David Pennington

Jun Huang

CS 585

Milestone 2

**Project Assessment**

**Shambler Team Overall Progress**

Everything is going better than expected as of a couple weeks ago. Just about every one is participating in some way to the project.

* I have written all of the code so far for the crawler, but that was the original plan in the first place so I’m not too worried about that.
* Nick has taken charge of making sure that the computers don’t die and also set up the database. Nick also helped me debug any system problems that we were having.
* Tommy has also helped out with making sure the db doesn’t go down.
* Conner wrote a script that handles the api fetchers on the multilab
* Jun and John helped out with code review.
* Abigail and Brandon Helped out with defining the content they wanted for their projects.

As always in projects, things don’t always go to plan. We had a couple problems with my code that we quickly figured out during testing, but some of the more tough things like OS specific stuff didn’t pop up until we started crawling with large numbers of crawlers.

* Os X (any bsd) does not play nice with the sockets library in python. It was written with epoll in mind. Therefore when I used the sockets library it caused all sorts of bugs that we were not expecting to work around.
* The database had to be reinstalled 5-6 times. Postgresql is a very cool open source DB, but OSX tried to index it constantly because we didn’t disable that in the install, which put a huge strain on our DB computer.
* We had no problems with the frontier. It never went down after the final code change.
* The API fetchers worked perfectly on the Multilab. Furthermore the script that Conner wrote kept them up if tumblr stopped responding to it perfectly.

Now it seems like we have the necessary amount of data to start making some cool things with it.

**Pagerank Team Progress**

Jun and I have made good progress on the PageRank implementation. We do not have a final version of the code for computation but will within the week. Jun is at PAX East this week and will be returning Tuesday.

For this project I decided to use octave, partly because I am taking the ML course from coursera where they use octave for matrix computation, and because matrices are really easy in octave.

My code will look short to you when I put it in this paper but it is the full implementation of the original PageRank and even has all of the middle steps.

One thing that we still need to do is change the matrix representation to an adjacency list, but that will take some time figure out as the book did not describe in detail how to optimize for the adjacency list version of H. This may require changing programming languages only because it changes how octave will handle the matrices. But for now our system will work for some pretty large datasets.

**Code Review**

the idea is really simple for our code, implement a pagerank algorithm on our tumblr data. For this problem we chose a combination of Python and octave. The python will be used to parse the dataset into a matrix of links. Then Octave will be used to calculate the Google Matrix and then the pagerank vector. Octave can handle matrices as single variables, and has very cool, very easy matrix operations ( ` = transpose, \* = cross product, … ) this made it very easy to write the code.

The python program is still under development. We have decided on the link structure for tumblr in that reblogs will be seen as outlinks and notes in general will be seen as inlinks. This takes advantage of our notes structure that we originally thought was going to be useless. Unfortunately we don’t have the whole dataset due to tumblr’s API killing notes after 50 but we will get a good idea of the structure with reblogs because of the nature of them showing up as posts.

Furthermore, we follow the standard algorithm and found some sample data that already had a calculated PageRank to test our algorithm on and it worked, so I would assume that it will work as required. I have not tested this at scale yet but I plan to test it on a million randomly generated pages before we test on the dataset (mostly because every team is in the process of building their data from the database).

%our initial matrix this one is a test matrix. When we run production level code, it will be built from a database via the python and loaded from a test file

H = [0,1/2,1/2,0,0,0; 0,0,0,0,0,0; 1/3,1/3,0,0,1/3,0; 0,0,0,0,1/2,1/2; 0,0,0,1/2,0,1/2; 0,0,0,1,0,0];

%our randomness value

alpha = .9;

%the number of links

n = size(H,1)

%max iterations before cutting off convergence

max\_iterations = 10;

%build our stochastic matrix from our link matrix

S = zeros(size(H));

for i=1:size(H,1),

%if there are no links on site, give the random value

%to all links

if (sum(H(i,:)) == 0)

for j=1:size(H,2),

S(i,j) = 1/n;

end

%else just put the row from H

else

for j=1:size(H,2),

S(i,j) = H(i,j);

end

endif

end

%build the vector of ones for the google matrix

e = ones(n,1);

%build our google matrix from our stochastic matrix

G = alpha\*S+(((1-alpha)/n)\*(e\*e'));

%build our pagerank vector

pagerank = (1/n)\*ones(1,n);

%run the power method on our google matrix and pagerank vector

for i=1:max\_iterations,

pagerank = pagerank\*G;

end

%print our for each page

pagerank

% print a sorted list of our pages by pagerank

pagerank\_sorted\_list = zeros(1,n);

sorting\_rank = pagerank;

for i=1:n,

max\_val = 0;

max\_index = 0;

for j=1:n,

if max\_val < sorting\_rank(1,j),

max\_val = sorting\_rank(1,j);

max\_index = j;

endif

end

sorting\_rank(1,max\_index) = 0;

pagerank\_sorted\_list(1,i) = max\_index;

end

%print out our sorted rank, the value are indexes

pagerank\_sorted\_list

This will be our algorithm for calculating pagerank, it will return the ordering as well as the pagerank for each blog.

As discussed, there may be another program that will transfer the data from the database. That will probably be written in python.

**Final Deliverables**

Within the weekend (April 14th) we will start a copy of our database values into a dataset that will be useable for octave. By next week we plan to have at least a subset of our blogs ranked based on other blogs in the subset (April 20th). We plan to create a new database that will be hosted from a computer that can handle it, and then hand that off to Tommy to use in his search algorithms before the bof (April 26th).