brands and companies [4].

978-1-5090-3239-6/17/$31.00©2017IEEE

- Enlarge the scope of data availability for research by harnessing contents from across the globe as opposed to convenient sampling [2].
- For policy research to support real-time feedback on policy measures [2].
- In bioinformatics web services do not provide for data integration from all sources. Certain databases and tools are not compliant with different web services. Web scraping provides an effective mechanism to overcome this drawback [6].
- Commercial advertisements can be strategically placed on the web sites being browsed by a user based on browsing content and context. Google's "*AdSense*" platform is a popular example utilizing this technique [4].
- Comparison shopping and developing competitive intelligence by businesses is another popular example of web data extraction [4].

B. Motivation

The authors are in the process of implementing an eLearning system by harnessing the massive volumes of online learning objects from the Internet. These objects are to be extracted automatically using web data extraction technologies for use in the subsequent phases of the project. Before this data can be used, the data needs to be located, harvested and structured into text files. This data is then to be processed using various NLP and AI techniques for examining the suitability of the contents for meeting eLearning objectives.

C. Research Challenges

In the construction of a web scraper, numerous technical challenges need to be overcome and workarounds facilitated. The aim here is not to enlist or detail all the barriers that a developer may encounter but provide a concise view of the common problems.

- Web content on the same site is distributed across a number of pages. Some of the pagination parameters like "next" button are easy to follow, but at times the use of graphical symbols for pagination can render the extraction of content challenging [3].
- Content loaded statically is easy to mine but content generated by dynamic scripting tools like JavaScript poses its own problems [3].
- Web sites to prevent resource hogging of their valuable server resources employ methods to detect and halt the operation of robots using numerous measures ranging from static CAPTCHAs, obscured images and reverse Turing tests demanding specific actions from the user [3, 7].
- Enabling a cookie on a scraper gives the target site an impression of a human user but the access mechanisms employed by the scraper can be monitored using a cookie and the interaction may be aborted [3].
- It is important to schedule the sequence of URL requests suitably. Too frequent interaction may cause overloading of the target server, the robot to be noticed and result in disconnection [3].
- Human intervention has to be balanced with suitable degree of automation for ensuring accurate performance [4].
- A fundamental requirement of web scrapers is to extract large volumes of data in short span of time. This requirement can at times conflict with honoring target site commitments defined in "*robots.txt*" [4].
- Most significantly the use of web scrapers must satisfy legal and ethical considerations. This requires privacy requirements and copyright restrictions to be duly enforced [4].

In a nutshell the biggest challenge of a building a web scraper is to construct one that functions more like a human and less like a robot.

III. REVIEW OF STATE-OF-THE-ART

Prior to reviewing the state-of-the-art pertinent to web scraping, it is instructive to understand the working of a generic web scraper. Broadly the job of a data scraper is to extract and collate contents in a systematic manner to facilitate further analysis of data. A software robot mimics the actions of human user's web traversal, access to the websites and extracts contents of relevance to the user [6]. A web scraper functioning can be summarized via a three step process.

1. A connection is established to the target site via HTTP protocol. The target website adapts its interaction based on the identification of the robot subject to the robot adhering to the terms of service described in its "robots.txt" file. Areas accessible for data extraction and its scheduling for accessing these resources are intimated to the robot. This ensures server load balancing. Failure on the part of the robot to work per the agreement may result in an end to the interaction, possibly additional penalties like blacklisting or reporting it to the internet service provider.
2. In the next step, the relevant document is retrieved by the robot and the contents are extracted based on HTML parser libraries, regular expressions, programming logic or machine learning approaches.
3. The final step involves the transformation of web content into suitable format per requirements of the application requesting the services of a web scraper.

A number of tools, frameworks and API's both commercial and open source are widely available [3, 4]. Implementation of a web scraper can be achieved by using libraries for popular programming languages, frameworks and workstation environments [6]. All the approaches mentioned above utilize a range of techniques to extract content. One of the most popular approaches to data extraction is to exploit the document structuring afforded by the DOM model for HTML pages [4]. Since a large number of pages are now being

rendered using CSS and JavaScript, the increasing heterogeneity of web pages cannot be adequately handled by tree-based approaches. An alternative approach that is gaining traction is the use of vision based techniques for content extraction. By focusing on rendering rather than on tree structuring they seem to address a wider range of web pages. But vision-based techniques face a couple of major drawbacks. One is that they need to be rendered prior to processing, thereby consuming time when a huge amount of data is to be processed. The second drawback is they ignore structural cues that can be of aid in semantic processing [8]. The tools provided below exploit one of the techniques for data extraction discussed in this paragraph.

It is possible to develop web scrapers using popular programming languages like Java, Python, and Perl by incorporating third party *libraries*. Libraries provide access to the sites and contents are parsed using tools like regular expressions. Libraries typically have two main components. The first component provides support to major HTTP features like authentication, history management, cookie management and SSL certificates. The second component provides parsing (*separating the useful*) support like XPath matching and HTML tree construction [6].

