Volume 5, Issue 5, MAY 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Novel Approach to Web Scraping Technology

Rahul Dhawani, Mrudav Shukla, Priyanka Puvar, Bhagirath Prajapati
A.D. Patel Institute of Technology
Gujarat, India

Abstract— The digital world is growing with a pace that exceeds the speed of any man made fastest prime movers. Here the term growing is used in context to the size of data. At 487bn gigabytes (GB), if the world's rapidly expanding digital content were printed and bound into books it would form a stack that would stretch from Earth to Pluto 10 times. The main contributors to this digital warehouse are social media, government surveillance cameras and plenty of other independent websites which are updated on daily basis such as inventories system of companies, their daily revenues as well as E Commerce websites that comes up with FMCG's on daily basis. In this digital age, this web data is the most essential resource for any business. The main focus of this paper is to highlight the collection of data through scraping as API's are not available for each and every data source.

Keywords— Crawler, Scraper, HTML Tags.

I. INTRODUCTION

Many times data is not easily accessible – although it does exist. As much as we wish everything was available in CSV or the format of our choice – most data is published in different forms on the web. What if you want to use the data to combine it with other datasets and explore it independently? [1] One of the solutions is Screen Scraping. Screen Scraping is the technique to capture the data that is being displayed in human readable format on the destination terminal and to replicate it at the source terminal for further processing. Screen scraping is sometimes referred to as terminal emulation [2].

Though there are other ways to get the data out of the web i.e from web-based APIs, such as interfaces provided by online databases and many modern web applications (including Twitter, Facebook and many others). This is a fantastic way to access government or commercial data, as well as data from social media sites [3]. Another is to extract data fromPDFs. This is very difficult, as PDF is a language for printers and does not retain much information on the structure of the data that is displayed within a document. Extracting information from PDFs is beyond the scope of this paper, but there are some tools and tutorials that may help you do it [3]. But the advantage of scraping is that you can do it with virtually any web site — from weather forecasts to government spending, even if that site does not have an API for raw data access.

However, screen scraping is not an independent process. Before scraping the output, Crawlers are responsible to navigate to the destination terminal. The search key entered at the source machine, engages the crawlers to navigate through the links on the web. Once the crawlers successfully reaches the correct page that matches up with the search string, scraping process starts.

II. TERMINOLOGIES

Screen scraping has certain building blocks. They are as follows:

1. *Python Shell :* As mentioned above, screen scraping consists of essentially two tasks, first is to load the web page and the second is to parse the HTML of the page to locate intended information. These tasks are made simple by python as it offers tools to address these tasks. "urllib2" library is to load the urls and "beautiful soup" library is to parse the web page.
2. *urllib2 : urllib2* is an updated version of urllib. This library has a rich set of functions contained in it to open the urls. In addition to the main link of the web page, useful bits are hidden under hyperlinks on the page. urllib2 even in this case addresses the need.

The urllib2 module defines the following functions [4]:

urllib2.urlopen(*url[, data[, timeout[, cafile[, capath[, cadefault[, context]]]]]]*)

i. url : url can be a string as well as a request object.
ii. data : This is portion is used if there's an additional data to be sent to the server. Currently HTTP requests are the only ones that uses this section[4].
iii. timeout : If there has to be break after certain number of attempts, timeout, which is an optional parameter is used.
iv. cafile / capath: They specifies set of trusted CA certificates for HTTP requests[4]. They are optional.
v. cadefault : This parameter is ignored.

vi. context : This parameter deals with ssl(secure socket layer) and is again an optional parameter.

3. *Anatomy of the web page* : The information is contained in the web page and this information is structured. HTML is responsible for the structure of the web page. Thus to scrape the pages, one needs to have thorough insight on HTML tags for information is contained in between these pages. Say, is used to mark paragraph in the web page or <table></table> is used to arrange the information in a tabular form. There are many other tags between which information is located.

4. *Beautiful Soup :* Once the web page is loaded by the urlopen function of urllib2, there arises the need to sniff around the web page to get specific information. Beautiful Soup library provides objects to parse through the HTML tags of the web page similar to the DOM parser. Say, while scraping reviews from any ecommerce website, these reviews are located between of a certain <table></table> tag, thus beautiful soup parser object will first locate "table" tag and again inside this tag it'll sniff for "p" tag.

