Tufts University
Department of Mathematics
Math 50 Midterm Project 2

Due: Tuesday, November 22, at 1:30 p.m. (in class).

A major modern use of Markov Chains is as the basis of the PageRank algorithm used by Google to rank webpages when you search. In this project, you'll gather data for a portion of the web, create a series of transition processes modeling someone surfing that portion of the web, and make conclusions about the relative importance of the pages that you "searched".

Consider a basic model of surfing the web. You're looking at your favorite website (for example, `http://neumann.math.tufts.edu/~scott/math50`) and see that there are a number of links from this page to other pages. You follow one of these chosen at random (with equal probability for each page that is linked to, regardless of how many times the link appears). This brings you to a new page, where you do the same thing, and so on. This describes a transition process.

To model this transition process as a Markov Chain, you only need to count the number of unique out-links from each page to assign the probabilities. If page i has k out-links, then the probability of transitioning from i to a linked page j is $1/k$ (and the probability of transitioning from i to an unlinked page is 0).

To generate the "adjacency graph" of a piece of the web, download the file `http://www.mathworks.com/moler/ncm/surfer.m`. The syntax for this program is `[U,G] = surfer('http://www.mywebsite.here',N);`, where `http://www.mywebsite.here` is the website that you want to start from, and N is the number of websites that you want to include in your graph. The output comes in two pieces, U is an array of URLs of the websites surfed, and G is a graph of the connections between the websites. Your life will be easier if you immediately convert `G = full(G)*1.0`, which makes G a matrix of numbers instead of a collection of binary data.

Two pitfalls exist in this model. First, a page may have no out-links. In this case, you can assume that it is equally likely that you transition from a page with no out-links to any other page in your graph. Secondly, there is no guarantee that the graph that you generate will be irreducible. To avoid this case, you can assume that, if you are on a page with outlinks, there is only a fixed probability, p, that you stay on that page and, with probability $1 - p$ you navigate (at random) to any page in your graph (whether it is linked or not). This is known as the "random surfer" model where, at any time, you might randomly navigate to another page or, if you get stuck, you will randomly navigate to another page.

Your project is to pick a website, pick a reasonable N (for reference, it took my laptop (over wifi) about 30 minutes to generate the graph for 500 sites starting from `http://www.tufts.edu`), and compute the simplified PageRank transition matrix described above from the data that you gather, for various values of p. (The "usual" value of p is 0.85, but try both larger and smaller values.) For each value of p, compute the eigenvector with eigenvalue 1 using matlab's `eig` command, scale it to be a stationary probability vector, and report the probabilities and URLs for the 10 most probable webpages (the first page of "hits" for a generic search from your chosen website). Try to also compute the stationary probability vector directly using a power method starting from an initial probability distribution that says that you are at the URL you started at with probability 1 (and at all other URLs in your graph with probability 0). Discuss how this depends on p. Compare both the convergence of the values in the stationary probability vector and the convergence of the relative ranking of the most probable websites. Design an alternative to the "random surfer" model that you feel is an improvement on it, and see how using this affects your results.

Note: two good sources of information to help you get started on this are `http://en.wikipedia.org/wiki/PageRank` and `http://www.mathworks.com/moler/lu.pdf`