Review of Using Scrapy to Crawl Online Text Information

Reference Link: https://scrapy.org/
By Jianci Zhai (Angela) - jianciz2

Introduction:
This review is to provide a quick summary of my experience using Scrapy, a very popular package on python to crawl text messages. This review will fit into the scheme of “Useful software toolkits for processing text data or building text data applications”, and will also cover a short comparison between Scrapy and BeautifulSoup, another popular package to crawl search results.

My background is not from computer science, so I’ve learned my coding skills mostly from taking online courses. As far as I remember, to learn how to use BeautifulSoup, I have only spent 5 minutes finishing a small assignment on Coursera. Simply providing an URL, and using some basic loops combined with CSS notations to parse the text file - done. However, to learn how to use Scrapy, I have to take an entire course on Datacamp (4 hours total) to just feel a little more comfortable using Scrapy FOLLOWING youtube videos. So in terms of difficulty, Scrapy is much more challenging for coding newbies like me, because it requires more understanding in the customized functions/classes developed just for Scrapy package- a.k.a. Learning how to speak like the coders who created Scrapy is the first step, not like BeautifulSoup is simply parsing text strings.

For example, to start the project you have to first launch a “spider” class (a class that defines the behavior of the crawler) with the Scrapy.Spider object as the input. In this class, we will first initiate the program with one or a set of urls to create an “invisible” response object that will be passed over to one or more parser steps. The Scrapy project can be very handy working within command line environments. We can also define the folder structure for a project, and organize the program output by the key words we use to crawl.

BeautifulSoup will crawl the complete string set from the website, so it feels more secure when we have the search results first saved on the computer, then we can take a break and have a cup of tea then return to continue working on it. For Scrapy, if the project is not set properly (a lot of time the work is in the command line environment), it won’t produce anything. Even when it works, the output / message is a little harder to understand - it takes me a couple hours to understand the output messages telling me my crawler was blocked by the search engines - the developers definitely should think more about how to improve the message to make it more user friendly.

Compared to BeautifulSoup where we download everything and then parse, the approach of Scrapy taking is to parse the page first following xpath or css scripts, so it's harder to start because we have to define all the html attributes/class appropriately. However, once it starts obviously it will be more efficient then BeautifulSoup because only relevant information will be downloaded onto the computer, so there is huge savings in terms of I/O.

When first start my project, I was thinking about using BeautifulSoup to retrieve the search results, then later I realize it would be pretty hard to do so, because I have to manually select and input the urls of search results to BeautifulSoup - it will be very tediously say if we want to collect a couple thousands of search results. By comparison, Scrapy provides the functionality to extract the links from the url initially received, and crawl them one by one. What's even better is that we can launch several spiders (each is a singular crawler created by Scrapy performing its own task) - which indicates a sort of parallel programming, as compared to BeautifulSoup the design is more linear (of course you can do some parallel programming tricks as well - but there will take some time and use other libraries). Its parse-first-crawl-later approach, combined with extracting links from pages with an army of spider-bots crawling the internet at the same time, means Scrapy will be much more efficient than BeautifulSoup and much easier to scale up. No wonder even though it's harder to learn, eventually it still win out to be the most popular choice of web crawler for seasoned programmers.

One thing I haven't had a chance to test deeply on the BeautifulSoup side is that Scrapy supports a wider variety of HTML parsers, and it's more flexible to create neatly formatted csv files, so it's easier to create the text scraping pipelines. BeautifulSoup seems to offer some as well, but just not as many as Scrapy offers. For example, when I want to crawl for search results, there is a function from Scrapy that is readily built for this task: LinkExtractor.

Conclusion:
Winner:

- Newbie friendly - BeautifulSoup
- All other aspects - Scrapy

Well, after saying so much, honestly in the end I have to admit, the real hard piece for finalizing a tiny school project on online crawling is really not choosing between BeautifulSoup or Scrapy - the hard part is to work around the firewall built by Google or whatever. Whichever first comes with its own API that can smartly avoid being blocked by search engines - it will be the winner.