Web-Scraping:
How Charlotte Is A Life Saver Again

Fayang Pan

October 17, 2013

Contents

1 An Evaluation of Web-Scraping **3**

1.1 An Overview of Web-Scraping 4

1.2 Usage of Web-Scraping 4

1.3 Advantages of Web-Scraping 5

1.3.1 Fast 5

1.3.2 Convenient for Data Analysis 5

1.3.3 Accessible 6

1.4 Disadvantages of Web-Scraping 6

1.4.1 Legality 6

1.4.2 Intricate and Dynamic Web Structures 7

1.4.3 Instability 7

2 My Internship **8**

2.1 Overview of My Internship at Pinnacle Solutions, Inc. 8

2.2 How Web-Scraping Became the Best Option for PSI 9

2.3 Scrapy, a Python Framework 10

3 Scrapy Explained in Greater Detail **12**

3.1 Test How Much You Can See 12

3.2 Test If the Data is Scrapable 13

3.3 Iterators 14

Chapter 1

An Evaluation of Web-Scraping

Introduction

Web-scraping is a way to collect existing information from the Internet. It is a nascent, powerful, but disputable subject in the arena of modern technology. Using tools to simulate viewing and downloading of a webpage through browser, the spider (a metaphoric name for the part that goes into each web structure) goes into the webpage and crawl (a metaphor for copy-and-paste) desirable contents into a formatted data structure. Since this technology only requires access to the webpages, theoretically, anything that can be downloaded can be scraped off the Internet.

In the past summer, I worked with Pinnacle Solutions, Inc.[1], a small company in Indianapolis, Indiana, that deals everything with data. My job was solely to develop web-scrapers. In this paper, I will provide an overview and a discussion of web-scraping, followed by a description of my internship experience.

[1]For information, visit http://psiconsultants.com/

1.1 An Overview of Web-Scraping

In general, there are three basic ways to collect data: from primary resources, e.g., conducting surveys and studies; from secondary resources, e.g., databases like the U.S. Census; and from existing pieces of information, e.g., product reviews from *Amazon.com.* Web-scraping focuses on the last way: to harvest data from a structured webpage, e.g., HTML[2], to a structured format, e.g., *.json*[3] and *.csv*[4]. It provides an amalgamation of the deluge of data spread over websites.

1.2 Usage of Web-Scraping

As web-scraping does not generate new data or results, it seems useless. However, people from various backgrounds may find web-scraping very useful. Here are few examples:

If a student were to conduct a study on what lessons people learn from documentaries like *Food, Inc.*, he/she could go to *imdb.com*, *amazon.com* and *rottentomatoes.com* to scrape all the viewers reviews. As there are thousands of reviews for *Food, Inc.*, web-scraping will help the student save much time trying to copy and paste every review. From the integrated results, the students might do a word frequency count to guess what kind of impression the majority have.

If a company were to have launched a product, they could use web-scraping to monitor their product, their competitors products, and their partners products. If Amazon wants to know how successful is the new Kindle Fire HD, they can web-scrape all the reviews about Kindle Fire HD, about Nexus 7, and about Kindle Fire original version. Through analysis, Amazon is able to collect customers feedback promptly.

If a civil service official were to monitor the price change of thousands of commodities,

[2]Abbreviation of HyperText Markup Language
[3]Abbreviation of JavaScript Object Notation
[4]Abbreviation of Comma-Separated Values

and if all these commodities are sold online, the official can use web-scraping to collect the prices. The scraper will just copy the price from the product page of a online retailer website, and paste all of them in a spreadsheet. The shift in prices may reflect inflation/deflation, and the data can be used in economic research.

If a data analyst from Apple were to study how the public responds whenever a new product is released, he/she can collect the review dates from review websites and plot a frequency graph. The number of reviews generate per day may reflect how responsive the customers are to a new product.

1.3 Advantages of Web-Scraping

1.3.1 Fast

Web-scraping allows people to copy and paste from webpages faster. For instance, to harvest 10,000 pieces of Amazon.com product reviews by copying and pasting every little piece of information into an excel spreadsheet, doing it manually will take extended time and be prone to mistakes. Web-scraping technology, on the other hand, makes the process much faster. By specifying certain paths in the webpage structure, the spiders go into specific attributes and fields to crawl information. With the help of web-scraping, 10,000 reviews can be copied and pasted into a *.csv* file within 15 minutes[5].

