A review of programming languages for web scraping from software repository sites

[1]Mohan Prakash, [2]Dr. Ekbal Rashid
[1]Ph.d Scholar, Jharkhand Rai University, Ranchi
[2]Associate Professor & HOD, Deptt.of Computer Science & Engineering
Jharkhand Rai University, Ranchi

Abstract - The choice of tool and language for exercising any particular area of work is the foremost task for any researcher and this also holds true for data extraction activity. This paper highlights the pros and cons of four programming languages C, Java, PHP and Python with respect to the libraries and other programming features these languages offer for scraping websites and extraction of data from software repositories. This paper also discusses in brief the related libraries and the methods of using them. The paper can be considered to be a review work of the above mentioned four languages and may help the reader to explore their usefulness and utility with respect to the related task.

Introduction

Faced with the work of collecting data from open source project repositories,analyzing and usingit, the author in the very beginning faced the need to select out a suitable tool to deal with the situation. This followed the study of the numerous tools available for website scraping, their use and comparison to finally decide on a few of them. As the research to select a proper tool advanced, it became evident to the author that every tool comes with some positive and some negative aspect. The choice of the tool or the language for carrying out a particular job of website scraping depends on many factors. The most important of these factors being: the purpose of scraping the particular website, the parameters that will be used to scrape the website, the programming choice of the individual, the platform that is to be used and also to some extent the individual liking of the researcher. The author feels that situations may be such that a particular language may seem to suit a particular stage of work and at another stage another language may be more useful. Some tool may be more useful for a particular type of data extraction activity while as the need changes other tools may become relevant. Looking at the scenario with the vast number of languages and tools available, the author decided to divide the research into two parts, one, a study of languages that would involve coding for data extraction, and two, a study of the tools that may help in data extraction without writing a single line of code. In this paper, the author discusses exclusively about the coded options of data scraping and extraction. In this respect again the options to explore are not few. Study of all such options requires time and extended effort which would probably hinder the main research work. The author hence has explored a selected number of options with the remark that this selection is only of representative nature and in no way exhaustive. Four options have been discussed in this paper. The first is native code – C/C++, then a very powerful object oriented programming language our times, namely Java, the third is a scripting language PHP and lastly a modern and dynamically typed language Python. Notwithstanding the bias that may set in while making such comparisons, the author would like to stress that this discussion is not a total generalization, rather it is a particular area of study mainly related to extracting data from software repositories of collaborative projects for subsequent analysis and research. Readers may feel that the discussion is largely opinionated and thus not adequate from academic perspective. Nevertheless, the possibility of debate is open and such debates will surely have all the potential to channelize the subject in proper direction in the future.

Significance of Research

This research will help one understand which language is better suited for extracting data from software repositories. Obviously this suitability is strictly in terms of the present context of things that is extracting data for the purpose of trying to understand the growth of software repositories. This will help to understand the need to extract data from such places as well as throw light on the various methods that are available for the same. The discussion about each language is accompanied by the pros and cons that the researcher may have to face while dealing with any particular language. Knowing these, the user may select a suitable type of language for his or her own research purpose or may even be able to select different languages for needs at different stages of research. This research may augment the need to exclusively pursue this topic as a separate field of interest, as an area of research in the coming days. This may also usher the urge to look into other areas and bring forth a comparative study of different programming languages that might be used in other fields of study. The significance also lies in the fact that the general features of some programming languages that have been discussed by this paper may be useful in most particular cases too.

