Methods of automation for system for collecting data from Internet construction

Ageykin M.A, Andrianov A.V., Chugunov V.R. Lychagin K.A., Novopashin M.A.

JSC EC-leasing, Moscow, Varshavskoe shosse 125,
mageykin@ec-leasing.ru

Abstract. Article describe a set of methods for construction automation of specialized internet crawlers to collect data ether from static internet pages or dynamic pages. Described methods well combined with micro-service architecture in big projects.

Keywords: Internet, html, scrapy, xhtml, Ajax, crawling

According to IBM strategic forecast, all companies in the next 5 years will be divided into winners and losers depending on quality of making corporate decisions. Research and case studies provide evidence that a well-designed and appropriate computerized decision support system can encourage fact-based decisions, improve decision quality, and improve the efficiency and effectiveness of decision processes. There is resource that we all have aplenty: a large amount of open data, both structured and unstructured. This report introduces the concept of acquiring data from big data sources such as social media, news, mobile and smart devices, weather information, and information that is collected via sensors and using this data to get new quality of predictive analytics. Predictive analytics today in many companies is perceived as an evolutionary step in business analytics and is used primarily to build a forecast based on the same data on which reports are built. Nevertheless, this does not take into account the enormous importance of external factors in forecasting and nowcasting. In this report will be shown cases from various areas where external data are the basis for predictive analytics and allow gain results unattainable to the forecast based only on enterprise data.

Customers are increasingly asking what data can be used to improve the quality of analytics and forecasting. Strange as it may seem, there is a lot of such data, but even data that can be accessed legally from a legal point of view rarely have APIs to receive them, so you often have to collect data from various sites which are generally quite different from each other. Authors would like to share experience in the construction of a typical project parsing. If there are strict requirements to the quality of the data received, the data is collected using specialized crawlers tuned to specific sources. For each data source, a separate crawler is created (the class for determining a particular site or group of sites). In the crawler the following handlers are defined:

- for a specific page in order to isolate the semantic information, separating it from the design
- a handler for scanning links and sending queries to the queue for uploading and further processing pages
- the processor of the archive or site map (if available) to bypass the time interval of the records and to queue page requests. Scheduled crawl pages with content retention

Specialized crawlers usually differ depending on the type of sites with which they should work. First of all: they are static or dynamic.

In the case of static pages for crawling, retrieving links and content parsing, we generally use such tools for Python as a scrapy framework or robobrowser.

Scrapy is a web crawling framework, which has done all the heavy lifting that is needed to write a crawler. This is the most effective way to create crawlers and probably no article dedicated to crawling today can not do without mentioning this framework.

RoboBrowser combines the best of two excellent Python libraries: Requests and BeautifulSoup. RoboBrowser represents browser sessions using Requests and HTML responses using BeautifulSoup, transparently exposing methods of both libraries. The use of RoboBrowser is usually convenient in case we want the crawler behaved as much as possible to the user and did not cause suspicion as a robot.

Regardless of the technology chosen, you can create a spider that performs GET requests, extract data from an HTML document, process and export data. To work with html, you will need to study the structure of the site and describe the actions of spiders what information to bypass and what information to collect using Xpath or CSS selector. The universal answer is that it is better to use Xpath or CSS selector, and this is primarily a matter of personal preferences. Personally, authors recommend using Xpath, but in a number of projects, it can be more convenient to use the CSS selector, so one need to know these two methods of addressing the html elements.

However, in the modern world, working with static sites does not allow you to access all the required information. Many of the modern sites have so-called dynamic pages, in which updating of information occurs without reloading the page by executing a JavaScript-code substituting the displayed part. In this case, the loaded page itself will most likely not contain any content. The content that is supposed to be displayed will be obtained after the page is loaded through Ajax-requests, and not necessarily in the html format. Most likely, the data will be received in json format, and then displayed on the page in accordance with the markup.

There are two approaches to retrieving information from such sites:

A simple but resource-intensive method is preliminary javascript rendering with subsequent analysis and retrieval of information from the html page.