1.3.2 Convenient for Data Analysis

Web-scraping improves the efficiency of data analysis. More often than not, tables of data are more ready to be processed than plain text. As web-scraping grabs from a structured format, it is able to copy and paste all the information needed into another structured format.

[5]The result is based on running of my own scraper

For instance, after a *.csv* file is generated by scraping data from Amazon product reviews, all cells in the ratings column have a range of 1 to 5 for number of stars, all cells in the review body column will contain texts of the review bodies. A well-structured database will then provide foundation for further data mining and other modeling techniques.

1.3.3 Accessible

Web-scraping is also increasingly accessible today. There are browser extensions like *firebug* and *iMacros*, programs like *Wget*, programming languages like *Perl*, *Hadoop* and *Java* which have libraries to support sending HTTP requests, and programming language extensions like *Scrapy* and *Beautiful Soup* for *Python*. It is becoming more feasible for anyone to learn web-scraping, and to harvest data beyond what traditional ways like web API requests[6] can offer.

1.4 Disadvantages of Web-Scraping

1.4.1 Legality

Web-scraping has been a disputable issue. While it is free to download many webpages, usage of the data within might not be in compliance with the terms and conditions of the source. Currently, there is no law forbidding web-scraping, nor is there any way to ban downloading of webpages(footnote needed, but I cannot find any law passages anyway). Lawsuits were fought between companies like American Airlines against FareChase (http://www.fornova.net/documents/AAFareChase.pdf), but it is impossible to ban people from downloading webpages. Practically, people use free information with citations. Frankly, as activities beyond downloading the webpages are operated locally, it is very hard to track

[6]requesting information through interacting with website's database

what information the user really scraped from. Therefore, there is a grey area in law and practice that people who use web-scraping tools find time and space to develop their projects.

1.4.2 Intricate and Dynamic Web Structures

As more advanced web structures emerge, scraping becomes more difficult. AJAX, for instance, sends out XMLHTTPRequest only when the user performs certain maneuvers on browser. Also, embedded JavaScript contents may not be downloaded. The advent of these dynamic ways of loading web contents brings new challenges to web-scraping.

1.4.3 Instability

Another underlying problem comes from the fact that web-scraping is based on webpage structure. If the website adds a few toolbars, the locations of HTML attributes will change, resulting in crash of the scraper. If the website has an unusual encoding for text, the scraper may crash, too. These potential problems manifest web-scrapings intrinsic dependence of web interface.

Chapter 2

My Internship

2.1 Overview of My Internship at Pinnacle Solutions, Inc.

Pinnacle Solutions, Inc.(PSI) is a data company which helps clients to analyze and interpret their databases. Founded in 1996 by Kalamazoo College alumnus Donald Penix, Jr.(DJ), the company currently has over 20 employees, located in downtown Indianapolis, Indiana. PSI primarily uses *SAS* and *Futrix* [1] for processing data, and *Amazon Web Service* for some server management.

I was a summer intern at PSI in 2011. At that time, DJ was very interested in sentiment analysis, a technology aims to discover and analyze sentiment. For instance, a comment like "I love this iPad!" gives a positive sentiment, and one like "I am sick of iPad!" gives a negative sentiment of iPad. Sentiment analysis will congregate related comments and analyze them, which involves data mining and natural language processing. After research, we found it not feasible, as none of the employees has the expertise in those. So we broke it down into sections, and data collection is the first step towards sentiment analysis.

[1]For more information, see Glossary of Terms

2.2 How Web-Scraping Became the Best Option for PSI

Analysis of data starts from collection of them. In order to harvest usable data from the Internet, there are many ways other than web-scraping.

First, one can extract necessary information through web APIs. Many websites do have APIs available. Twitter, for instance, published a new version of API months ago. Through API, users can request information directly from the webs database, and the data will be clean and formatted.(In Twitters case, *.json* format.) However, to obtain more data, users may need to buy permit/license. Moreover, many websites may have limited information provided in their web APIs, or may not have APIs at all. In that case, even though users can view the data from the browser, they cannot download it through APIs. Therefore, for a small company, it might be expensive and insufficient to rely on APIs for data harvesting.

