Technology for X-Informatics
PageRank

June 20 2013

Geoffrey Fox
gcf@indiana.edu
http://www.infomall.org/

Associate Dean for Research,
School of Informatics and Computing
Indiana University Bloomington
2013

Big Data Ecosystem in One Sentence

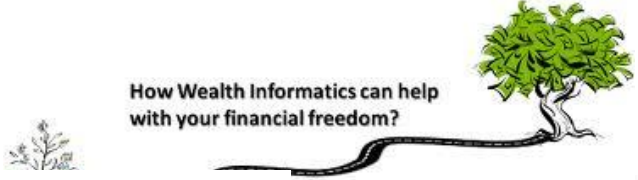
Use Clouds running Data Analytics Collaboratively processing Big Data to solve problems in X-Informatics (or e-X)

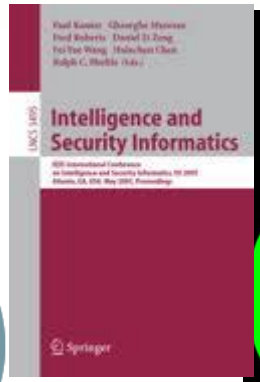
X = Astronomy, Biology, Biomedicine, Business, Chemistry, Climate, Crisis, Earth Science, Energy, Environment, Finance, Health, Intelligence, Lifestyle, Marketing, Medicine, Pathology, Policy, Radar, Security, Sensor, Social, Sustainability, Wealth and Wellness with more fields (physics) defined implicitly

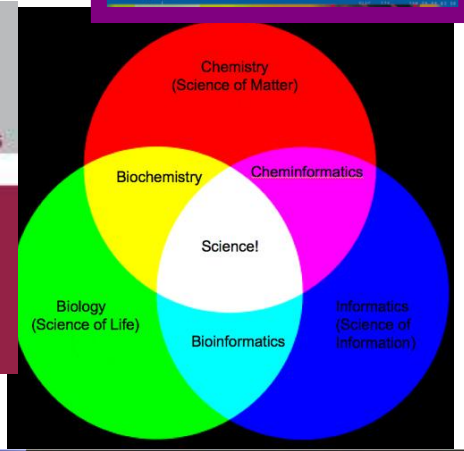
Spans Industry and Science (research)

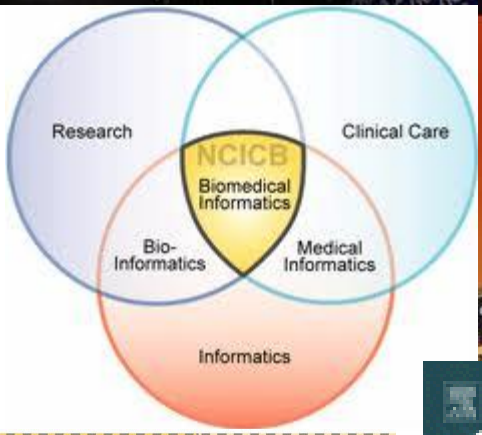
Education: Data Science see recent New York Times articles

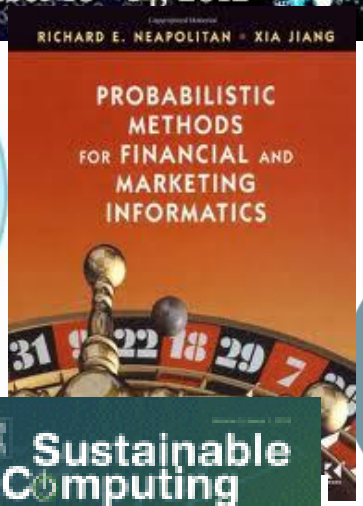
http://datascience101.wordpress.com/2013/04/13/new-york-times-data-science-articles/

