COMP3220 — Document Processing and the Semantic Web

Week 02 Lecture 2: Web Search

Diego Mollá

Department of Computer Science
Macquarie University

COMP3220 2020H1

Programme

Reading

- Lecture Notes
- Tanase & Radu's Lecture on PageRank algorithm: http://www.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture3/lecture3.html

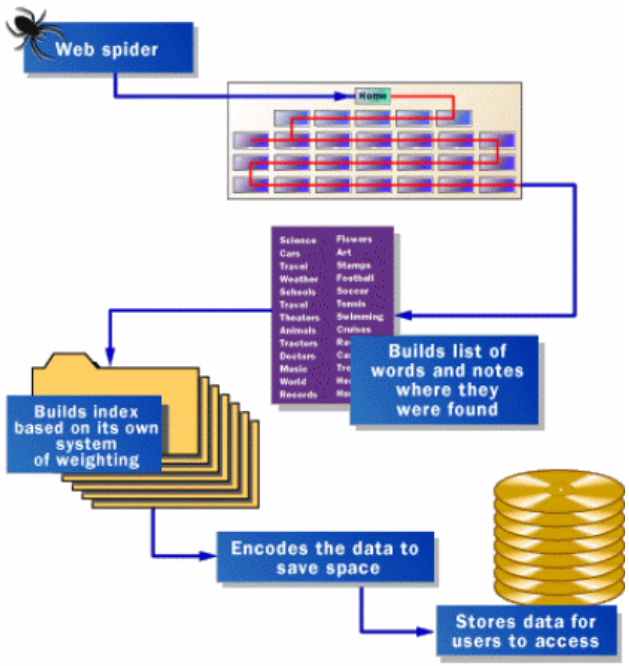
Additional Resources

- Brin and Page (1998), "The Anatomy of a Large-Scale Hypertextual Web Search Engine" — by the founders of Google http://infolab.stanford.edu/~backrub/google.html

Programme

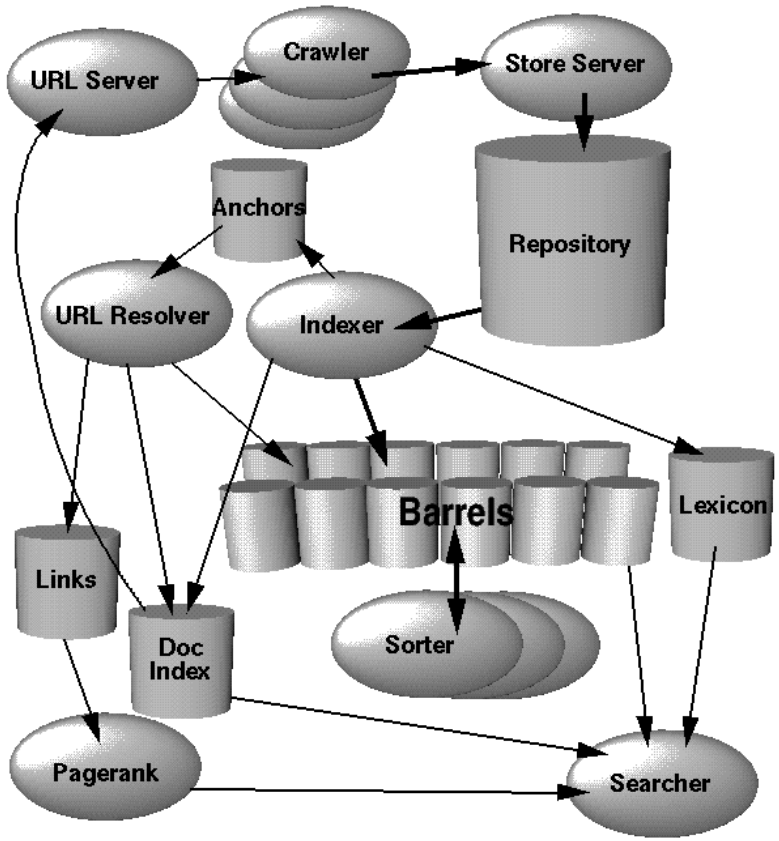
Crawling the Web

- Web search engines keep an off-line snapshot of the Web.
- This snapshop is created and maintained by crawlers.
- Crawlers (spiders, ants, ...) are programs that fetch Web pages.