Libraries incorporated into programming languages do come with certain drawbacks. Firstly they need to be incorporated to provide access to the site and secondly a component needs to be configured to parse contents from the web. The aforementioned drawbacks are overcome through *frameworks*. A popular framework is Scrapy (http://scrapy.org) for Python. At the end of each iteration parse function(s) operate on the extracted content [6].

A third approach to web scraping is by using *desktop-based* environments. A characteristic of these support environments is the rich graphical user interface provided, enabling a user to select elements of a web page without the need for sophisticated programming skills for specifying regular expressions or "*XPath*" queries. These environments also provide for facilities to output the content as CSV, text, XML format or to insert content into databases. Compared to the first two approaches they are mostly commercial applications with very limited API facilities [6].

IV. LEGAL CONSIDERATIONS & SCRAPING ETHICS

No discussion can be complete without mentioning the *legal* and *ethical* aspects. It must be noted that use of web scraping is privilege provided by websites and not a right. This implies a huge responsibility and judicious use of web scraping. Operators must be familiar with IP and cyber security laws applicable to their jurisdictions [2].

Certain aspects need to be duly considered. It is incumbent upon the operator of the data extraction tool to adhere to the terms of usage defined by the target website. Legal fraternity has to yet to keep pace with the developments in automatic data extraction due to newness of the domain, dynamic nature of the technology and lack of suitable precedents. There is a lack of adequate case law precedence, thereby implying decision making on a case by case basis. Courts have generally applied case laws applicable in the past to copyright issues for protecting proprietary content. A broad rule of thumb in decision making has been the perceived damage caused to the target site by infringing on its data and/or load placed on its hardware resources [5].

In applications involving the fair use clause, for example, for research, teaching, criticism, or artistic expression reuse is permitted subject generally to restrictions on commercial exploitation. Rulings of legal cases have substantial elements of ambiguity and inconsistency from case to case. To ensure not to attract penal provisions of law it is advisable to scrape only publicly available data that is unencrypted. At times sites do require login and password to access their facilities to discourage access by automated tools; it is safe to assume that access to such sites by a scraper may be considered illegal. In case of any doubt, a safe bet would be to seek written permission from the website provider [2].

There are certain aspects not strictly governed by explicit laws but are guided by ethical conduct. It is the responsibility of the user to ensure that frequency of access restrictions is honored to avoid overburdening resources of the target system [6]. The vast army of automated tools and the consequent burden placed by them on the target web resources has led to restrictions to outright prevention mechanisms being imposed by various organizations. There may be clear instructions on websites to automated tools to avoid specific site areas or the site altogether. Web sites have the right to block automated tools from their site and conversely programming tools exist to override these restrictions. It is advisable to respect the terms of conditions of access and avoid workarounds. Ethics demand that an automated tool refrains from obtaining data from a site unwilling to part with it. At times sites implement *honeypots* to trap malignant tools into collecting irrelevant data [2].

By combining data across multiple sites a researcher is often able to identify the end user. These have great implications for privacy, especially in the medical domain and social science domains. The researcher must ensure that he/she adheres to universal ethical guidelines of conducting research in letter and spirit [4].

V. WEB SCRAPER FOR HARVESTING ELEARNING RESOURCES

The objective of the web data scraper is to harvest data from identified sites and convert it to raw text files. These text files are then processed by an eLearning system using NLP and machine learning techniques to identify their suitability for personalized learning.

Figures 1 and 2 provide an architectural overview and actual working of the web scraper by the means of an illustration respectively. Various components are integrated seamlessly in the form of a service-oriented system. The working of the system can be easily understood by the means of a use case. Assume that a user is keen on learning about 'C' programming language functions from the web. In the first step the user defines a set of keywords, one per line in a text file and submits to the "*BackLink Seeder*". The keywords are executed against a search engine and based on parameter settings about eight to ten web links per keyword are output. These links are then fed to the "*WebSearch Seeder*", which

uses a crawler to extract the contents of the targeted sites. This extracted content is then passed onto a scraper for parsing and the scraped content is output in the form of text files onto a workstation.

Some of the unique aspects of the design are the simplicity of operation, adaptability to a wide range of domains, optimal utilization of resources, applicability to a wide range of popular file formats (HTML, PDF, ODT, Word, XLS etc) and almost instantaneous provision of data as opposed to traditional mechanisms which may take from hours to days for extracting the content. The web scraper has been implemented using Python programming language.

