Web Scraping

Chapter · May 2017

DOI: 10.1007/978-3-319-32001-4_483-1

1 author:

Bo Zhao
University of Washington Seattle
60 PUBLICATIONS **1,174** CITATIONS

W

Web Scraping

Bo Zhao
College of Earth, Ocean, and Atmospheric Sciences, Oregon State University, Corvallis, OR, USA

Web scraping, also known as web extraction or harvesting, is a technique to extract data from the World Wide Web (WWW) and save it to a file system or database for later retrieval or analysis. Commonly, web data is scrapped utilizing Hypertext Transfer Protocol (HTTP) or through a web browser. This is accomplished either manually by a user or automatically by a bot or web crawler. Due to the fact that an enormous amount of heterogeneous data is constantly generated on the WWW, web scraping is widely acknowledged as an efficient and powerful technique for collecting big data (Mooney et al. 2015; Bar-Ilan 2001). To adapt to a variety of scenarios, current web scraping techniques have become customized from smaller *ad hoc*, human-aided procedures to the utilization of fully automated systems that are able to convert entire websites into well-organized data set. State-of-the-art web scraping tools are not only capable of parsing markup languages or JSON files but also integrating with computer visual analytics (Butler 2007) and natural language processing to simulate how human users browse web content (Yi et al. 2003).

The process of scraping data from the Internet can be divided into two sequential steps; acquiring web resources and then extracting desired information from the acquired data. Specifically, a web scraping program starts by composing a HTTP request to acquire resources from a targeted website. This request can be formatted in either a URL containing a GET query or a piece of HTTP message containing a POST query. Once the request is successfully received and processed by the targeted website, the requested resource will be retrieved from the website and then sent back to the give web scraping program. The resource can be in multiple formats, such as web pages that are built from HTML, data feeds in XML or JSON format, or multimedia data such as images, audio, or video files. After the web data is downloaded, the extraction process continues to parse, reformat, and organize the data in a structured way. There are two essential modules of a web scraping program – a module for composing an HTTP request, such as Urllib2 or selenium and another one for parsing and extracting information from raw HTML code, such as Beautiful Soup or Pyquery. Here, the Urllib2 module defines a set of functions to dealing with HTTP requests, such as authentication, redirections, cookies, and so on, while Selenium is a web browser wrapper that builds up a web browser, such as Google Chrome or Internet Explorer, and enables users to automate the process of browsing a website by programming. Regarding data extraction, Beautiful Soup is designed for

L.A. Schintler, C.L. McNeely (eds.), *Encyclopedia of Big Data*,
DOI 10.1007/978-3-319-32001-4_483-1

scraping HTML and other XML documents. It provides convenient Pythonic functions for navigating, searching, and modifying a parse tree; a toolkit for decomposing an HTML file and extracting desired information via lxml or html5lib. Beautiful Soup can automatically detect the encoding of the parsing under processing and convert it to a client-readable encode. Similarly, Pyquery provides a set of Jquery-like functions to parse xml documents. But unlike Beautiful Soup, Pyquery only supports lxml for fast XML processing.

Of the various types of web scraping programs, some are created to automatically recognize the data structure of a page, such as Nutch or Scrapy, or to provide a web-based graphic interface that eliminates the need for manually written web scraping code, such as Import.io. Nutch is a robust and scalable web crawler, written in Java. It enables fine-grained configuration, paralleling harvesting, robots.txt rule support, and machine learning. Scrapy, written in Python, is an reusable web crawling framework. It speeds up the process of building and scaling large crawling projects. In addition, it also provides a web-based shell to simulate the website browsing behaviors of a human user. To enable nonprogrammers to harvest web contents, the web-based crawler with a graphic interface is purposely designed to mitigate the complexity of using a web scraping program. Among them, Import.io is a typical crawler for extracting data from websites without writing any code. It allows users to identify and convert unstructured web pages into a structured format. Import.io's graphic interface for data identification allows user to train and learn what to extract. The extracted data is then stored in a dedicated cloud server, and can be exported in CSV, JSON, and XML format. A web-based crawler with a graphic interface can easily harvest and visualize real-time data stream based on SVG or WebGL engine but fall short in manipulating a large data set.

