Xian Chen (25781347)
CS446 HW4
PageRank

Discussion

Due to the size of the corpus, the implemented PageRank program does not complete in a reasonable time frame. Steps were taken to reduce the memory cost of the program but the trade off is runtime performance issues. Therefore, there is output file of the top 50 links for the program (Part 2). If there was an output, the difference between the two lists is that the links that are PageRank takes into account context. Therefore, pages that have many source links (many links that point to that page) are ranked higher and because of this, if the popular pages point to another page, it is more heavily weighted in the calculation for that page's ranking. For example, lets consider the case where the popular site New York Times links to an unknown blog site and the case where a less popular site, lets say someone's personal website, links to that unknown blog. The unknown blog's page rank will be higher if New York Times links to it rather than the less popular personal webpage. Therefore the ranking of the first 50 links from Part1 and Part2 will definitely be quite different—especially towards the end of the list. Just finding the number of in links of a website is a good estimate for PageRank but it does not fair well in practice due to issues like link spam. Although link spam is still an issue in PageRank, PageRank deals with it much better.

Another difference between simply retrieving the pages with the most in links and retrieving the pages with the highest page rank is that computationally, finding the pages that has the most in links is much faster and less memory expensive. The output file was obtained in about 20 seconds after the start execution of the program (inLink.py). My first attempt in creating the PageRank algorithms calls for a series of lists—which became memory expensive. To meliorate this problem, I decided to load the links.srt file into a database and have the program loop over the series of tables in the database when it needs to compare or retrieve entries. Although this causes less strain on memory, looping over tables in a database proves to be quite slow, resulting in runtime issues. A better implementation will require the use of graphs. Thereby the program can just follow the vertices (webpages) of a graph rather than have it loop over a list or a file.

Included files:
Part 1: topInLinks.txt
Part 4: pagerank.py (links.db is not included due to its size)