Features of beautiful soup can be listed as follows[5]:

i. Beautiful Soup provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree: a toolkit for dissecting a document and extracting what you need. It doesn't take much code to write an application.
ii. Beautiful Soup automatically converts incoming documents to Unicode and outgoing documents to UTF-8.
iii. Beautiful Soup sits on top of popular Python parsers like lxml and html5lib, allowing you to try out different parsing strategies or trade speed for flexibility.

III. AN OVERVIEW OF CRAWLERS

Web crawlers are programs that exploit the graph structure of the Web to move from page to page [6]. The typical aim of the crawler is to exploit the web recursively and save the instance of the pages into local repository. These repositories are then used for further process e.g. building a web search engine. A simple procedure undertaken by a crawler is shown in the figure.

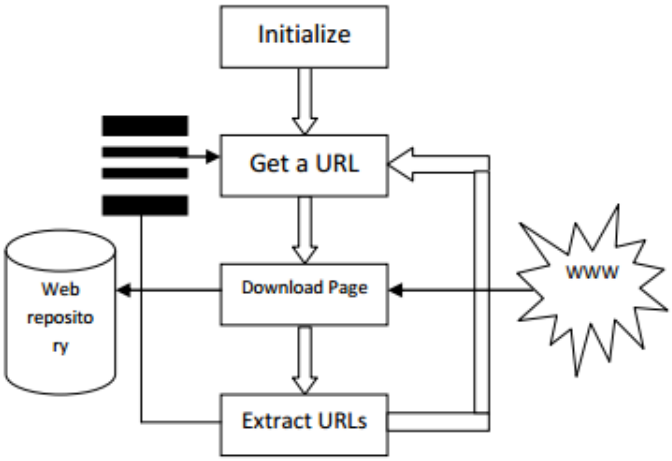

Figure: Flow Diagram of Crawling Process

The working of a web crawler may be discussed as follows [7]:

1. Selecting a starting seed URL or URLs
2. Adding it to the frontier
3. Now picking the URL from the frontier
4. Fetching the web-page corresponding to that URL
5. Parsing that web-page to find new URL links
6. Adding all the newly found URLs into the frontier

Go to step 2 and reiterate till the frontier is empty Thus a web crawler will recursively keep on inserting newer URLs to the database repository of the search engine. So we can see that the major function of a web crawler is to insert new links into the frontier and to choose a fresh URL from the frontier for further processing after every recursive step.
Crawlers can however be categorized into: 1) General Purpose Crawler and 2) Focused Crawler.

1. *General Purpose Crawler:* A general purpose crawler, crawls out almost all the urls that are obtained on the path given by seed i.e. a preselected URL. While employing these crawlers time and space complexity of the results should be considered since these crawlers increases the network traffic.
2. *Focused Crawler:*When the crawler is designed to crawl through only specific pages i.e. one that matches the selection criteria are known as Focused Crawlers. Using focussed crawlers gives a better performance as they does not increase the network traffic.

IV. PROCESS OF SCRAPING

The practice of scraping involves sifting through the seed document (i.e. a web page) and collecting or gathering the required data in the most suitable format. One of the pros of this method is: the structure of the document's data is preserved.

This seed document is parsed by passing it to the BeautifulSoup constructor. In crummy's terms it is referred to as making of the soup. HTML parsing of any document starts with:

```
soup = BeautifulSoup(open("url"))
```

The HTML document is chunked into python objects by Beautiful Soup. These objects are mainly: tag, navigablestring, beautifulsoup and comment[5].

- Tag: These are normal tags found in HTML. Also these tags contains attributes in many cases.
  ```
  <tag @attribute="att1">
  </tag>
  ```
- NavigableString: The text found between the opening and the closing HTML tag of the document is NavigableString object.
  ```
  <tag @attribute="att1">
      Navigable String
  </tag>
  ```
- Beautiful Soup: These object differs from normal HTML tag for it lacks name and attributes that are present in Tag object.
- Comment: Comment is similar to Navigable String.

