CS347

Lecture 6

April 25, 2001

©Prabhakar Raghavan

Today's topic

- Link-based ranking in web search engines

Web idiosyncrasies

- Distributed authorship
  - Millions of people creating pages with their own style, grammar, vocabulary, opinions, facts, falsehoods …
  - Not all have the purest motives in providing high-quality information - commercial motives drive "spamming".
  - The open web is largely a marketing tool.
    - IBM's home page does not contain ***computer***.

More web idiosyncrasies

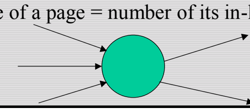
- Some pages have little or no text (gifs may embed text)
- Variety of languages, lots of distinct terms
  - Over 100M distinct "terms"!
- Long lists of links
- Size: >1B pages, each with ~1K terms.
  - Growing at a few million pages/day.

Link analysis

- Two basic approaches
  - Universal, query-independent ordering on all web pages (based on link analysis)
    - Of two pages meeting a (text) query, one will always win over the other, *regardless* of the query
  - Query-specific ordering on web pages
    - Of two pages meeting a query, the relative ordering may vary from query to query

Query-independent ordering

- First generation: using link counts as simple measures of popularity.
- Two basic suggestions:
  - Undirected popularity:
    - Each page gets a score = the number of in-links plus the number of out-links (3+2=5).
  - Directed popularity:
    - Score of a page = number of its in-links (3).



Query processing

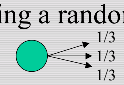
- First retrieve all pages meeting the text query (say ***venture capital***).
- Order these by their link popularity (either variant on the previous page).

Spamming simple popularity

- *Exercise*: How do you spam each of the following heuristics so your page gets a high score?
- Each page gets a score = the number of in-links plus the number of out-links.
- Score of a page = number of its in-links.

Pagerank scoring

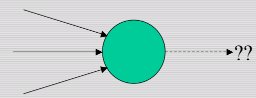
- Imagine a browser doing a random walk on web pages:
  - Start at a random page
  - At each step, go out of the current page along one of the links on that page, equiprobably



- "In the steady state" each page has a long-term visit rate - use this as the page's score.

Not quite enough

- The web is full of dead-ends.
  - Random walk can get stuck in dead-ends.
  - Makes no sense to talk about long-term visit rates.



Teleporting

- At each step, with probability 10%, jump to a random web page.
- With remaining probability (90%), go out on a random link.
  - If no out-link, stay put in this case.

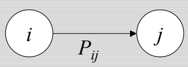
Result of teleporting

- Now cannot get stuck locally.
- There is a long-term rate at which any page is visited (not obvious, will show this).
- How do we compute this visit rate?

Markov chains

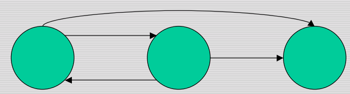
- A Markov chain consists of n <u>states</u>, plus an $n \times n$ <u>transition probability matrix</u> **P**.
- At each step, we are in exactly one of the states.
- For $1 \le i,j \le n$, the matrix entry P_{ij} tells us the probability of j being the next state, given we are currently in state i.

$P_{ii}>0$ is OK.



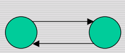
Markov chains

- Clearly, for all i, $\sum_{j=1}^{n} P_{ij} = 1.$
- Markov chains are abstractions of random walks.
- *Exercise*: represent the teleporting random walk from 3 slides ago as a Markov chain, for this case:



Ergodic Markov chains

- A Markov chain is <u>ergodic</u> if
  - you have a path from any state to any other
  - you can be in any state at every time step, with non-zero probability.



Not ergodic (even/ odd).

Ergodic Markov chains

- For any ergodic Markov chain, there is a unique long-term visit rate for each state.
  - *Steady-state distribution*.
- Over a long time-period, we visit each state in proportion to this rate.
- <u>It doesn't matter where we start.</u>

Probability vectors

- A probability vector $\mathbf{x} = (x_1, \ldots x_n)$ tells us where the walk is at any point.
- E.g., $(\underset{1}{0}00\ldots\underset{i}{1}\ldots00\underset{n}{0})$ means we're in state i.

More generally, the vector $\mathbf{x} = (x_1, \ldots x_n)$ means the walk is in state i with probability x_i.