Second, there are available third-party scraping/data harvesting services. There are applications like *Mozenda*[2], and highly customizable services such as *scrapinghub*[3], *GNIP*[4], and *Open Amplify*[5]. However, these services are not cheap to obtain. Moreover, in the long run, as PSI needs more info from the Internet, the dependency on these services will be ever increasing. Therefore, as a data company, it would be a wise choice if PSI can come up with a low cost, independent system to fetch information that is so close yet so far from them.

Third, as they are certified *SAS* reseller, they could use a software called *SAS Sentiment Analysis Studio*. However, this application is not only expensive, but also dependent on the web API.

At end of the day, a free software capable of fetching data from the Internet was desirable,

[2]http://mozenda.com/
[3]http://scrapinghub.com/
[4]http://gnip.com/
[5]http://www.openamplify.com/

and web-scraping seemed most promising. Many web-scraping resources are free, so the cost of doing web-scraping is low. As many programming languages and their extensions are open sources, many people contribute to the library and trouble-shooting, so the technical support is very active.

After some research, I chose Scrapy, a Python web-scraping framework for the job.

2.3 Scrapy, a Python Framework

To quote from its website, "Scrapy is a fast high-level screen scraping and web crawling framework, used to crawl websites and extract structured data from their pages. It can be used for a wide range of purposes, from data mining to monitoring and automated testing."[6]

Scrapy depends not only on Python, but also few other libraries. Scrapy uses Twisted[7], an event-driven networking engine, and Zope[8], a web application server framework. The installation guide can be found on Scrapy's website[9].

There are few basic ideas or classes in Scrapy framework.

First, spider. *Spider.py* is responsible for going into the webpages and crawl data from then. At the very least, user needs to specify domain, starting url, and where in the webpage to crawl from. If the user wants to scrape from multiple tables or multiple pages, or even conditionally scrape from certain pages, he/she can define those in *Spider.py*

Second, items. *Items.py* is a class specifing the column names in result. In the end, the result would be a table consisting of all the scraped data, and items defines what are they. Items in the same class will be in the same table.

Third, settings. *Settings.py* includes information such as what info will be written into the log file, limit on frequency of requests sent, and other customizable features. In the event

[6]http://scrapy.org/
[7]http://twistedmatrix.com/trac/
[8]http://zope2.zope.org/about-zope-2/what-is-zope-2
[9]http://doc.scrapy.org/en/latest/intro/install.html

of a user login is needed, the user can pre-fill that in the login to avoid access denial.

Forth, pipeline. *Pipelines.py* decides how the scraped data are processed. By default, all data will be dumped in a single file. However, if the user wants to add a filter to get rid of some data, pipeline handles that. If the user wants to split the results from one spider into two seperate files, he/she can do that through coding in *pipelines.py*.

Scrapy has more advanced features, which would fall outside of the scope of this paper. In the next chapter, I would like to offer a high level approach of web-scraping using Scrapy.

Chapter 3

Scrapy Explained in Greater Detail

3.1 Test How Much You Can See

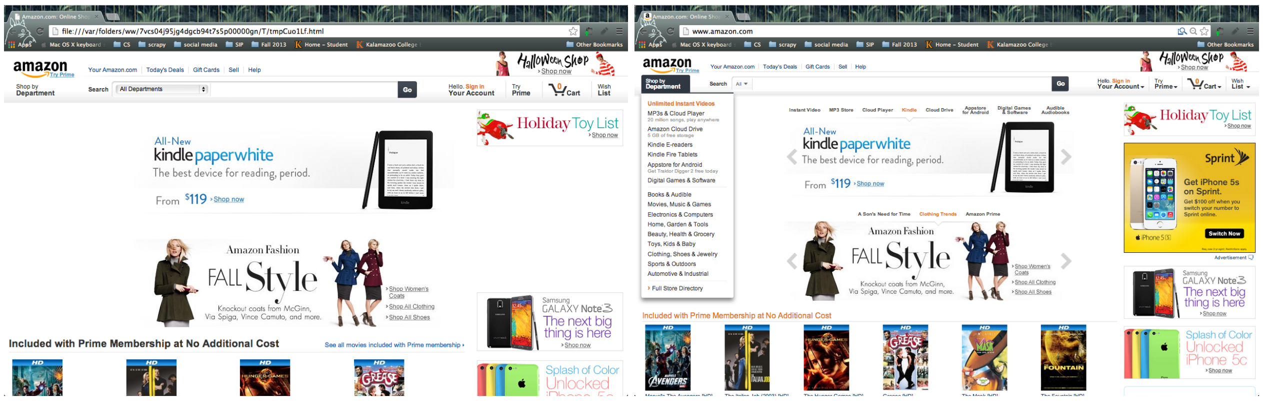
Web-scraping depends on whatever the browser can ”see”, and there are times when the browser sees less than a person does. To check, one can use the following line of code in cmd/terminal

```
scrapy shell http://amazon.com
```