Related Work

Ricardo A. and Serrao C. compares different tools like ASP seek, Bixo, Crawler4J, DataParkSearch, Ebot, GNUWget, Grub, Hritrix, HyperEstaier, mnoGoSearch, Nutch, OpenSerarchServer, OpenWebSpider, Sphider and YaCy on the basis of five parameters: language, object oriented or not, indexing, platforms, and database used. The website for Law and Technology Resources for Legal Professionals LLRX opines that "Extracting data from the World Wide Web (WWW) has become an important issue in the last few years as the number of web pages available on the visible internet has grown to over 20 billion pages with over 1 trillion pages available from the invisible web. Tools and protocols to extract all this information have now come in demand as researchers as well as web browsers and surfers want to discover new knowledge at an ever increasing rate . . .". Then it goes on to list a number of tools for extracting information from websites along with their links. This information is very useful,but there is no scope of comparison between the tools here. Grasso G. et al. in a scholarly article comments, ". . . scraping web sites remains a challenging task. Most scraping programs are still developed as ad-hoc solutions using complex stack of languages and tools." and then there is an exhaustive discussion on OXPath which extends Xpath which parses HTML by using CSS properties. Fernandez O. C. in his article talks about some tools for web scraping. Quoting from the same, " . . . Web scraping techniques and scraping tools rely in the structure and properties of the HTML language. This facilitates the task of scraping for tools and robots, stopping humans from the boring repetitive and error prone duty of manual data retrieval. Finally these tools offer data in friendly formats for later procession and integration: JSON, XML, CSV, XLS or RSS. . . .it is recommended that a good scraping tool provides a simple and programmable API. The concept of API is very relevant in this topic." There are some novel tools that can be used for mining software repositories. In an article published by Kim S. et al., the authors have said in the abstract, "In order to analyze software repositories it is necessary to first extract raw data from the version control and problem tracking systems. This poses two challenges: (1) extraction requires a non trivial effort, and (2) the results depend on the heuristics used during extraction. These challenges burden researchers that are new to the community and make it difficult to benchmark software repository mining since it is almost impossible to reproduce experiments done by another team. . . . TA-RE collects extracted data from software repositories in order to build a collection of projects that will simplify extraction process . . ." The paper goes on to discuss TA-RE a tool for data mining. Another very interesting discussion is about Boa which is claimed to be a domain specific language for mining source code. Although this is related to a particular domain, the authors Dyer R. et al., discusses about Boa's Depth First Search strategy and other features.

Discussions

The four languages as mentioned above have been discussed below:

Using the native language C/C++

Out of the many options available, the author identified a three step process for data extraction using native language:

1. Download the html page using the library libcurl:

This library supports HTTP and HTTPS other than other protocols such as FTPS, FTP, TFTP, SFTP, Telnet, IMAP, POP3, SMTP and some more. Currently the library runs in the same fashion on many platforms such as Linux, Solaris, Windows, Symbian, macOS, DOS, Android, Blackberry, Darwin, etc. So we can say that this library is totally portable. The library libcurl also supports many HTTPS certificates, FTP uploading, cookies, Kerberos and much more. The libcurl library has the functionality of SSL/TLS support through OpenSSL and others. It is IPv6 compatible and free. So the first step involves downloading the html page using curl. In Linux the command used is:

```
$ curl <URL>
```

This downloads the page and the output is by default the terminal, but a file can be mentioned where the data is to be saved. From the Linux terminal we can use:

```
$curl <URL>>><filename>
```

2. Convert the downloaded html page to valid well formed xml using libtidy

Developers can integrate libtidy which is a static and dynamic C library with many languages. It is the library on which HTML Tidy works. It is used to tidy up html page. The code using the libtidy library should go through the following steps: initializing document using function tidyCreate, converting it into xml using function tidyOptSetBool, capturing diagnostics using tidySetErrorBuffer, parsing the input using tidyParseString, tidying things up using the function tidyCleanAndRepair and then print if necessary.

3. Parse the xml document using libxml library:

The library libxml is portable and works on Linux, Windows and MacOS but works best on Linux as it is normally included in these distributions. The library libxml does not require any other dependency other than ANSI C API. During its configuration libxml uses libz and iconv libraries. These libraries are readily available with Linux distribution and should not cause any problems to any user. Next one can write code using these libraries to parse an xml page and select what is needed. There are many examples of such codes available on the internet.

The advantages that the author feels with such kind of data scraping are:

1. The native languages provide more flexibility and power and the user is in total control.
2. The low level work ensures speed and reliability.

However there are several disadvantages:

1. Very long sections of code are to be written.
2. The learning process is comparatively tougher.
3. One has to take care of numerous conditions (like downloading, converting and parsing separately)
4. Compiled languages are usually not considered suitable for web development activities.
5. It is a static language and has its disadvantages.
6. One may have to modify the program quiet frequently.
7. Every time a program is modified it has to be compiled.
8. One may have to deal with memory management and string handling.