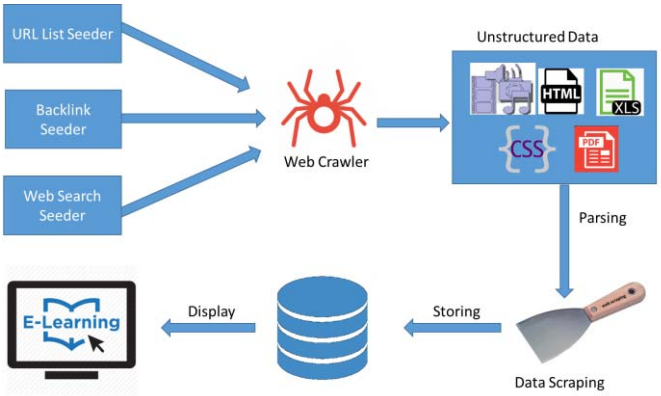

Figure 1: Architecture of a Web Scraper

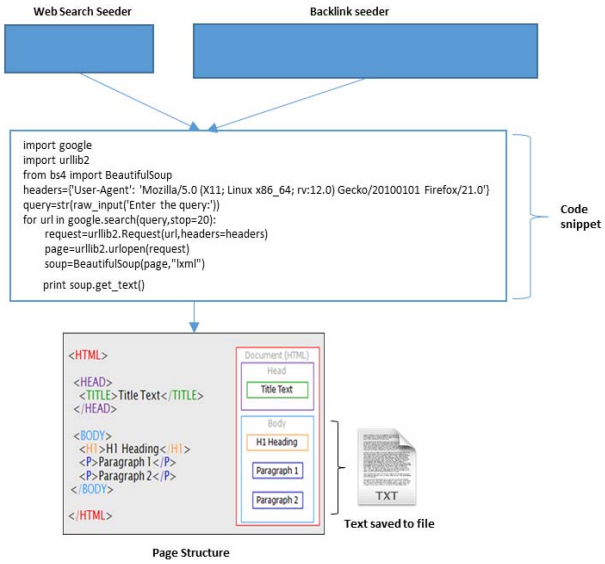

Figure 2: Illustration of the Web Scraper operation

VI. CONCLUSION

The paper makes an attempt to highlight the benefits of automatic data extraction tools and their role as a significant component in the development of knowledge-based systems. The paper commenced by making a case for the web as a massive data source available for a large number of real-world applications and the suitability of using scrapers for data extraction. Different elements relevant to the construction of a web scraper like research challenges, legal considerations, ethical and technical aspects were highlighted. It can be inferred from this paper that constructing a web data extraction tool is a highly technical process requiring expertise and effort. On the flip side, such a system provides a huge return on investment and access to humungous amounts of data which otherwise would require considerable time and human resources [5]. In the age of data, it is an invaluable tool for organizations operating at the enterprise level or for research institutions by providing unprecedented access to volumes of diverse data.

Finally, the authors demonstrated the implementation of web scraping for an eLearning application overcoming traditional limitations of speed, handling of diverse resource formats and applicability across domains.

ACKNOWLEDGMENTS

The authors are grateful to Prof. Kamal Ghanshala president of Graphic Era Society for providing excellent research environment. All the funding support, infrastructure facilities and encouragement provided by the Graphic Era group of institutions is greatly appreciated.

REFERENCES

[1] Schrenk, Michael. *Webbots, spiders, and screen scrapers: A guide to developing Internet agents with PHP/CURL*. No Starch Press, 2012.

[2] Landers, Richard N., Robert C. Brusso, Katelyn J. Cavanaugh, and Andrew B. Collmus. "A Primer on Theory-Driven Web Scraping: Automatic Extraction of Big Data From the Internet for Use in Psychological Research." (2016).

[3] Meschenmoser, Philipp, Norman Meuschke, Manuel Hotz, and Bela Gipp. "Scraping Scientific Web Repositories: Challenges and Solutions for Automated Content Extraction." *D-Lib Magazine* 22, no. 9/10 (2016).

[4] Ferrara, Emilio, Pasquale De Meo, Giacomo Fiumara, and Robert Baumgartner. "Web data extraction, applications and techniques: a survey." *Knowledge-based systems* 70 (2014): 301-323.

[5] Barcaroli, Giulio, Alessandra Nurra, Marco Scarnò, and Donato Summa. "Use of web scraping and text mining techniques in the Istat survey on "Information and Communication Technology in enterprises"." In *Proceedings of Quality Conference*, pp. 33-38. 2014.

[6] Glez-Peña, Daniel, Anália Lourenço, Hugo López-Fernández, Miguel Reboiro-Jato, and Florentino Fdez-Riverola. "Web scraping technologies in an API world." *Briefings in bioinformatics* 15, no. 5 (2014): 788-797.

[7] Vikram, Shardul, Yinan Fan, and Guofei Gu. "SEMAGE: a new image-based two-factor CAPTCHA." In *Proceedings of the 27th Annual Computer Security Applications Conference*, pp. 237-246. ACM, 2011.

[8] Levacher, Killian, Éamonn Hynes, Séamus Lawless, Alexander O'Connor, and Vincent Wade. "A framework for content preparation to support open-corpus adaptive hypermedia." In *International Workshop on Dynamic and Adaptive Hypertext: generic Frameworks, approaches and Techniques*, pp. 1-11 (2009).

**All literature in the reference section above has last been accessed on December 2016, via Google Scholar*