Marcus Rosti

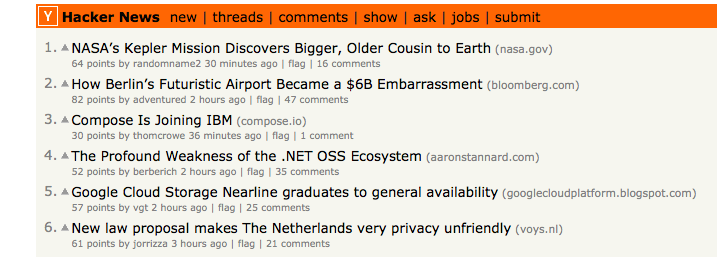
mer3ef

CS 5010

23rd July 2015

Python: Web Scrapper

## Problem description

I love YCombinator's [Hacker News](https://news.ycombinator.com/). It provides a great platform for discussing tech news and has a minimalist interface for fewer distractions. As such, the websites lends itself to scrapping and analysis. The page lists titles with hyperlinks to a website or a discussion forum on the site with users voting on titles to vet the more interesting ones.

It could be a good barometer for tech opinions and trends but we will need to collect the data from it first.

The webpage is written entirely in html. In the above figure, it even looks as if the entire form lies in one large html table. When I looked at the source code, I was right! The title and link lie in a row and the next row after that is contained in a “class=subtext” row. One downside of their setup is that the title row has “class=title” however, so do 30 other elements. The file looks ripe for scrapping

The biggest hurdle in scrapping websites lies in picking the right package. [Beautiful Soup](http://www.crummy.com/software/BeautifulSoup/), a python library, downloads a webpages html and leaves it in a “soup” of data. It automatically parses the data by its html and css tags and allows python to access that data programmatically. Beautiful Soup stores the html in a tree that can be accessed by repeated accessing them as if they were public fields as well as accessing them various find and select functions.

I began parsing the data by looking for unique tags and then extracting the contents of each one. I wrote functions to harvest the raw data and create genuine entries for my future table. Each function takes in a line number corresponding to the relevant line in the table. After finishing parsing, I print out the data and pipe it into the csv.

In spyder, I explored the different analytical aspects of this project. Given the small dataset and rapidly changing information, I did see a connection between title length and points. However, the even stronger connection was between points and comments. Posts that are controversial receive higher attention since users are not allowed to vote down until a certain point threshold for the account. My conclusion is that it increases user engagement since the site forces an irate user to voice their opinion in the comments rather than with a voting system. Thus these comments get more votes overall since this person will want their comments seen. The point system creates a gamification of online interactions. I suspect that git commit would work in the same way.

My next step would be just to collect more data. I only scrapped the first page of the website and didn't follow the links to download comments. It would be interesting to consider how the content of comments contribute to the success and failure of posts. Also, my next step would be to introduce some sort of natural language processing or sentiment analysis. Perhaps the positivity or negative of comments contribute to the post.