Using Object Oriented language Java

There are many ways to do web scraping with Java. The author discusses two of the most popular and important libraries used in this respect. One is the Jsoup and the other is HtmlUnit. Jsoup is a Java library used to parse HTML. It uses CSS, jquery and DOM to extract data. It has APIs to facilitate this process. The normal steps included in the activity are scraping and parsing HTML document from a file or string or URL. The second would be extracting the required data with CSS or other selectors. One can also modify HTML components. If necessary one can clean up the HTML document and then produce a tidy output. If the structure of the HTML document is known, then we can use DOM like methods to parse the HTML document. We have many methods such as getElementById(String id), getElementByTag(String tag), getElementByClass(String className), getElementByAttribute(String key) One can also manipulate the data using suitable methods. One can also play with the elements with CSS or jquery type selectors. The queries in this way are very robust. They also have great power. Several methods that may be used to parse and manipulate data in this way are: tagname, that finds elements by tag, ns|tag, #id, .class, [attribute] etc.

HtmlUnit is actually a browser without a GUI. It provides APIs that allow a user to fill forms click links and do other things that one normally does on a browser with the help of graphical tools. This is very useful if one wants to imitate a particular browser. There is a WebClient constructor. One has to pass a browser version into this constructor and there are constants available for some common browsers. The user can find a particular element by using some get methods. Or one can also use the Xpath to query elements. One can easily submit a form by following the steps: go to the required page, get the form and find the submit button and the field that is to be changed or submitted, modify the field and submit the form by clicking the submit button. One can also specify a proxy server if one needs to connect through one. An extensive API documentation is available that will allow users to use such APIs in the programming work.

Another possibility that should be listed here is the use of WebDriver like Selenium. Although it is normally a tool that may be used in automated web application testing, we can also use its APIs to extract the required information from a website. A browser like Firefox can be automated and used effectively. Although the author did not explore the possibility of this kind, this can be a useful area of exploration in the search of data mining tools.

The advantages of using Java are:

1. It is a popular language with many powerful libraries freely available.
2. Eclipse and Intellij are powerful IDEs that help analyze and organize code.
3. The success of Java is tested when it comes to big projects.
4. Concurrency is something that Java is believed to be better in.
5. A specific web browser can be simulated.
6. Complicated web pages having diverse and dynamic contents can be easily scraped.
7. Tools such as WebDriver can even take screenshots of the web page.

The disadvantages of using Java for the present work are;

1. Code in Java is too verbose and difficult to write.
2. It takes more time to write code in Java.

3. Java is a statically typed language.

4. It is necessary to compile every time any modification is made.

5. The overhead of compilation may make things unnecessarily more costly.

6. The powerful IDEs without which working in Java becomes nearly impossible take up lots of machine resources.

7. A compiled language is not advisable to do web development tasks.

8. Often it becomes necessary to restart a web browser and that consumes resources and also might cause a security issue.

9. Often the scraping process is slower as one has to wait for the entire web page to be loaded and only then the elements can be accessed.

10. Since we are often simulating web browsers there is also the disadvantage of many unnecessarily files being loaded generating heavy traffic thus causing problems.

11. The web scraping is not secret. It can be detected. Although this may give rise to ethical questions, this may be listed as a disadvantage

Using the scripting language PHP

A very easy and intuitive method for parsing and scraping data from repositories is using the scripting language PHP. For this, one needs the PHP Simple HTML DOM Parser. This is an HTML DOM parser written in PHP. This can easily extract required data from an HTML page. One can select tags from an HTML page just like jquery. Contents can be extracted from the HTML page in a single line. The latest version of Simple HTML DOM Parser is available for free on Sourceforge. The online documentation is very useful. The steps involved to procure HTML elements are: creating COM from a file or URL using the file_get_html method which requires an argument. The argument can be the URL or the file name from which data is to be extracted. Then we can use many other functions such as find('img') for extracting images and find('a') for extracting links. We can modify many elements by creating a DOM first and then finding a specific 'div' and then putting the required text in the proper place. The function file_get_html(<URL>) will just dump all the contents of the web page as a plain text. Data can be scraped easily by running a loop and selecting each 'div' element and converting its contents into plaintext, storing them as a string and showing them when required. There are numerous useful things that can be performed: creating HTML DOM objects, finding HTML elements using even descendant selectors, nested selectors, attribute filters, and also using texts and comments. One can also access the attributes of an HTML element and also add or remove the attributes as required. One can easily traverse the DOM tree and convert the contents of the DOM object into plain text and dump the same as required. If necessary the parsing styles may also be customized to suit the needs of the users.