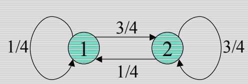
$$\sum_{i=1}^{n} x_i = 1.$$

Change in probability vector

- If the probability vector is $\mathbf{x} = (x_1, \ldots x_n)$ at this step, what is it at the next step?
- Recall that row i of the transition prob. Matrix **P** tells us where we go next from state i.
- So from **x**, our next state is distributed as **xP**.

Computing the visit rate

- The steady state looks like a vector of probabilities $\mathbf{a} = (a_1, \ldots a_n)$:
  - a_i is the probability that we are in state i.



For this example, $a_1=1/4$ and $a_2=3/4$.

How do we compute this vector?

- Let $\mathbf{a} = (a_1, \ldots a_n)$ denote the row vector of steady-state probabilities.
- If we our current position is described by **a**, then the next step is distributed as **aP**.
- But **a** is the steady state, so **a**=**aP**.
- Solving this matrix equation gives us **a**.
  - (So **a** is the (left) eigenvector for **P**.)

Another way of computing **a**

- Recall, regardless of where we start, we eventually reach the steady state **a**.
- Start with any distribution (say **x**=(*10…0*)).
- After one step, we're at **xP**;
- after two steps at $\mathbf{xP}^2$, then $\mathbf{xP}^3$ and so on.
- "Eventually" means for "large" *k*, $\mathbf{xP}^k = \mathbf{a}$.
- Algorithm: multiply **x** by increasing powers of **P** until the product looks stable.

Pagerank summary

- Preprocessing:
  - Given graph of links, build matrix **P**.
  - From it compute **a**.
  - The entry a_i is a number between 0 and 1: the pagerank of page *i*.
- Query processing:
  - Retrieve pages meeting query.
  - Rank them by their pagerank.
  - Order is query-*independent*.

The reality

- Pagerank is used in google, but so are many other clever heuristics
  - more on these heuristics later.

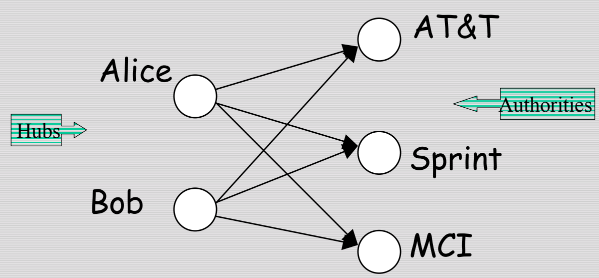
Query-dependent link analysis

- In response to a query, instead of an ordered list of pages each meeting the query, find <u>two</u> sets of inter-related pages:
  - *Hub pages* are good lists of links on a subject.
    - e.g., "Bob's list of cancer-related links."
  - *Authority pages* occur recurrently on good hubs for the subject.

Hubs and Authorities

- Thus, a good hub page for a topic *points* to many authoritative pages for that topic.
- A good authority page for a topic is *pointed* to by many good hubs for that topic.
- Circular definition - will turn this into an iterative computation.

The hope



Long distance telephone companies

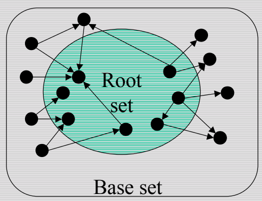
High-level scheme

- Extract from the web a <u>base set</u> of pages that *could* be good hubs or authorities.
- From these, identify a small set of top hub and authority pages;
  - iterative algorithm.

Base set

- Given text query (say ***browser***), use a text index to get all pages containing ***browser.***
  - Call this the <u>root set</u> of pages.
- Add in any page that either
  - points to a page in the root set, or
  - is pointed to by a page in the root set.
- Call this the base set.

Visualization



Assembling the base set

- Root set typically 200-1000 nodes.
- Base set may have up to 5000 nodes.
- How do you find the base set nodes?
  – Follow out-links by parsing root set pages.
  – Get in-links (and out-links) from a *connectivity server*.
  – (Actually, suffices to text-index strings of the form ***href="<u>URL</u>"*** to get in-links to <u>*URL*</u>.)

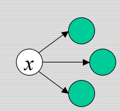
Distilling hubs and authorities

- Compute, for each page *x* in the base set, a <u>hub score</u> *h(x)* and an <u>authority score</u> *a(x)*.
- Initialize: for all *x, h(x)←1; a(x) ←1*;
- Iteratively update all *h(x), a(x)*; ⇦Key
- After iteration, output pages with highest *h()* scores as top hubs; highest *a()* scores as top authorities.