Web scraping can be used for a wide variety of scenarios, such as contact scraping, price change monitoring/comparison, product review collection, gathering of real estate listings, weather data monitoring, website change detection, and web data integration. For examples, at a micro-scale, the price of a stock can be regularly scraped in order to visualize the price change over time (Case et al. 2005), and social media feeds can be collectively scraped to investigate public opinions and identify opinion leaders (Liu and Zhao 2016). At a macro-level, the metadata of nearly every website is constantly scraped to build up Internet search engines, such as Google Search or Bing Search (Snyder 2003).

Although web scraping is a powerful technique in collecting large data sets, it is controversial and may raise legal questions related to copyright (O'Reilly 2006), terms of service (ToS) (Fisher et al. 2010), and "trespass to chattels" (Hirschey 2014). A web scraper is free to copy a piece of data in figure or table form from a web page without any copyright infringement because it is difficult to prove a copyright over such data since only a specific arrangement or a particular selection of the data is legally protected. Regarding the ToS, although most web applications include some form of ToS agreement, their enforceability usually lies within a gray area. For instance, the owner of a web scraper that violates the ToS may argue that he or she never saw or officially agreed to the ToS. Moreover, if a web scraper sends data acquiring requests too frequently, this is functionally equivalent to a denial-of-service attack, in which the web scraper owner may be refused entry and may be liable for damages under the law of "trespass to chattels," because the owner of the web application has a property interest in the physical web server which hosts the application. An ethical web scraping tool will avoid this issue by maintaining a reasonable requesting frequency.

A web application may adopt one of the following measures to stop or interfere with a web scrapping tool that collects data from the given website. Those measures may identify whether an operation was conducted by a human being or a bot. Some of the major measures include the following: HTML "fingerprinting" that investigates the HTML headers to identify whether a visitor is malicious or safe (Acar et al. 2013); IP reputation determination, where IP addresses with a recorded history of use in website assaults that

will be treated with suspicion and are more likely to be heavily scrutinized (Sadan and Schwartz 2012); behavior analysis for revealing abnormal behavioral patterns, such as placing a suspiciously high rate of requests and adhering to anomalous browsing patterns; and progressive challenges that filter out bots with a set of tasks, such as cookie support, JavaScript execution, and CAPTCHA (Doran and Gokhale 2011).

Further Readings

Acar, G., Juarez, M., Nikiforakis, N., Diaz, C., Gürses, S., Piessens, F., & Preneel, B. (2013). Fpdetective: Dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC conference on computer & communications security*. New York: ACM.

Bar-Ilan, J. (2001). Data collection methods on the web for infometric purposes – A review and analysis. *Scientometrics, 50*(1), 7–32.

Butler, J. (2007). Visual web page analytics. Google Patents.

Case, K. E., Quigley, J. M., & Shiller, R. J. (2005). Comparing wealth effects: The stock market versus the housing market. *The BE Journal of Macroeconomics, 5*(1), 1.

Doran, D., & Gokhale, S. S. (2011). Web robot detection techniques: Overview and limitations. *Data Mining and Knowledge Discovery, 22*(1), 183–210.

Fisher, D., Mcdonald, D. W., Brooks, A. L., & Churchill, E. F. (2010). Terms of service, ethics, and bias: Tapping the social web for CSCW research. *Computer Supported Cooperative Work (CSCW), Panel discussion*.

Hirschey, J. K. (2014). Symbiotic relationships: Pragmatic acceptance of data scraping. *Berkeley Technology Law Journal, 29*, 897.

Liu, J. C.-E., & Zhao, B. (2016). Who speaks for climate change in China? Evidence from Weibo. *Climatic Change, 140*(3), 413–422.

Mooney, S. J., Westreich, D. J., & El-Sayed, A. M. (2015). Epidemiology in the era of big data. *Epidemiology, 26*(3), 390.

O'Reilly, S. (2006). Nominative fair use and Internet aggregators: Copyright and trademark challenges posed by bots, web crawlers and screen-scraping technologies. *Loyola Consumer Law Review, 19*, 273.

Sadan, Z., & Schwartz, D. G. (2012). Social network analysis for cluster-based IP spam reputation. *Information Management & Computer Security, 20*(4), 281–295.

Snyder, R. (2003). Web search engine with graphic snapshots. Google Patents.

Yi, J., Nasukawa, T., Bunescu, R., & Niblack, W. (2003). *Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques*. Data Mining, 2003. ICDM 2003. Third IEEE International Conference on, IEEE. Melbourne, Florida, USA.