CS345
Data Mining

Link Analysis 2
Page Rank Variants

Anand Rajaraman, Jeffrey D. Ullman

Topics

- This lecture
  - Many-walkers model
  - Tricks for speeding convergence
  - Topic-Specific Page Rank

Random walk interpretation

- At time 0, pick a page on the web uniformly at random to start the walk
- Suppose at time t, we are at page j
- At time t+1
  - With probability β, pick a page uniformly at random from O(j) and walk to it
  - With probability 1-β, pick a page on the web uniformly at random and teleport into it
- Page rank of page p = "steady state" probability that at any given time, the random walker is at page p

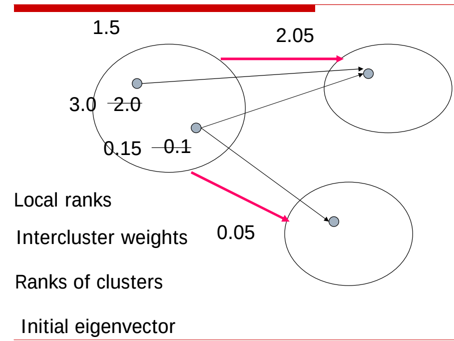
Many random walkers

- Alternative, equivalent model
- Imagine a large number M of independent, identical random walkers ($M \gg N$)
- At any point in time, let M(p) be the number of random walkers at page p
- The page rank of p is the fraction of random walkers that are expected to be at page p i.e., **E**[M(p)]/M.

Speeding up convergence

- Exploit locality of links
  - Pages tend to link most often to other pages within the same host or domain
- Partition pages into clusters
  - host, domain, …
- Compute local page rank for each cluster
  - can be done in parallel
- Compute page rank on graph of clusters
- Initial rank of a page is the product of its local rank and the rank of its cluster
  - Use as starting vector for normal page rank computation
  - 2-3x speedup

In Pictures



Other tricks

- Adaptive methods
- Extrapolation
- Typically, small speedups
  - ~20-30%

Problems with page rank

- Measures generic popularity of a page
  - Biased against topic-specific authorities
  - Ambiguous queries e.g., jaguar
  - This lecture
- Uses a single measure of importance
  - Other models e.g., hubs-and-authorities
  - Next lecture
- Susceptible to Link spam
  - Artificial link topographies created in order to boost page rank
  - Next lecture

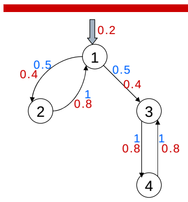
Topic-Specific Page Rank

- Instead of generic popularity, can we measure popularity within a topic?
  - E.g., computer science, health
- Bias the random walk
  - When the random walker teleports, he picks a page from a set S of web pages
  - S contains only pages that are relevant to the topic
  - E.g., Open Directory (DMOZ) pages for a given topic (www.dmoz.org)
- For each teleport set S, we get a different rank vector $\mathbf{r}_S$

Matrix formulation

- $A_{ij} = \beta M_{ij} + (1-\beta)/|S|$ if $i \in S$
- $A_{ij} = \beta M_{ij}$ otherwise
- Show that **A** is stochastic
- We have weighted all pages in the teleport set S equally
  - Could also assign different weights to them

Example



Suppose $S = \{1\}$, $\beta = 0.8$

Node	Iteration 0	1	2…	stable
1	1.0	0.2	0.52	0.294
2	0	0.4	0.08	0.118
3	0	0.4	0.08	0.327
4	0	0	0.32	0.261

Note how we initialize the page rank vector differently from the unbiased page rank case.

How well does TSPR work?

- Experimental results [Haveliwala 2000]
- Picked 16 topics
  - Teleport sets determined using DMOZ
  - E.g., arts, business, sports,…
- "Blind study" using volunteers
  - 35 test queries
  - Results ranked using Page Rank and TSPR of most closely related topic
  - E.g., bicycling using Sports ranking
  - In most cases volunteers preferred TSPR ranking

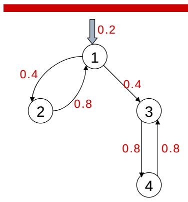
Which topic ranking to use?