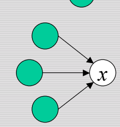
Iterative update

- Repeat the following updates, for all *x*:

$$h(x) \leftarrow \sum_{x \rightarrow y} a(y)$$



$$a(x) \leftarrow \sum_{y \rightarrow x} h(y)$$



Scaling

- To prevent the *h()* and *a()* values from getting too big, can scale down after each iteration.
- Scaling factor doesn't really matter:
  - we only care about the relative values of the scores.

How many iterations?

- Claim: relative values of scores will converge after a few iterations:
  - in fact, suitably scaled, *h()* and *a()* scores settle into a steady state!
  - proof of this comes later.
- In practice, ~5 iterations get you close to stability.

Japan Elementary Schools

Authorities

- The American School in Japan
- The Link Page
- ‰ª□è□s—§ˆä“c□¬Šw□Zƒz□[ƒ□ƒy□[ƒW
- Kids' Space
- ˆÀ□é□s—§ˆÀ□é□¼•”□¬Šw□Z
- ‹{□é‹³ˆç‘åŠw•□‘®□¬Šw□Z
- KEIMEI GAKUEN Home Page (Japanese)
- Shiranuma Home Page
- fuzoku-es.fukui-u.ac.jp
- welcome to Miasa E&J school
- □_“Þ□ìŒ§□E‰¡•l□s—§’†□ì□¼□¬Šw□Z,Ìƒy
- http://www...p/~m_maru/index.html
- fukui haruyama-es HomePage
- Torisu primary school
- goo
- Yakumo Elementary,Hokkaido,Japan
- FUZOKU Home Page
- Kamishibun Elementary School...

Hubs

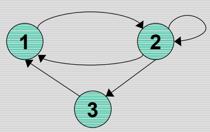
- schools
- LINK Page-13
- “ú–{,ÌŠw□Z
- □a‰„□¬Šw□Zƒz□[ƒ□ƒy□[ƒW
- 100 Schools Home Pages (English)
- K-12 from Japan 10/...rnet and Education)
- http://www...iglobe.ne.jp/~IKESAN
- ,l,f,j□¬Šw□Z,U”N,P‘g•¨Œê
- □ÒŠ—’¬—§□ÒŠ—“Œ□¬Šw□Z
- Koulutus ja oppilaitokset
- TOYODA HOMEPAGE
- Education
- Cay's Homepage(Japanese)
- –y“ì□¬Šw□Z,Ìƒz□[ƒ□ƒy□[ƒW
- UNIVERSITY
- ‰J—³□¬Šw□Z DRAGON97-TOP
- □Â‰ª□¬Šw□Z,T”N,P‘gƒz□[ƒ□ƒy□[ƒW
- ¶µ°é¼ÂÁ© ¥á¥Ë¥å¡¼ ¥á¥Ë¥å¡¼

Things to note

- Pulled together good pages regardless of language of page content.
- Use *only* link analysis after base set assembled
  - iterative scoring is query-independent.
- Iterative computation after text index retrieval - significant overhead.

Proof of convergence

- $n \times n$ adjaceny matrix **A**:
  - each of the n pages in the base set has a row and column in the matrix.
  - Entry $A_{ij} = 1$ if page i links to page j, else =0.



	1	2	3
1	0	1	0
2	1	1	1
3	1	0	0

Hub/authority vectors

- View the hub scores *h()* and the authority scores *a()* as vectors with n components.
- Recall the iterative updates

$$h(x) \leftarrow \sum_{x \rightarrow y} a(y)$$

$$a(x) \leftarrow \sum_{y \rightarrow x} h(y)$$

Rewrite in matrix form

- **h**=**Aa**.
- **a**=**A**^{t}**h**.

Recall A^t is the transpose of A.

Substituting, **h**=**AA**^{t}**h** and **a**=**A**^{t}**Aa**.

Thus, **h** is an eigenvector of **AA**^{t} and **a** is an eigenvector of **A**^{t}**A**.

Resources

- MIR 13
- The Anatomy of a Large-Scale Hypertextual Web Search Engine
  - *http://citeseer.nj.nec.com/brin98anatomy.html*
- Authoritative Sources in a Hyperlinked Environment
  - *http://citeseer.nj.nec.com/kleinberg97authoritative.html*
- Hypersearching the Web
  - *http://www.sciam.com/1999/0699issue/0699raghavan.html*
- Dubhashi resource collection covering recent topics
  - *http://www.cs.chalmers.se/~dubhashi/Courses/intense00.html*