A Web Crawler Design for Plant Data Collection

201620700 Xiang, Qin
Advisor Li, Jie

November 17th, 2016

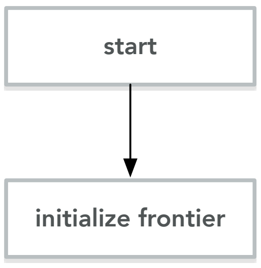


Figure 1: Basic Flow of Web Crawler

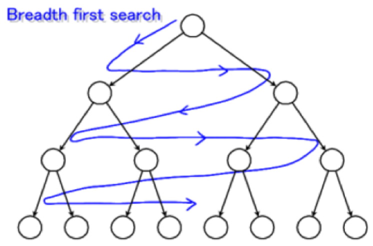


Figure 2: Breadth-first Search

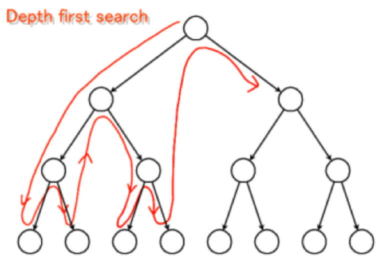


Figure 3: Depth-first Search

1 Introduction

A web crawler[1] (also known as a robot or a spider) is a system for the bulk downloading of web pages. Web crawlers are used for a variety of purposes. Most prominently, they are one of the main components of web search engines, systems that assemble a corpus of web pages, index them, and allow users to issue queries against the index and find the web pages that match the queries. A related use is web archiving, where large sets of web pages are periodically collected and archived for posterity. A third use is web data mining, where web pages are analyzed for statistical properties, or where data analytics is performed on them. Finally, web monitoring services allow their clients to submit standing queries, or triggers, and they continuously crawl the web and notify clients of pages that match those queries.

The basic flow of web crawler is shown in Figure 1. This is a sequential example. The starting pages are called seed. Web crawler starts to crawl from seed. Then in the crawl process, we may have a list, sometimes called frontier, to crawl. Every time we extract one URL from the list, access the URL to get the information in it. After that, we may add URLs into the frontier. To analyze the data, we need to store the information we crawled from the pages into the repository, like database. At last we may need a check to determine when to stop crawling.

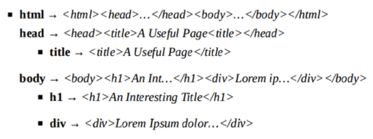
There are two normal search strategies. One is Breadth-first search(Figure 2), the other is Depth-first search(Figure 3). Breadth-first search can be implemented by queue, which uses the FIFO(First In, First Out). And Depth-first search can be implemented by the stack(LIFO) or recursion.

Another existed method is originally used to looking for external links of one page. I intend to use it not only for searching for external links, but also searching for the data about plant. And then store the data in database, clean the data. At last, I intend to make an analysis about the collected data. So, I make changes of the existed method, let it grab the data from pages, not just links. When crawling job of one page is finished, it will ask if we need to continue crawling. What's more, I add some checks on the links, so that the program will not crash.

2 Related Research

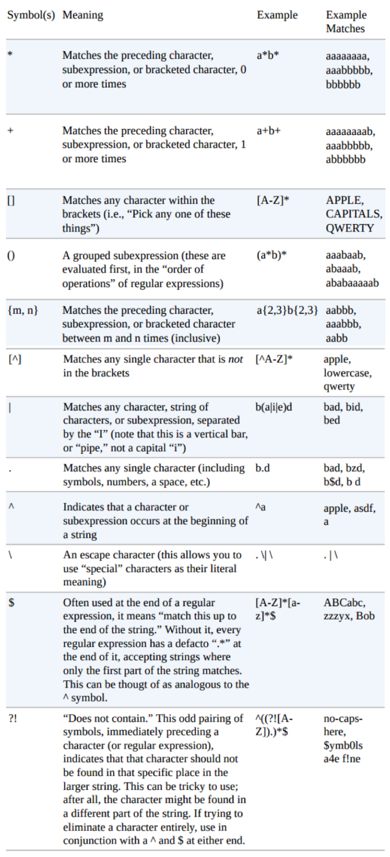
2.1 BeautifulSoup

The BeautifulSoup is a function library, which tries to make sense of the nonsensical; it helps format and organize the messy web by fixing bad HTML and presenting us with easily-traversible Python objects representing XML structures. For instance, the BeautifulSoup can transform a html page into a BeautifulSoup object, with the following structure(Figure 4):