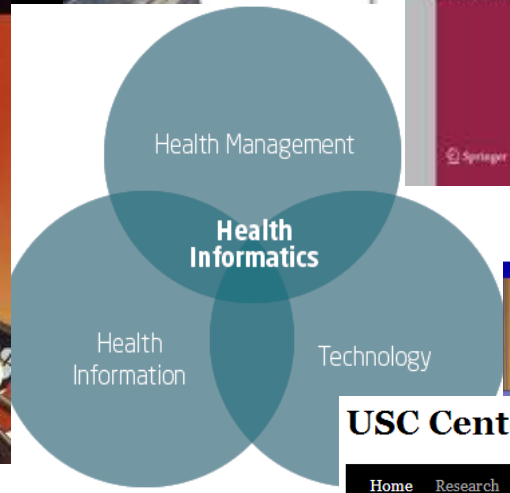


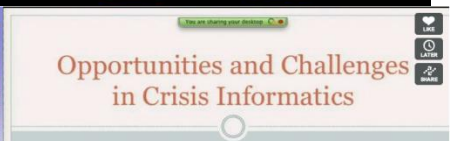


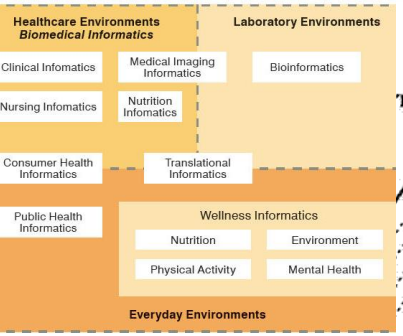


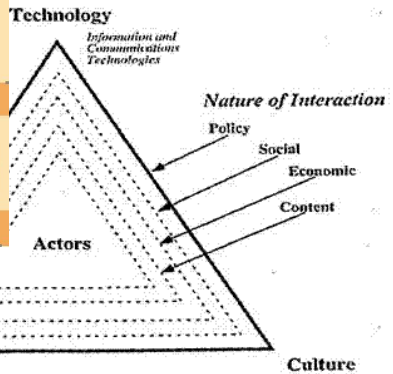




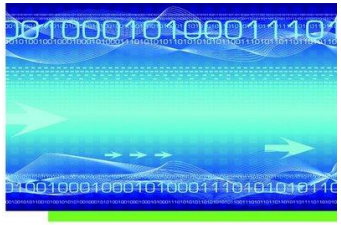








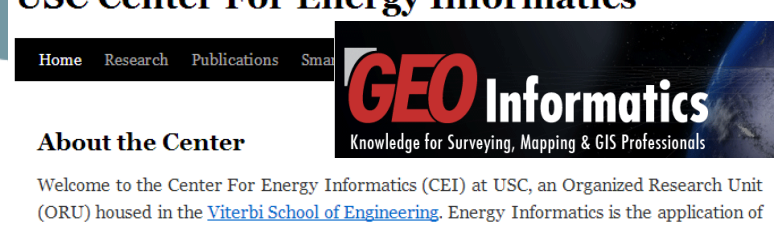
Social Informatics

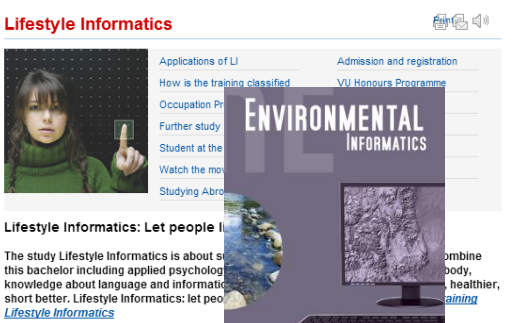


Noelia Penelope Greer (Ed.)

Business Informatics

Information technology, Management,





PageRank in Python

Calculate PageRank from Web Linkage Matrix

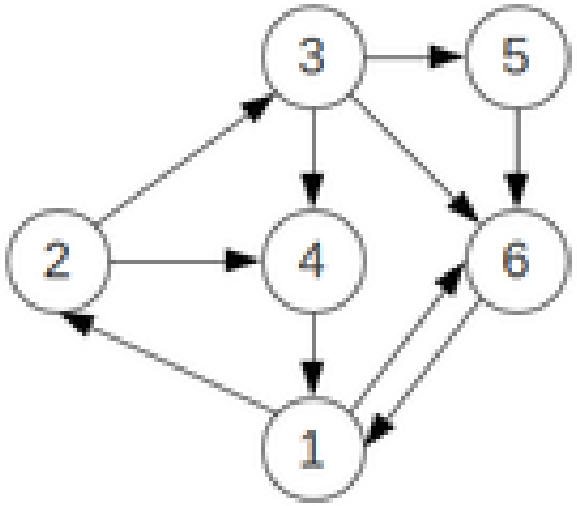
PageRank Python Resources

- **PageRank from an adjacency graph**
- http://glowingpython.blogspot.com/2011/05/four-ways-to-compute-google-pagerank.html
- http://www.rose-hulman.edu/~bryan/googleFinalVersionFixed.pdf
- See file pagerank2.py

- **PageRank for an Actual Page**
- **http://www.schurpf.com/google-pagerank-python/**
- http://coreygoldberg.blogspot.com/2010/01/python-lookup-google-pagerank-score.html
- See file pagerank1.py