- User can pick from a menu
- Use Bayesian classification schemes to classify query into a topic
- Can use the context of the query
  - E.g., query is launched from a web page talking about a known topic
  - History of queries e.g., “basketball” followed by “jordan”
- User context e.g., user’s My Yahoo settings, bookmarks, …

Evaporation model

- Alternative, equivalent interpretation of page rank
  - Instead of random teleport
- Assume random surfers “evaporate” from each page at rate $(1-\beta)$ per time step
  - those surfers vanish from the system
- New random surfers enter the system at the teleport set pages
  - Total of $(1-\beta)M$ at each step
- System reaches stable state
  - evaporation at each time step = number of new surfers at each time step

Evaporation-based computation



Suppose S = {1}, β = 0.8

Node	Iteration 0	1	2…	stable
1	0.2	0.2	0.264	0.294
2	0	0.08	0.08	0.118
3	0	0.08	0.08	0.327
4	0	0	0.064	0.261

Note how we initialize the page rank vector differently in this model

Scaling with topics and users

- Suppose we wanted to cover 1000’s of topics
  - Need to compute 1000’s of different rank vectors
  - Need to store and retrieve them efficiently at query time
  - For good performance vectors must fit in memory
- Even harder when we consider personalization
  - Each user has their own teleport vector
  - One page rank vector per user!

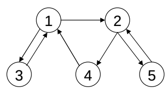
Tricks

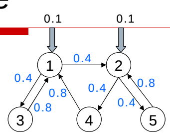
- Determine a set of basis vectors so that any rank vector is a linear combination of basis vectors
- Encode basis vectors compactly as partial vectors and a hubs skeleton
- At runtime perform a small amount of computation to derive desired rank vector elements

Linearity Theorem

- Let S be a teleport set and $\mathbf{r}_S$ be the corresponding rank vector
- For page $i \in S$, let $\mathbf{r}_i$ be the rank vector corresponding to the teleport set {i}
  - $\mathbf{r}_i$ is a vector with N entries
- $\mathbf{r}_S = (1/|S|) \sum_{i \in S} \mathbf{r}_i$
- Why is linearity important?
  - Instead of 2^N biased page rank vectors we need to store N vectors

Linearity example

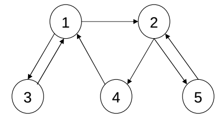




Let us compute $r_{\{1,2\}}$ for $\beta = 0.8$

Node	Iteration 0	1	2…	stable
1	0.1	0.1	0.164	0.300
2	0.1	0.14	0.172	0.323
3	0	0.04	0.04	0.120
4	0	0.04	0.056	0.130
5	0	0.04	0.056	0.130

Linearity example



$r_{\{1,2\}}$	r_1	r_2	$(r_1+r_2)/2$
0.300	0.407	0.192	0.300
0.323	0.239	0.407	0.323
0.120	0.163	0.077	0.120
0.130	0.096	0.163	0.130
0.130	0.096	0.163	0.130

Intuition behind proof

- Let's use the many-random-walkers model with M random walkers
- Let us color a random walker with color i if his most recent teleport was to page i
- At time t, we expect M/|S| of the random walkers to be colored i
- At any page j, we would therefore expect to find $(M/|S|)r_i(j)$ random walkers colored i
- So total number of random walkers at page j = $(M/|S|)\Sigma_{i \in S} r_i(j)$

Basis Vectors

- Suppose T = union of all teleport sets of interest
  - Call it the teleport universe
- We can compute the rank vector corresponding to any teleport set $S \subseteq T$ as a linear combination of the vectors $\mathbf{r_i}$ for $i \in T$
- We call these vectors the basis vectors for T
- We can also compute rank vectors where we assign different weights to teleport pages

Decomposition

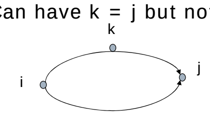
- Still too many basis vectors
  - E.g., |T| might be in the thousands
  - N|T| values
- Decompose basis vectors into partial vectors and hubs skeleton

Tours

- Consider a random walker with teleport set {i}
  - Suppose walker is currently at node j
- The random walker's tour is the sequence of nodes on the walker's path since the last teleport
  - E.g., i,a,b,c,a,j
  - Nodes can repeat in tours – why?
- Interior nodes of the tour = {a,b,c}
- Start node = {i}, end node = {j}
  - A page can be both start node and interior node, etc

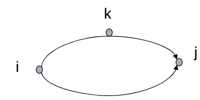
Tour splitting

- Consider random walker with teleport set {i}, biased rank vector r_i
- $r_i(j)$ = probability random walker reaches j by following some tour with start node i and end node j
- Consider node k
  - Can have k = j but not k = i

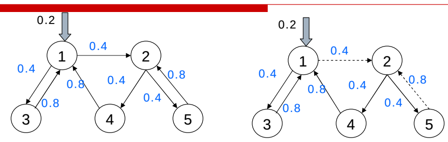


Tour splitting

- Let $r_i^k(j)$ be the probability that random surfer reaches page j through a tour that includes page k as an interior or end node.
- Let $r_i^{\sim k}(j)$ be the probability that random surfer reaches page j through a tour that does not include k as an interior or end node.
- $r_i(j) = r_i^k(j) + r_i^{\sim k}(j)$