- **html** → *<html><head>...</head><body>...</body></html>*
  - **head** → *<head><title>A Useful Page<title></head>*
    - **title** → *<title>A Useful Page</title>*
  - **body** → *<body><h1>An Int...</h1><div>Lorem ip...</div></body>*
    - **h1** → *<h1>An Interesting Title</h1>*
    - **div** → *<div>Lorem Ipsum dolor...</div>*

Figure 4: Parsed by BeautifulSoup



Symbol(s)	Meaning	Example	Example Matches
*	Matches the preceding character, subexpression, or bracketed character, 0 or more times	a*b*	aaaaaaaa, aaabbbbb, bbbbbb
+	Matches the preceding character, subexpression, or bracketed character, 1 or more times	a+b+	aaaaaaaab, aaabbbbb, abbbbbb
[]	Matches any character within the brackets (i.e., "Pick any one of these things")	[A-Z]*	APPLE, CAPITALS, QWERTY
()	A grouped subexpression (these are evaluated first, in the "order of operations" of regular expressions)	(a*b)*	aaabaab, abaaab, ababaaaaab
{m, n}	Matches the preceding character, subexpression, or bracketed character between m and n times (inclusive)	a{2,3}b{2,3}	aabbb, aaabbb, aabb
[^]	Matches any single character that is *not* in the brackets	[^A-Z]*	apple, lowercase, qwerty
|	Matches any character, string of characters, or subexpression, separated by the "I" (note that this is a vertical bar, or "pipe," not a capital "i")	b(a|i|e)d	bad, bid, bed
.	Matches any single character (including symbols, numbers, a space, etc.)	b.d	bad, bzd, b$d, b d
^	Indicates that a character or subexpression occurs at the beginning of a string	^a	apple, asdf, a
\	An escape character (this allows you to use "special" characters as their literal meaning)	.\\|\	.|\
$	Often used at the end of a regular expression, it means "match this up to the end of the string." Without it, every regular expression has a defacto ".*" at the end of it, accepting strings where only the first part of the string matches. This can be thougt of as analogous to the ^ symbol.	[A-Z]*[a-z]*$	ABCabc, zzzyx, Bob
?!	"Does not contain." This odd pairing of symbols, immediately preceding a character (or regular expression), indicates that that character should not be found in that specific place in the larger string. This can be tricky to use; after all, the character might be found in a different part of the string. If trying to eliminate a character entirely, use in conjunction with a ^ and $ at either end.	^((?![A-Z]).)*$	no-caps-here, $ymb0ls a4e f!ne

Figure 5: Common Regular Expressions

2.2 Regular expressions in Python

Aregular expression, regex or regexp is a sequence of characters that define a search pattern. Usually this pattern is then used bystring searching algorithmsfor "find" or "find and replace" operations onstrings. The Figure 5 lists some commonly used regular expression symbols, with a brief explanation and example. These 12 symbols are the most commonly used regular expressions in Python, and can be used to find and collect most any string type. A regular expression can be inserted as any argument in a BeautifulSoup expression, allowing you a flexibility in finding target elements.

