

## Assignment 4

### IMPLEMENTING THE PAGE RANK ALGORITHM

As part of this assignment we will be implementing Google's Page Rank Algorithm. For this assignment you will be restricted to crawling specific departments at the University. Additionally, the crawl depth is set to **5** i.e. you will follow hyper-links from a page recursively up to a maximum depth of **5**. The websites that you are expected to crawl are listed below:

1. <http://www.bmb.colostate.edu/index.cfm>
2. <http://www.biology.colostate.edu/>
3. <http://www.chm.colostate.edu/>
4. <http://www.cs.colostate.edu/cstop/index.html>
5. <http://www.math.colostate.edu/>
6. <http://www.physics.colostate.edu/>
7. <http://www.colostate.edu/Depts/Psychology/>
8. <http://www.stat.colostate.edu/>

Note that in some cases you are expected to follow redirects. For e.g. when you try to crawl <http://www.cs.colostate.edu> you will be redirected to <http://www.cs.colostate.edu/cstop/index.html>. You will not crawl hyper-links that lead to a domain different from the one listed above. The crawler needs to be able to cope with both relative and absolute URLs.

When you compute the scores for the nodes in your graph it may sometimes entail several rounds for the scores to stabilize. We will set the maximum number of rounds for a node's score to stabilize at **25**. Your implementation needs to detect and cope with special cases where two nodes may simply toggle scores in successive rounds of your scoring process. We discussed some of these situations in class.

When you crawl a page, that page becomes a node in your graph. For each node (identified by a unique URL) you will maintain a list of the words that appear in the document. We will not record words that appear in the stop list at <http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop>.

Additionally, as part of this assignment we will not be taking into account non-HTML documents (you can ignore .doc and .pdf, but not .cfm, .asp and .php documents for example). In some cases the redirects will be performed in Javascript; here, you can look for the `window.location` tag to follow the redirect. Though you will not be crawling non-HTML documents (such as .doc and .pdf) you will account for these links in your score for the page in question.

**Note:**

To cope with scenarios where there are dead-end pages (with no outgoing links) that result in the cumulative score of the graph decreasing in successive rounds, each dead-end node will distribute its score over all the other (excluding itself) nodes in the graph. This redistribution of scores will span domains.

When a search request is submitted to your interface you will check for the occurrence of the search terms in nodes with the highest score first (The number of times a word occurs in the page does not affect its order). Results need to be displayed in the descending order of their scores.

The root nodes of each of the departments need to be maintained on different machines. This ensures that several of your links will span two machines. All nodes within a given domain will be part of the graph hosted on a machine (or a set if you have assigned multiple machines for a domain).

**Additional Notes:**

1. Use of a database is NOT allowed.
2. You are allowed to use an external HTML parser which will allow you to discard html markups from your documents.  
If you are doing this assignment in Java such a parser is available at <http://htmlparser.sourceforge.net/>.
3. Use of any other third-part software is disallowed.
4. You need to report the total number of pages from each domain that you have crawled, and also the cumulative number of pages that been crawled.
5. You will also report the page rank scores for the top 250 pages/nodes.
6. Results must be presented in the order of their scores.