After Scrapy finish analyzing the web address, it goes into its prompt and asks for further instructions. Here, the user can key in

```
view(response)
```

A browser window opens and shows you what the website is like in Scrapy's perspective.


file:///var/folders/ww/7vcs04j95jg4dgcb94t7s5p00000gn/T/tmpCuo1Lf.html
www.amazon.com
amazon
All-New kindle paperwhite
The best device for reading, period.
From $119 › Shop now
Holiday Toy List
Amazon Fashion
FALL Style
Included with Prime Membership at No Additional Cost

What scrapy is able to download locally V.S. what the user can see.

3.2 Test If the Data is Scrapable

While in theory, any data can be seen can be downloaded, and any downloaded data can be scraped, in practice, it is not that optimistic. Even if Scrapy can see the data the user needs, the difficulty of scraping them varies. For instance, in the event of embeded *JavaScript* code, getting the result of a function call will be more difficult.

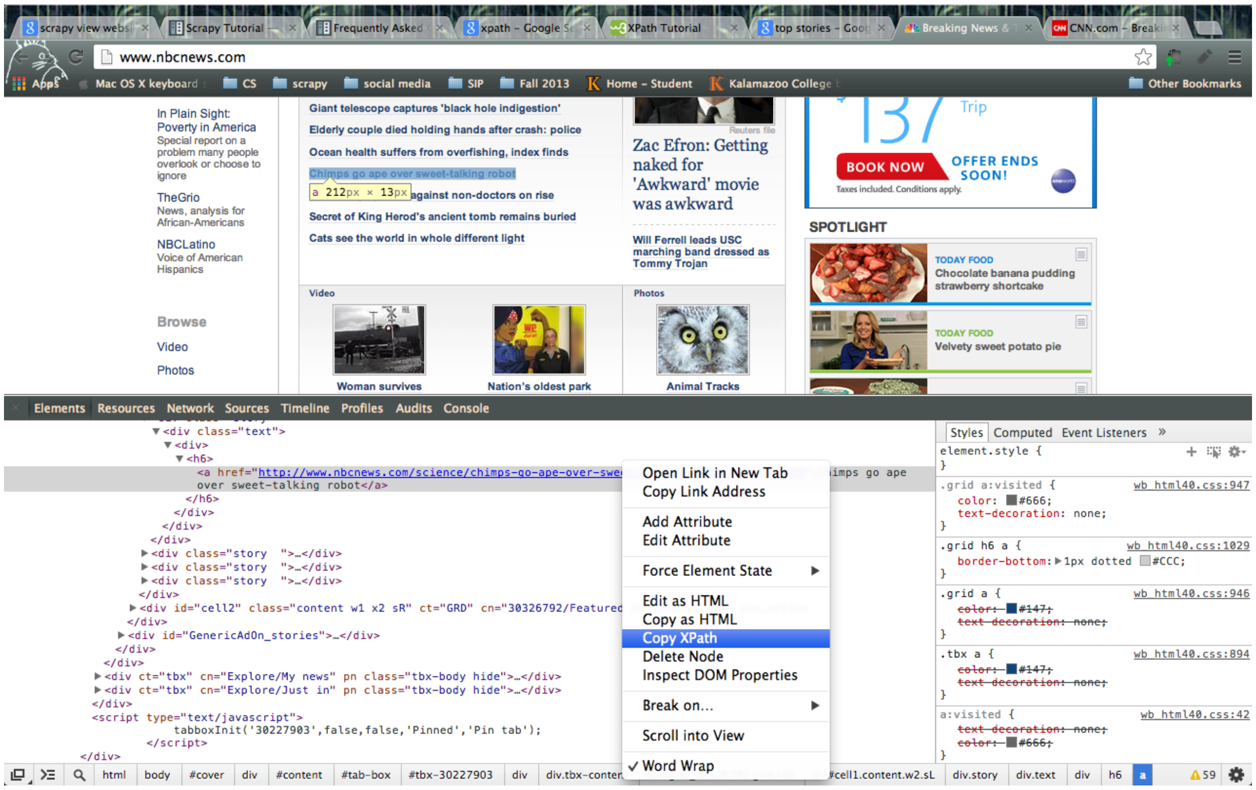
After setting up a project with few commands[1], it is wise to test if certain data are scrapable. Before knowing that, one needs to specify where exactly the data are located. To establish a standard way of referring to a certain field in the webpage, XPath[2] is used.