A library is required for doing all this stuff. It is actually a simple PHP file called simple_html_dom.php and it is freely available at Sourceforge. This library has to be included in the PHP script to be used for scraping data with the normal code: require('simple_html_dom.php');

Advantages of using PHP:

1. It is freely available

2. It is a scripting language.

3. It requires only a single file as the required library.

4. It is very easy to code.

5. It is accepted by almost all enterprises and also supported.

6. Easy to learn and use.

Disadvantages of using PHP:

1. Fewness of libraries can be a disadvantage.

2. There is no scope of memory management.

3. PHP has no support for parallel works.

4. PHP has no support for Unicode (although this is not a problem in the current situation).

5. Additional PHP libraries may be written but they will make things slower.

6. Not easy to debug code.

7. Implementing security in PHP is difficult and long names of functions are to be used.

8. Maintenance is difficult.

9. It is a weakly typed language.

10. The entire script is re initialized and run from the very beginning for every web request that is made.

Using the dynamic language Python

There are many powerful Python libraries available for scraping web sites. Some of them are *Requests, Beautiful Soup, lxml, Selenium* and *Scrapy*. To extract raw HTML for any activity, *Requests* can be used which is very simple and besides extracting raw data, it can also post to forms and access APIs. If one needs to parse the extracted data and select what is required, *Beautiful Soup* can be a good choice. Python's standard library provides the default parser for *Beautiful Soup*. This is equally simple to handle. Next comes*lxml* which is very useful for fast web scraping jobs. It works on Xpath and CSS selectors. This is also an HTML parser like *Beautiful Soup* but users often prefer *lxml* for speed and *Beautiful Soup* if there is a need to handle untidy documents. Although nowadays both can be integrated into each other, so *lxml*is generally preferred. Next comes*Selenium* which is very useful for handling JavaScript. *Selenium* is actually a web driver that actually simulates web browser and can enable one click on links and enter data into forms. However, if one requires to do all of the above and build a complete spider that can crawl through websites in a very systematic way then the best thing to use may be *Scrapy*.

Hence among the many things that are available with Python the author has chosen to discuss only one, namely Scrapy which is an entire framework for scraping websites in a systematic way. According to the official website of Scrapyit is written in Python, and it depends on some Python packages: l*xml*, which is a good HTML and XML parser, *parsel*, which is written on top of *lxml* and which facilitates data extraction. Then there is a library which helps in dealing with web page encoding and URL which is called *w3lib*. A library called *twisted* which is a asynchronous networking framework. Finally to deal with network security, there are libraries called *cryptography* and *pyOpenSSL*

*Scrapy*is particularly useful when it comes to advanced tasks like extracting structured data. According to its official website this structured data "… can be used in a wide range of useful applications like data mining, information processing or historical archival. Even though Scrapy was originally designed for web scrapping, it can be used to extract data using APIs . . . or as a general purpose web crawler." Installing Scrapy in a dedicated virtual environment is usually recommended. One can write spiders using Scrapy. Spiders are Python classes that set the definitions of the web scraping techniques. In these classes the ways of parsing and crawling web pages are defined. The work of the spiders generally go through a four step process: first, initial requests are generated to crawl URLs and a callback function is specified to handle the response from these requests. Next the downloaded response is parsed and data extracted. Thirdly page contents are parsed using selectors and items are generated using the parsed data. Finally items that the spider class returns shall be stored in a proper data base or a data warehouse as per the requirement.

