Trevor Moyer, Alex Powell

Progress Report - intelligent browsing

(HN = HackerNews, the site we are crawling/making extension for)

Since our initial proposal, we have been working on the web crawler. When we first started making the web crawler, we were going to use javascript, but found that it would be easier to write the web crawler in python with the aid of the library beautiful soup, and then store that info from the web crawler on a back end system, with a front end handling all the requests for information. Initially we planned to do search/retrieval for articles, however, we realized that HN actually had a pretty decent search functionality already, making our initial project not too useful. So, we decided to pivot the project to do clustering on these articles and the extension will allow you to find similar articles that have been linked on HN.

For the web crawler itself, we are crawling through about a year's worth of articles from the site. Within those articles, we are only gathering specific articles that we feel are more valuable to a search process, and that tend to be longer, such as github repos and news stories. In order to create a more efficient crawler, we created a blacklist functionality to not crawl websites we feel do not add much to the search such as social media postings, or youtube links. We also implemented multiprocessing to ensure that the crawler is efficient from a time perspective as well. In terms of storage required to store all the articles we are looking over, we found that the amount of articles in a year was about 10gb, so we may have to do compression of some sort in the future. However, after implementing this and running it for an hour, we got permanently banned on 1 ip for too many requests.

After investigating, we realized that there was in fact an exposed API for HackerNews. Using this, we were able to fetch a specified number of articles between 2 UNIX timestamps. However, there were actually many articles posted each day and doing a GET request for each URL to get the HTML so we can get the text was taking a long time. Thus, we first retrieved all the articles for a timestamp interval and then split up the url's among different processes and they all did GET requests in parallel, which significantly improved the speed. The next step for this is to use Process pooling as it will lower the overhead of spawning num_threads processes for each day. After that, we will be able to move on to the actual clustering/front end.

Overall, so far we have completed the crawler for the project, and started learning some javascript for the coming parts. Our plan for the remainder of the time is to to create the front end portion of this extension that does the queries, and then link that to the web crawler directory.