To retrieve a certain XPath, one can use an built-in *"Inspect Element"* tool, or an extension such as *firebug*[3], available for *Firefox* and *Google Chrome*. Here is an example using *"Inspect Element"* in *Google Chrome* to extract an XPath from *http://www.nbcnews.com*:

[1]For more information, visit http://doc.scrapy.org/en/latest/intro/tutorial.html. The website has a simple example on how to start a project

[2]Abbreviation of "XML Path Language".For more information, see http://www.w3.org/TR/xpath/

[3]Visit http://getfirebug.com/ for more information



```
▼<div class="text">
  ▼<div>
    ▼<h6>
       <a href="http://www.nbcnews.com/science/chimps-go-ape-over-swe…imps go ape
       over sweet-talking robot</a>
     </h6>
   </div>
  </div>
 </div>
▶<div class="story  ">…</div>
▶<div class="story  ">…</div>
▶<div class="story  ">…</div>
</div>
▶<div id="cell2" class="content w1 x2 sR" ct="GRD" cn="30326792/Featured
</div>
▶<div id="GenericAdOn_stories">…</div>
</div>
</div>
▶<div ct="tbx" cn="Explore/My news" pn class="tbx-body hide">…</div>
▶<div ct="tbx" cn="Explore/Just in" pn class="tbx-body hide">…</div>
</div>
<script type="text/javascript">
        tabboxInit('30227903',false,false,'Pinned','Pin tab');
     </script>
</div>
```

```
element.style {
}
.grid a:visited {                 wb_html40.css:947
  color: ■#666;
  text-decoration: none;
}
.grid h6 a {                      wb_html40.css:1029
  border-bottom:▶1px dotted □#CCC;
}
.grid a {                         wb_html40.css:946
  ~~color: ■#147;~~
  ~~text-decoration: none;~~
}
.tbx a {                          wb_html40.css:894
  ~~color: ■#147;~~
  ~~text-decoration: none;~~
}
a:visited {                       wb_html40.css:42
  ~~text-decoration: none;~~
  ~~color: ■#666;~~
}
```

```
html  body  #cover  div  #content  #tab-box  #tbx-30227903  div  div.tbx-conten…  #cell1.content.w2.sL  div.story  div.text  div  h6  a
```

And the XPath of the link is *//*[@id="cell1"]/div[8]/div/div/h6/a.*

After knowing a specific XPath, one can write related code to test if Scrapy can recognize the XPath and scrape the data.If one piece of data is scrapable, it is likely that the entire table that piece is located, can be scraped. If the table is scrapable, it is also likely that similar tables can be scraped, too.

3.3 Iterators

Web-scraping relies heavily on the structure of the webpage. If the webpage is very organize, it is possible for the user to use XPath to find all entries in a table, and send out request to scrape the next page.

Glossary of Terms

Ajax, JavaScript

Amazon Product Reviews

API

Business Intelligence

Crawl

.csv, .json, .xml

data harvesting

data mining

encoding, decoding

HTTPRequest

SAS, Futrix

sentiment analysis

spider

scraper

web-scraping