Example

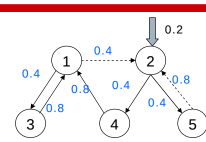


Let us compute $r_1^{\sim 2}$ for $\beta = 0.8$

Node	Iteration 0	1	2…	stable
1	0.2	0.2	0.264	0.294
2	0	0	0	0
3	0	0.08	0.08	0.118
4	0	0	0	0
5	0	0	0	0

Note that many entries are zeros

Example



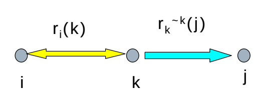
Let us compute $r_2^{\sim 2}$ for $\beta = 0.8$

Node	Iteration 0	1	2…	stable
1	0	0	0.064	0.094
2	0.2	0.2	0.2	0.2
3	0	0	0	0.038
4	0	0.08	0.08	0.08
5	0	0.08	0.08	0.08

Rank composition

- Notice:
  - $r_1^2(3) = r_1(3) - r_1^{\sim 2}(3)$
    $= 0.163 - 0.118 = 0.045$
  - $r_1(2) * r_2^{\sim 2}(3) = 0.239 * 0.038$
    $= 0.009$
    $= 0.2 * 0.045$
    $= (1-\beta) * r_1^2(3)$
  - $r_1^2(3) = r_1(2)\ r_2^{\sim 2}(3) / (1-\beta)$

Rank composition

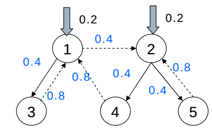


$r_i^k(j) = r_i(k) r_k^{\sim k}(j) / (1-\beta)$

Hubs

- Instead of a single page k, we can use a set H of “hub” pages
  - Define $r_i^{\sim H}(j)$ as set of tours from i to j that do not include any node from H as interior nodes or end node

Hubs example



H = {1,2}
$\beta = 0.8$

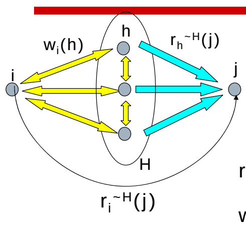
$r_2^{\sim H}$

Node	Iteration 0	1	stable
1	0	0	0
2	0.2	0.2	0.2
3	0	0	0
4	0	0.08	0.08
5	0	0.08	0.08

$r_1^{\sim H}$

Node	Iteration 0	1	stable
1	0.2	0	0.2
2	0	0	0
3	0	0.08	0.08
4	0	0	0
5	0	0	0

Rank composition with hubs



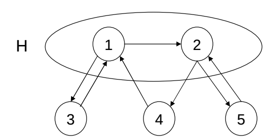
$r_i(j) = r_i^{\sim H}(j) + r_i^H(j)$

$r_i^H(j) = \Sigma_{h \in H} w_i(h) r_h^{\sim H}(j)/(1-\beta)$

$w_i(h) = r_i(h)$ if $i \neq h$

$w_i(h) = r_i(h) - (1-\beta)$ if $i = h$

Hubs rule example



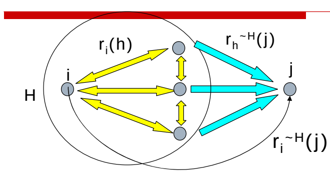
H = {1,2}
$\beta = 0.8$

$r_2(3) = r_2^{\sim H}(3) + r_2^H(3) = 0 + r_2^H(3)$
$= [r_2(1) r_1^{\sim H}(3)]/0.2 + [(r_2(2)-0.2) r_2^{\sim H}(3)]/0.2$
$= [0.192 * 0.08]/0.2 + [(0.407-0.2)*0]/0.2$
$= 0.077$

Hubs

- Start with H = T, the teleport universe
- Add nodes to H such that given any pair of nodes i and j, there is a high probability that H separates i and j
  - i.e., $r_i^{\sim H}(j)$ is zero for most i,j pairs
- Observation: high page rank nodes are good separators and hence good hub nodes

Hubs skeleton



- To compute $r_i(j)$ we need:
  - $r_i^{\sim H}(j)$ for all $i \in H$, $j \in V$
    - called the partial vector
    - Sparse
  - $r_i(h)$ for all $h \in H$
    - called the hubs skeleton

Storage reduction

- Say |T| = 1000, |H|=2000, N = 1 billion
- Store all basis vectors
  - 1000*1 billion = 1 trillion nonzero values
- Use partial vectors and hubs skeleton
  - Suppose each partial vector has N/200 nonzero entries
  - Partial vectors = 2000*N/200 = 10 billion nonzero values
  - Hubs skeleton = 2000*2000 = 4 million values
  - Total = approx 10 billion nonzero values
- Approximately 100x compression