This sifting of data is explained considering an example seed document. The document "book.html", as shown in the figure, contains the details of different books: year, title, author, publisher and price. Though the entire detail is available, only specific details for the book are required, e.g. title of the book published in 2000.

```
<html>
<title> List of books. </title>
<body>
<p Class: "Year1994">
<h5 Class: "title">TCP/IP Illustrated</h5>
<h5 Class: "author">Steven W. </h5>
<h5 Class: "publisher">Addison-Wesley</h5>
<h5 Class: "price"> 65.95</h5>
</p>
<p Class: "Year2000">
<h5 Class: "title"> Data on the Web </h5>
<h5 Class: "author"> Abiteboul Serge </h5>
<h5 Class: "publisher"> Morgan Kaufmann Publishers </h5>
<h5 Class: "price"> 39.95</h5>
</p>
</body>
</html>
```

Figure: : book.html

Before detailing the algorithm, let's first naively discuss the result. Here the "soup" is made out of the url "book.html". The tag <body> contains details of the two books within <p> tag. The details are wrapped into <h5> tag along with appropriate attributes. The target data to be scraped is contained within the later <p> tag. Over here the main concern is over <h5> tag with the {Class: "title"} attribute since its *navigatable string* consists of the required title of the book. Parsing this way should result in "Data on the Web" as it is published in the year 2000.

Following algorith shows python algorithm that scrapes the needed data.

```
frombs4import BeautifulSoup
soup = BeautifulSoup(open("book.html"))
for all in soup:
    book = all.findAll("body")
for InsideBody in book:
    AllParagraphTags = InsideBody.findAll("p", {"class" : "Year2000"} )
    for InsideParagraphTag in AllParagraphTags
      RequiredTitle = InsideParagraphTag.findAll("h5", {"class" : "title"}).string
print RequiredTitle
```

V. CONCLUSION

This paper thus provides insight of extracting the information from websites in cases where either data is not provided through API's or in cases where it is difficult to fetch the information due to structure or format of the web page. Paper suggests a system containing two major parts. The purpose of the first part is navigating the links directed to towards the

desired web page. The second part is detecting the HTML tags between which required information bits are located. This data is further scraped, printed on source terminal and can be utilized for knowledge discovery or data mining applications to improve business efficacy.

VI. USES

The ever improving technology of scraping has resulted into pros for many key areas of the world. This field with active developments shares a common goal with the semanitic web vision, an ambitious initiative that still requires breakthroughs in text processing, semantic understanding, artificial intelligence and human-computer interactions[8]. Some of the major improvements that are and can be brought about by scraper are enlisted below:

1. *Independence from native APIs :*

Scrapers are not dependent on native API's for information retrieval. A scraper's concern is to retreive human readable parts of the web page and in the WWW, HTML takes the charge of representing the data. The scraper, hence are built to snif these HTML tags of the pages and obtain information from them. Thus the websites that does not provide APIs can now be scraped instead.

2. *Building repositories for web search engine :*

Search engines are basically dependent on crawlers. Crawlers, starting from the seed URL addresses each URL found on its way. Scraper here can save the content of each and every URL provided the machine being employed has amazing time and space complexity.

3. *Analysis Purposes :*

Any business bigwigs concerned with customer service or national organizations like meterological department can use the data collected to enhance their working. For instance meterological department collects the data for future forecasts and here scraper can be used to note down data from almost any city of the world. Also mobile companies can scrape reviews for its new launch from different website and enhance depending on the pattern of satisfaction or dissatisfaction obtained from buyers' reviews.

4. *Categorization or Classification :*

Text categorization (TC – also known as text classification, or topic spotting) is the task of automatically sorting a set of documents into categories (or classes, or topics) from a predefined set[9]. Scrapers are fit to do this task by aiming for the key parts that indicates category of the document. For instance this technology can be employed in eBook libraries for classifying the books based on its contents such as: fiction, sci-fi or mythology.

ACKNOWLEDGMENTS

This research paper was supported by the Project Guide Prof. Bhagirath Prajapati and Prof. Priyanka Puvar as well as the institution A. D. Patel Institute of Technology, Gujarat, India.

REFERENCES

[1] Making data on the web useful: scraping
[2] Screen Scraping: Techopedia
[3] Getting Data from the Web: Data Journalism Handbook
[4] urllib2 — extensible library for opening URLs: https://docs.python.org
[5] Beautiful Soup Documentation – www. crummy.com
[6] *Crawling the Web*, Gautam Pant, Padmini Srinivasan and Filippo Menczer.
[7] *Web Crawler: A Review* , Md. Abu Kausar , V. S. Dhaka Dept, Sanjeev Kumar Singh
[8] *Web Scraping*, Wikipedia.
[9] *Text Categorization* by Fabrizio Sebastiani Dipartimento di Matematica Pura e Applicata Universita di Padova ` 35131 Padova, Italy