Scaling to the Web

- Crawling and indexing a collection of documents on itself is not difficult.
- The challenge is to scale to the entire Web.
- Main Web search engines employ massive parallel processes for distributed crawling and indexing.



Programme

Ranking Search Results

- Google's original search method was a simple Boolean search: "find all documents that contain all the query terms".
  - Google's current search method is a trade secret.
- A key innovation from Google was how to rank the results.
  - Another key innovation was how to scale up to the entire Web.
  - (Google's real success was how to monetise Web search but this is not the topic of this lecture.)
- The PageRank algorithm determines the importance of a page regardless of the query.
- The importance of a page is determined by how well linked it is.

PageRank

- The Web can be seen as a graph.
- The nodes are the HTML pages, the edges are the hyperlinks.
- PageRank can be defined recursively.

Formula of PageRank

$$PR(A) = \frac{1-d}{N} + d(\frac{PR(T_1)}{C(T_1)} + \cdots + \frac{PR(T_n)}{C(T_n)})$$

N = total number of documents;
T_i = page that links to A;
$C(T_i)$ = outgoing links from page T_i.

PageRank and Random Walks

- The PageRank of a page represents the probability of arriving at that page after a long sequence of random clicks.
- This "random surfer" will follow one of the links from a page with probability d: the "damping factor" (usually $d = 0.85$).
- Otherwise (i.e. with probability $1 - d$) the surfer goes to any page randomly.

$$PR(A) = \frac{1-d}{N} + d(\frac{PR(T_1)}{C(T_1)} + \cdots + \frac{PR(T_n)}{C(T_n)})$$

The formula is a variant of the eigenvector centrality measure used in network analysis.

PageRank and Random Walks

- The PageRank of a page represents the probability of arriving at that page after a long sequence of random clicks.
- This "random surfer" will follow one of the links from a page with probability d: the "damping factor" (usually $d = 0.85$).
- Otherwise (i.e. with probability $1 - d$) the surfer goes to any page randomly.

$$PR(A) = \frac{1-d}{N} + d\left(\frac{PR(T_1)}{C(T_1)} + \cdots + \frac{PR(T_n)}{C(T_n)}\right)$$

The formula is a variant of the eigenvector centrality measure used in network analysis.

PageRank and Random Walks

- The PageRank of a page represents the probability of arriving at that page after a long sequence of random clicks.
- This "random surfer" will follow one of the links from a page with probability d: the "damping factor" (usually $d = 0.85$).
- Otherwise (i.e. with probability $1 - d$) the surfer goes to any page randomly.

$$PR(A) = \frac{1-d}{N} + d\left(\frac{PR(T_1)}{C(T_1)} + \cdots + \frac{PR(T_n)}{C(T_n)}\right)$$

The formula is a variant of the eigenvector centrality measure used in network analysis.

Computing PageRank

- PageRank can be computed iteratively.
- At first iteration, we assume all nodes have same weight $1/N$.
- We then apply the formula to spread the weights to the neighbours.
  - We apply the formula until the PageRank values do not change.
- Given that the inherent graph is very sparse, only a few iterations are needed to converge.