In this case, you need to use a browser that handles javascript, in principle, any browser is suitable: Internet explorer, Mozilla Firefox, Google Chrome, PhantomJS, etc. On the other hand, you can create your own using node.js. Calling these browsers and getting the contents of the Internet pages from them through Selenium and the drivers for these browsers.

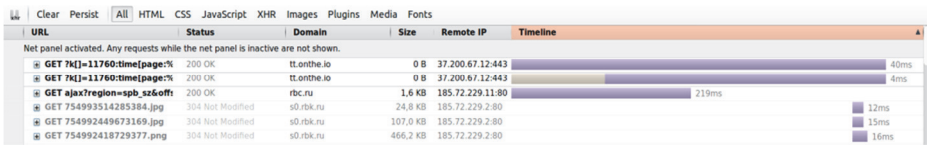
PhantomJS was originally designed to render JavaScript without displaying the content to the user, so for the data collection task it works faster than the others work and does not run additional windows regardless of the operating system. To the main disadvantages, one would refer to the greater laboriousness of debugging, because of the lack of a visual interface and possible problems with rendering JavaScript, since many developers do not support standards, there's aim to make site work in popular browsers.

Google Chrome now seems the most promising and convenient. It uses the most common JavaScript engine WebKit. Websites is primarily optimized for Chrome, either it is convenient for debugging due to the presence of a visual display for any actions. As well recently, a headless mode have been added. Now it works only on Linux and MacOS, and the version for MS Windows with support for headless mode should be released in October this year.

The second approach requires additional knowledge about the site code responsible for filling the page with information.

Consider the example of the site www.rbc.ru. To analyze the information exchange between the page in the browser and the server, one can use Firefox with the Firebug plugin or any other similar products are also suitable. This plugin starts with the hot key F12. We will be interested in the "Network" section, which contains all the additional requests from the page.

Consider one of the news sections, http://www.rbc.ru/spb_sz/. This page contains a table of contents containing links to the pages with the text of the news. Scrolling the page down to the end, you can notice that at a certain moment, additional news is downloaded. The network bar shows us the following:



The first two requests go to a domain that is not visually associated with the site, the last three requests are the download of pictures. We are interested in the GET ajax request from rbc.ru.

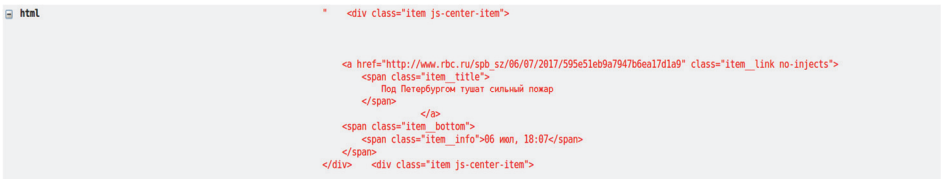
Let's consider it more attentively:

```
Http://www.rbc.ru/filter/ajax?region=spb_sz&offset=10&lim
it=12
```

We see three parameters transmitted by the GET method. In this case, the POST method does not transmit anything, but if you need to fill out some form, refine the data with filters, then this method can be used.

In this case, all three parameters are obvious to us: the region, the offset relative to the latest news and the amount of news in the request.

In this example, a html-code with an understandable structure is returned as a response from which you can select links to news pages:



Further, this html page processed as well as any static web page. In this article, we examined the basic options for creating specialized crawlers that are often used to obtain open data and competitive intelligence, for example, monitoring competitors' prices, monitoring the dynamics of their changes, etc. Such crawlers store information in a structured form, so information from them is conveniently stored in relational databases, and the results they obtained do not require cognitive technologies for further processing.

References

1. Anil Maheshwari “Data Analytics Made Accessible”
2. Davy Cielen, Arno Meysman, Mohamed Ali “Introducing Data Science: Big Data, Machine Learning, and more, using Python tools”
3. Eugene Rabchevsky, "Search, monitoring and analysis in social networks". URL: https://www.osp.ru/os/2015/04/13047968/
4. Kenneth Cukier “Big Data: A Revolution that Will Transform How We Live, Work, and Think”.