3 System Description

3.1 Purpose

There are two purposes to run this experiment:

- Make a web crawler to crawl the information

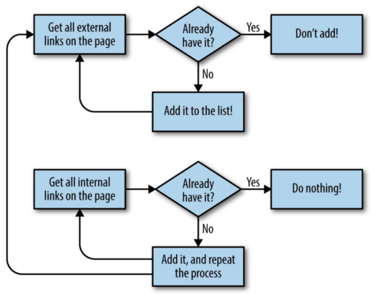
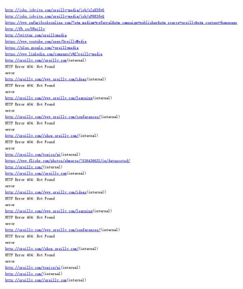

Get all external links on the page
Already have it?
Yes
Don't add!
No
Add it to the list!
Get all internal links on the page
Already have it?
Yes
Do nothing!
No
Add it, and repeat the process

Figure 6: The Stratege of Crawl



```
http://jobs.jobvite.com/oreilly-media/job/oZqU3fw6
http://jobs.jobvite.com/oreilly-media/job/oP683fwQ
https://www.safaribooksonline.com/?utm_medium=referral&utm_campaign=publisher&utm_source=oreilly&utm_content=homepage
http://fb.co/OReilly
http://twitter.com/oreillymedia
https://www.youtube.com/user/OreillyMedia
https://plus.google.com/+oreillymedia
https://www.linkedin.com/company/o%27reilly-media
http://oreilly.com//oreilly.com(internal)
HTTP Error 404: Not Found
error
http://oreilly.com//www.oreilly.com/ideas(internal)
HTTP Error 404: Not Found
error
http://oreilly.com//www.oreilly.com/learning(internal)
HTTP Error 404: Not Found
error
http://oreilly.com//www.oreilly.com/conferences/(internal)
HTTP Error 404: Not Found
error
http://oreilly.com//shop.oreilly.com/(internal)
HTTP Error 404: Not Found
error
http://oreilly.com/topics/ai(internal)
https://www.flickr.com/photos/ebmorse/7536436622/in/dateposted/
http://oreilly.com/(internal)
http://oreilly.com//oreilly.com(internal)
HTTP Error 404: Not Found
error
http://oreilly.com//www.oreilly.com/ideas(internal)
HTTP Error 404: Not Found
error
http://oreilly.com//www.oreilly.com/learning(internal)
HTTP Error 404: Not Found
error
http://oreilly.com//www.oreilly.com/conferences/(internal)
HTTP Error 404: Not Found
error
http://oreilly.com//shop.oreilly.com/(internal)
HTTP Error 404: Not Found
error
http://oreilly.com/topics/ai(internal)
http://oreilly.com/(internal)
http://oreilly.com//oreilly.com(internal)
```

Figure 7: Crawling

about the plant. And then store the data in the local database.

- Analyze the local data and make a prediction from the data, like climate.

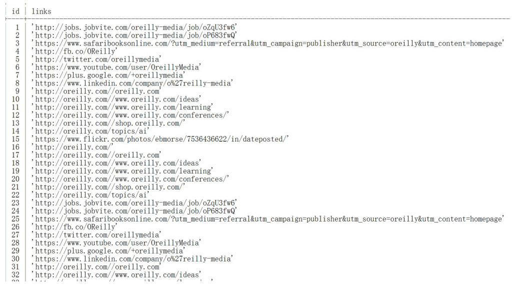
3.2 Proposed Model

Suppose that we intend to crawl the Internet from one starting page. We hope to grab all the information about this page, including its internal links and external links. After all the contents from this page, if we need to crawl more pages, we can start from one of its external links and continue this process. The basic flow is shown in Figure 6:

There are two loops in this method. One is to get the internal links. The other is to get the external links. I let it get the contents of the pages while it access a new page. And make a list to record the footprints. So that it will not access the same page again. And if we intend to continue crawling after all the internal links and external links are found, then a new starting page will begin from one of its external links. Furthermore, I add error checks on links and set the crawling interval to 1 second.