The advantages of Python:

1. Object oriented facilities are available but they are not compulsory to use.
2. Exceptions are used but not strictly enforced.
3. Language easy to learn and very intuitive (English like).
4. Indention enforces good programming practices.
5. Very powerful set of libraries.
6. Dynamically typed language.
7. Very few lines of code are required for tackling a given problem.
8. Users can spend more time on algorithms and problem solving rather than getting delayed with programming language features.
9. Has powerful build in data types such as lists, sequences, functions, sets and dictionaries.
10. It works on almost all platforms.
11. Run time check of array bounds possible.
12. Very fast development time.
13. One is able to compile and run the program until an error comes up.
14. Native languages such as C can be integrated into Python.
15. Python has a large community that is ready to help whenever required.
16. Contains wonderful libraries for data mining and analysis of data.

The main disadvantages are:

1. Being less verbose, it is often difficult to understand by reading the code.
2. Proper documentation absolutely essential for maintenance.
3. Normally considered not useful for big projects. (However Google and YouTube are using it).
4. Being dynamically typed, the variables and other things may become difficult to keep track of.

5. Python's flexibility makes it slow as the machine requires a lot of referencing. This can be somewhat solved if C is integrated into it. However it has to be determined whether the issue of speed is really any issue for any particular project.

Conclusions

The purpose of data scraping can be solved by using many programming languages. There is native code such as C and C++. There is the object oriented programming language namely Java. There is also the server side scripting language PHP. And finally there is the dynamically typed language Python. This list is not exhaustive. There are other options available too. There are very good options like Ruby, Perl, R, etc. that can be used. These may require further research and review. For the time being the author has discussed the pros and cons of the four above mentioned programming languages with respect to the task of extracting data from software repositories and their subsequent analysis for the purpose of trying to understand the trends in the growth of such collaborative software projects and also for trying to relate the growth trends with the improvement in the quality of the software projects. In this quest the author feels that the best language that can be used for this purpose is Python. In Python it would be in all probability most appropriate to use Scrapy which seems to be an excellent framework for data mining and also its proper storage and use. There may be a possibility that simulations of browsers may be required. In that case the use of Selenium would perhaps be most appropriate. Writing proper spiders would enable the author to extract needed data from the software repositories and if necessary use of Selenium would be appropriate if logging into websites is required for some purpose.

Future Scope

This paper discusses the options for extracting data from websites by writing code in different programming languages. While writing code would provide greater strength and flexibility, it also has the over head of typing and correcting syntactical errors and other things. Often things have to be learnt the hard way. There are several instances of people backing away for the sheer fear of programming. Hence the issue of mining software repositories when addressed cannot simply remained confined to coded tools. There are also tools that are available via the GUI and they do not require a single line of code to be written. Such tools may not be so flexible and powerful. They may actually use up a lot of system resources. However these can be readily used for small things (and also for big projects). A comparison of such tools and related review can be very useful for researchers who are trying to mine software repositories. Besides another review may be conducted on how the data that has been extracted from software repositories can be stored. There may be many methods and their comparative study would enable researchers select a tool that could suit their requirements. Another aspect in the future scope is the review of analytical tools for understanding the data that has been gathered from the software repositories. This though largely depends on what kind of analysis one is typically looking at, we need the comparison all the same. A statistical insight about how much code is used to extract data from a particular repository and how much time is taken to collect that data can also be presented in a paper. This will provide an objective understanding about the different programming languages mentioned here.

References

[1] http://www.llrx.com/2016/02/web-data-extractors-2016/
[2] Grasso G. et al., Effective Web Scraping with OXPath, Department of Computer Science, Oxford University.
[3] Fernandez O. C., Web Scraping: Applications and Tools, European Public Sector Information Platform, Topic Report No. 2015/10, December 2015, pg. 6-7
[4] Kim S. et al., TA-RE: an exchange language for mining software repositories, Proceedings of the 2006 international workshop on Mining software repositories, pg.22-25
[5] Dyer R. et al., Boa:Ultra Large Scale Software Repository and Source Code Mining, Computer Science and Technical Reports, (2015), paper 374
[6] https://jsoup.org/
[7] https://jsoup.org/cookbook/extracting-data/dom-navigation
[8] https://jsoup.org/cookbook/extracting-data/selector-syntax
[9] http://htmlunit.sourceforge.net/gettingStarted.html
[10] https://doc.scrapy.org/en/latest/intro/install.html#intro-install
[11] https://doc.scrapy.org/en/latest/intro/overview.html