The little Graph

- A = array([[0, 0, 0, 1, 0, 1],
- [1/2.0, 0, 0, 0, 0, 0],
- [0, 1/2.0, 0, 0, 0, 0],
- [0, 1/2.0, 1/3.0, 0, 0, 0],
- [0, 0, 1/3.0, 0, 0, 0],
- [1/2.0, 0, 1/3.0, 0, 1, 0]])



A(i,j) = 0 if page j does not point at page i
= 1/(Number of links on page j) otherwise
e.g. on first row, page 1 is pointed at by pages 4 and 6 which have no other links

Basic Formalism

- We create a damped array
  $M = d\,A + (1-d)\,S$
  where S is 6 by 6 matrix with all 1/6's
- d is usual value 0.85
- We set x_0 to be any vector normed to be 1 in sense sum of components 1 (not sum of square of components as components are probabilities) e.g. x_0 = ones(N)/N where N =6
- Then we need to iterate
  $x_0 \rightarrow Mx_0/\mathrm{Norm}(x_0)$

""" basic power method """

- def powerMethodBase(A,x0,iter):
  - for i in range(iter):
  - x0 = dot(A,x0)
  - x0 = x0/linalg.norm(x0,1)
  - return x0
- Better to test on change in x0 for stopping rather than guessing number of iterations as here

Matrix-Vector Power Method needs

- **numpy.linalg.norm**(x, option) Matrix or vector norm
  - For a 1D vector $\underline{x}$, it is $\sqrt{\{\sum_{i=0}^{N-1} x(i)^2\}}$ for option = 2 and $\sum_{i=0}^{N-1} |x(i)|$ for option = 1
  - We need Option 1
- **numpy.dot(a, b)** Dot product of two arrays.
  - For 2-D arrays it is equivalent to matrix multiplication, and for 1-D arrays to inner product of vectors (without complex conjugation). For N dimensions it is a sum product over the last axis of a and the second-to-last of b.
  - If a 2D and b 1D, it is matrix vector multiplication c = ab with $c(i) = \sum_{j=0}^{M-1} a(i, j)\ b(j)$

""" Modified basic power method """

- def powerMethodBase1(A,x0,thresh,iter):
  - x0 = x0/linalg.norm(x0,1)
  - x1 = x0 # The previous value of x0
  - for i in range(iter):
    - x0 = dot(A,x0)
    - x0 = x0/linalg.norm(x0,1)
    - stop = linalg.norm(x0-x1,1)
    - if stop < thresh:
      - print "Stop ",stop, " After ", i, " iterations"
      - break
    - x1 = x0
  - return x0

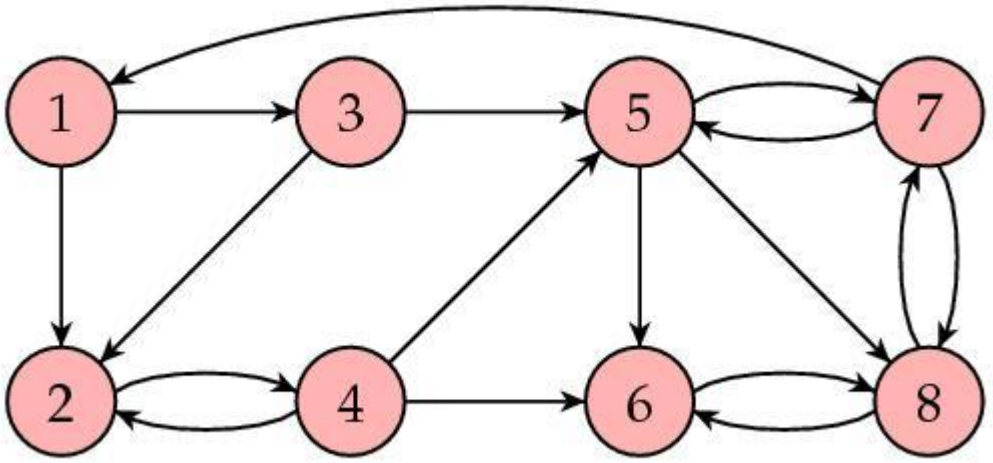
""" Original basic power method """

- def powerMethodBase1(A, x0, iter):
  - x0 = x0/linalg.norm(x0,1)
  - for i in range(iter):
    - x0 = dot(A,x0)
    - x0 = x0/linalg.norm(x0,1)
  - return x0
- They use 130 iterations; 20 gives fine answers for first matrix as seen in modified version
- More iterations needed for second case