3.3 Results

Results are shown in Figure 7 and Figure 8. To show the results, here only the links are collected, as there are still some challenges to be done. Like parse the contents in the pages. Also it is preferred to store more effictive data in database, not just copy all the contents of them.



```
 id | links
----+--------------------------------------------------------------------------------------------------------------
  1 | 'http://jobs.jobvite.com/oreilly-media/job/oZqU3fw6'
  2 | 'http://jobs.jobvite.com/oreilly-media/job/oP683fwQ'
  3 | 'https://www.safaribooksonline.com/?utm_medium=referral&utm_campaign=publisher&utm_source=oreilly&utm_content=homepage'
  4 | 'http://fb.co/OReilly'
  5 | 'http://twitter.com/oreillymedia'
  6 | 'https://www.youtube.com/user/OreillyMedia'
  7 | 'https://plus.google.com/+oreillymedia'
  8 | 'https://www.linkedin.com/company/o%27reilly-media'
  9 | 'http://oreilly.com//oreilly.com'
 10 | 'http://oreilly.com//www.oreilly.com/ideas'
 11 | 'http://oreilly.com//www.oreilly.com/learning'
 12 | 'http://oreilly.com//www.oreilly.com/conferences/'
 13 | 'http://oreilly.com//shop.oreilly.com/'
 14 | 'http://oreilly.com/topics/ai'
 15 | 'https://www.flickr.com/photos/ebmorse/7536436622/in/dateposted/'
 16 | 'http://oreilly.com/'
 17 | 'http://oreilly.com//oreilly.com'
 18 | 'http://oreilly.com//www.oreilly.com/ideas'
 19 | 'http://oreilly.com//www.oreilly.com/learning'
 20 | 'http://oreilly.com//www.oreilly.com/conferences/'
 21 | 'http://oreilly.com//shop.oreilly.com/'
 22 | 'http://oreilly.com/topics/ai'
 23 | 'http://jobs.jobvite.com/oreilly-media/job/oZqU3fw6'
 24 | 'http://jobs.jobvite.com/oreilly-media/job/oP683fwQ'
 25 | 'https://www.safaribooksonline.com/?utm_medium=referral&utm_campaign=publisher&utm_source=oreilly&utm_content=homepage'
 26 | 'http://fb.co/OReilly'
 27 | 'http://twitter.com/oreillymedia'
 28 | 'https://www.youtube.com/user/OreillyMedia'
 29 | 'https://plus.google.com/+oreillymedia'
 30 | 'https://www.linkedin.com/company/o%27reilly-media'
 31 | 'http://oreilly.com//oreilly.com'
 32 | 'http://oreilly.com//www.oreilly.com/ideas'
```

Figure 8: Storing in database

Column	Data Type	Introduction
sid	String	Physical unique identifier
content	String	JSON String of object
object	String	Name of object
count	Integer16	Count of sync time
resend	Boolean	Necessity of resending data
sendtime	Date	Data send time

Table 1: Columns of sender table

4 Evaluation

There are some merits about this method:

- It can reduce redundancy with a list to record the already accessed pages.

- To avoid unexpected error, there are checks to make sure it will not crush. Such as HTTP errors, server is not found and attributes of tag is not existed.

- To reduce the overhead to the network, the crawl is run at regular intervals.

5 Future Work

Eventhough we can get the links and print the pages, there are still some challenges are needed to be solved:

- How to parse the web pages we get. For example, deal with the increasing file format, like: PDF, Flash, SVG, RSS.

- Given finite storage capacity, in practice crawlers discard or retire low-quality and spam pages from their collections, to make room for superior pages.

- How to deal with dynamic pages.

- How to avoid violating the law.

- The way of storage in database.

- Python set a recursion limit to 1000 times. What to do if it reach the maximun recursion depth.

- How to grab picture an video.

- How to deal with the repeating segments appeared in URLs, like:$/blog/blog/blog/blog.../blog/title_of_blog.php$

- Clean the data once in the database.

What's more, the crawling method may be improved in several ways. After finishing the tasks, how to analyze the data is still a problem to be done.

References

[1] Christopher Olston and Marc Najork. Web crawling. *Foundations and Trends in Information Retrieval*, 4(3):175–246, 2010.