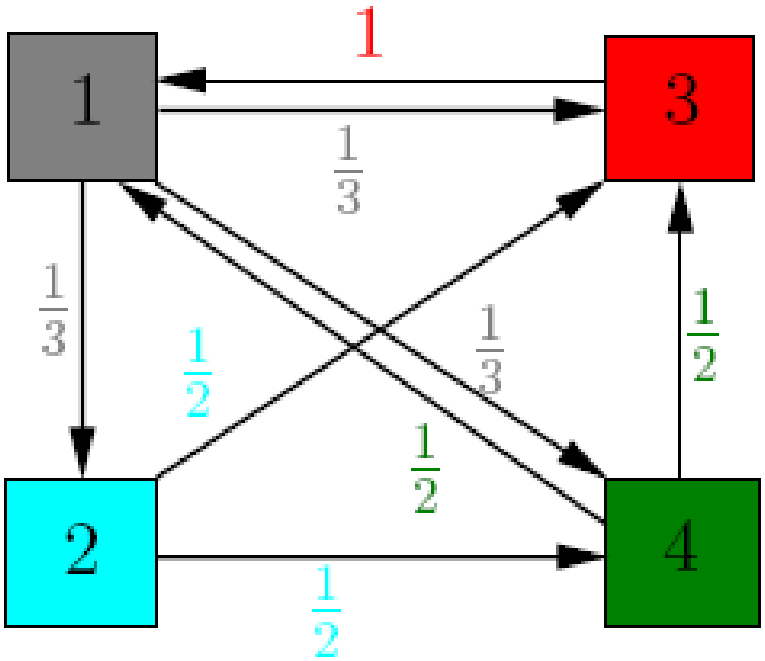
The Mathematics of Google Search

http://www.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture3/lecture3.html

The Transition Matrix of a Graph

We can express a graph as a matrix.

- Columns, rows: nodes.
- Cell(j,i): weight of the edge from node i to node j.



$$A = \begin{pmatrix} 0 & 0 & 1 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \end{pmatrix}$$

$A(j,i) = \frac{1}{C(i)}$ if there's a link from i to j.

Adding the Damping Factor

Adjusted Transition Matrix

$$M = d \times A + (1 - d) \times B$$

$$B = \frac{1}{4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$(1 - d) \times B$ corresponds to the term $\frac{1-d}{N}$ in the PageRank formula:

$$PR(A) = \frac{1 - d}{N} + d(\frac{PR(T_1)}{C(T_1)} + \cdots + \frac{PR(T_n)}{C(T_n)})$$

Spreading the Weights

Using the transition matrix and an initial set of weights, we only need to iteratively multiply the matrix with the weights (using the dot product between the matrix and the vector).

$$PR = M \cdot PR$$

Proof

$$\begin{aligned} PR(i) &= M(i,1) \times PR(1) + M(i,2) \times PR(2) + \cdots \\ &= (\tfrac{1-d}{N} + d \times A(i,1)) \times PR(1) + \\ &\quad (\tfrac{1-d}{N} + d \times A(i,2)) \times PR(2) + \cdots \\ &= \tfrac{1-d}{N} + d(\tfrac{PR(1)}{C(1)} + \tfrac{PR(2)}{C(2)} + \cdots) \end{aligned}$$

(note that $PR(1) + PR(2) + \cdots = 1$)

An Iteration in Python

```
>>> import numpy as np
>>> A = np.array([[0.,    0.,    1., 1./2.],
                  [1./3., 0.,    0., 0.   ],
                  [1./3., 1./2., 0., 1./2.],
                  [1./3., 1./2., 0., 0.   ]])
>>> M = 0.85*A + 0.15*(1./4.*np.ones((4,4))
>>> PR = 1./4.*np.ones((4,1))
>>> PR = np.dot(M,PR)
>>> PR
array([[ 0.35625  ],
       [ 0.10833333],
       [ 0.32083333],
       [ 0.21458333]])
```

Iteration Until Convergence in Python I

Code

```
epsylon = 0.01
iterations = 0
PR = 1./4.*np.ones((4,1))
oldPR = np.zeros((4,1))
while max(np.abs(oldPR-PR)) > epsylon:
    oldPR = PR
    PR = np.dot(M,PR)
    iterations +=1
print "PR after",iterations,"iterations:"
print PR
```

Iteration Until Convergence in Python II

Output

```
PR after 5 iterations:
[[ 0.36966846]
 [ 0.14289417]
 [ 0.28643227]
 [ 0.2010051 ]]
```

Take-home Messages

1. What is crawling? indexing? ranking?
2. What is the PageRank formula?
3. Implement PageRank in Python.

What's Next

Week 3

- Introduction to Statistical Classification.
- Submit Assignment 1 by Friday week 3.

Reading

- NLTK Chapter 6.
- Manning et al. Chapter 14.