Another 8 by 8 Matrix

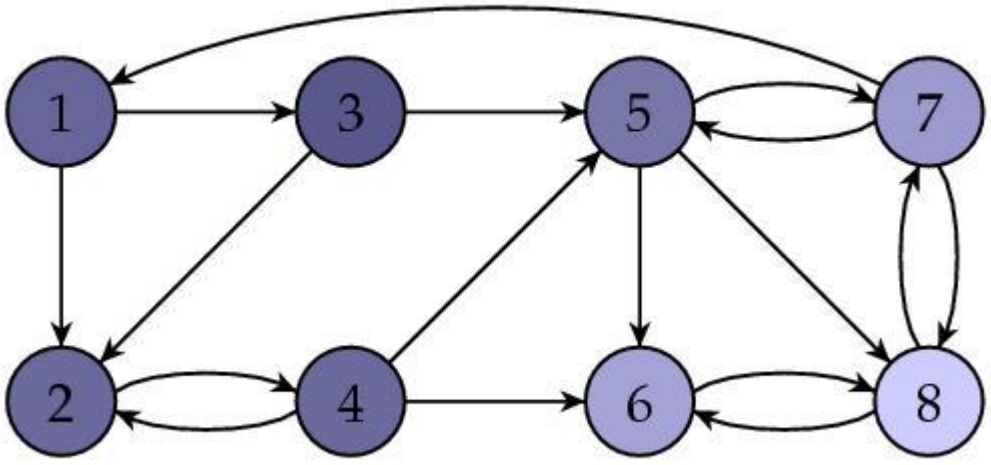
- See http://www.ams.org/samplings/feature-column/fcarc-pagerank
- Run with d = 1 (no damping)

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 \\ 1/2 & 0 & 1/2 & 1/3 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 1 & 1/3 & 0 \end{bmatrix}$$



Another 8 by 8 Matrix Contd.

- Result: 0.06 0.0675 0.03 0.0675 0.0975 0.2025 0.18 0.295
- This shows that page 8 wins the popularity contest. Here is the same figure with the web pages shaded in such a way that the pages with higher PageRanks are lighter.



3 Other Methods I

- We are iteratively solving the equations
- $x = dAx + (1-d)S_2x$ where S_2 is N by N matrix of 1's
- That's essentially **powerMethodBase** approach
- Now $\Sigma_{i=0}^{N-1} x(i) = 1$ (Probabilities add to one)
- So S_2x always = S_1 = [1, 1, 1] – the vector of 1's
- There we can iteratively solve
  $x = dAx + (1-d)S_1$
  - This is method **powerMethod**

3 Other Methods II (Last 2 Methods)

- Alternatively we can write $x = dAx + (1-d)S_1$
- as $(1- dA)x = (1-d)S_1$ and solve these N linear equations
  - This is method **linearEquations** and only works if $d<1$
  - I don't think it's a serious method for real problem
- Finally an equation like
- $x = [dA + (1-d)S_2]x$ is a special case of Matrix $[dA + (1-d)S_2]\ x = \lambda\ x$ which is an equation for eigenvalue λ and eigenvector x.
- The matrix we have is very special and has eigenvalue $\lambda = 1$ and this is largest eigenvalue
- Thus final method finds largest eigenvalue with general eigenvalue routine **maximalEigenvector**
  - This is "too general" for practical use
- In fact best way of finding largest eigenvalue of any matrix is the power method **powerMethodBase** and one does not use general method if you just need largest eigenvalue

PageRank in Real World

- The matrix A is a sparse matrix
- Number of pages ~25 Billion
- The number of links per page is ~20 http://uclue.com/?xq=1015
- Can use MapReduce and run in parallel with each "map task" responsible for a fraction of web pages
- *From:* http://www.ams.org/samplings/feature-column/fcarc-pagerank
- *With the value of chosen to be near 0.85, Brin and Page report that 50 - 100 iterations are required to obtain a sufficiently good approximation to PageRank. The calculation is reported to take a few days to complete.*
- *It is rumored that Google recomputes the PageRank vector roughly every month. Since the PageRank of pages can be observed to fluctuate considerably during this time, it is known to some as the Google Dance*

PageRank in Python

Calculate PageRank of a real page

Using Pagerank1.py

- You just change last line in script to fetch PageRank of a given URL
- print GetPageRank("http://www.infomall.org") returns 5
- This PageRank is modified (by Google)from formal definition as an integer between 1 and 10 and not a probability <1