Social and Economic Networks

Web search

Xiang Sun

2019 Fall

Outline

Reference

- Easley and Kleinberg, Chapter 14
- MIT Open Course Networks, Fall 2009, Lecture 7

Web search

- When you go to Google and type "MIT", the first result it displays is `www.mit.edu`, the home page of MIT University.
- * How does Google know that this is the best answer?
- This is a problem of information retrieval (信息检索).

Information retrieval

- Since 1960's, automated information retrieval systems were designed to search data repositories in response to keyword queries.
- Classical approach has been based on "textual analysis", i.e., look at each page separately without regard to the link structure.
- Problems:
  - a list of keywords is short and inexpressive.

  $\Rightarrow$ it is a very limited way to express a complex information need.
  - synonymy (同义词) and polysemy (多义词).
- Information retrieval requires well-trained people and well-written documents with a controlled style and vocabulary.

The problem of web search

1. It is much harder to rank web pages/documents according to a common criterion.
2. There is a correspondingly rich diversity in the set of people issuing queries, and the problem of multiple meanings becomes particularly severe.
3. Web has shifted much of the information retrieval question from a problem of scarcity to a problem of abundance.

The problem of web search (Cont.)

4. The dynamic and constantly-changing nature of Web content is another problem.
   - On September 11, 2001, many people ran to Google and typed "World Trade Center." But there was a mismatch between what people thought they could get from Google and what they really got.
   - Google at the time was built on a model in which it periodically collected Web pages and indexed them, the results were all based on pages that were gathered days or weeks earlier, and so the top results were all descriptive pages about the building itself, not about what had occurred that morning.
   - In response, Google and the other main search engines built specialized "News Search" features, which collect articles more or less continuously from a relatively fixed number of news sources.

Perspective

- In response to the one-word query "MIT," what are the clues that suggest MIT's home page is a good answer?
- There is not really any way to use features purely internal to the page www.mit.edu to solve this problem:
  - it does not use the word "MIT" more frequently or more prominently than thousands of other pages.
  - there is nothing on the page itself that makes it stand out.
- Rather, it stands out because of features on other Web pages: when a page is relevant to the query "MIT," very often www.mit.edu is among the pages it links to.

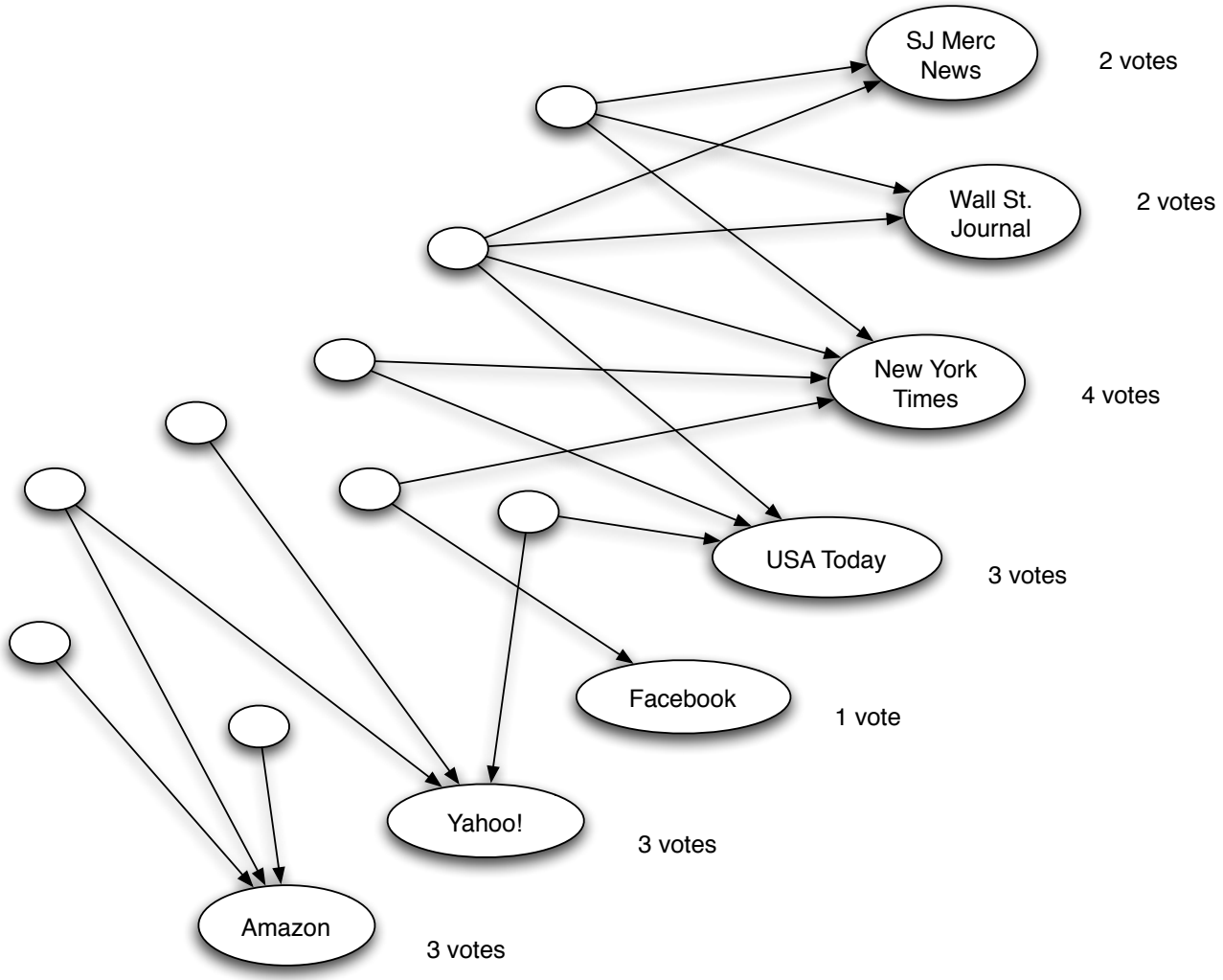
In-links

- Links are essential to ranking:
  - We can use them to assess the authority of a page on a topic, through the implicit endorsements that other pages on the topic confer through their links to it.
- In the case of the query "MIT,"
  - We could first collect a large sample of pages that are relevant to the query—as determined by a classical, text-only, information retrieval approach.
  - We could then let pages in this sample "vote" through their links: which page receives the greatest number of in-links from pages that are relevant to MIT?
  - Even this simple measure of link-counting works quite well for queries such as "MIT," where there is a single page that most people agree should be ranked first.

Voting by in-links

- Consider the one-word query "newspapers."
- Unlike the query "MIT," there is not necessarily a single, intuitively "best" answer here; there are a number of prominent newspapers on the Web, and an ideal answer would consist of a list of the most prominent among them.
- If you try this experiment, you get high scores for a mix of prominent newspapers (i.e. the results you'd want) along with pages that are going to receive a lot of in-links no matter what the query is—pages like Yahoo, Facebook, Amazon, and others.

Voting by in-links (Cont.)



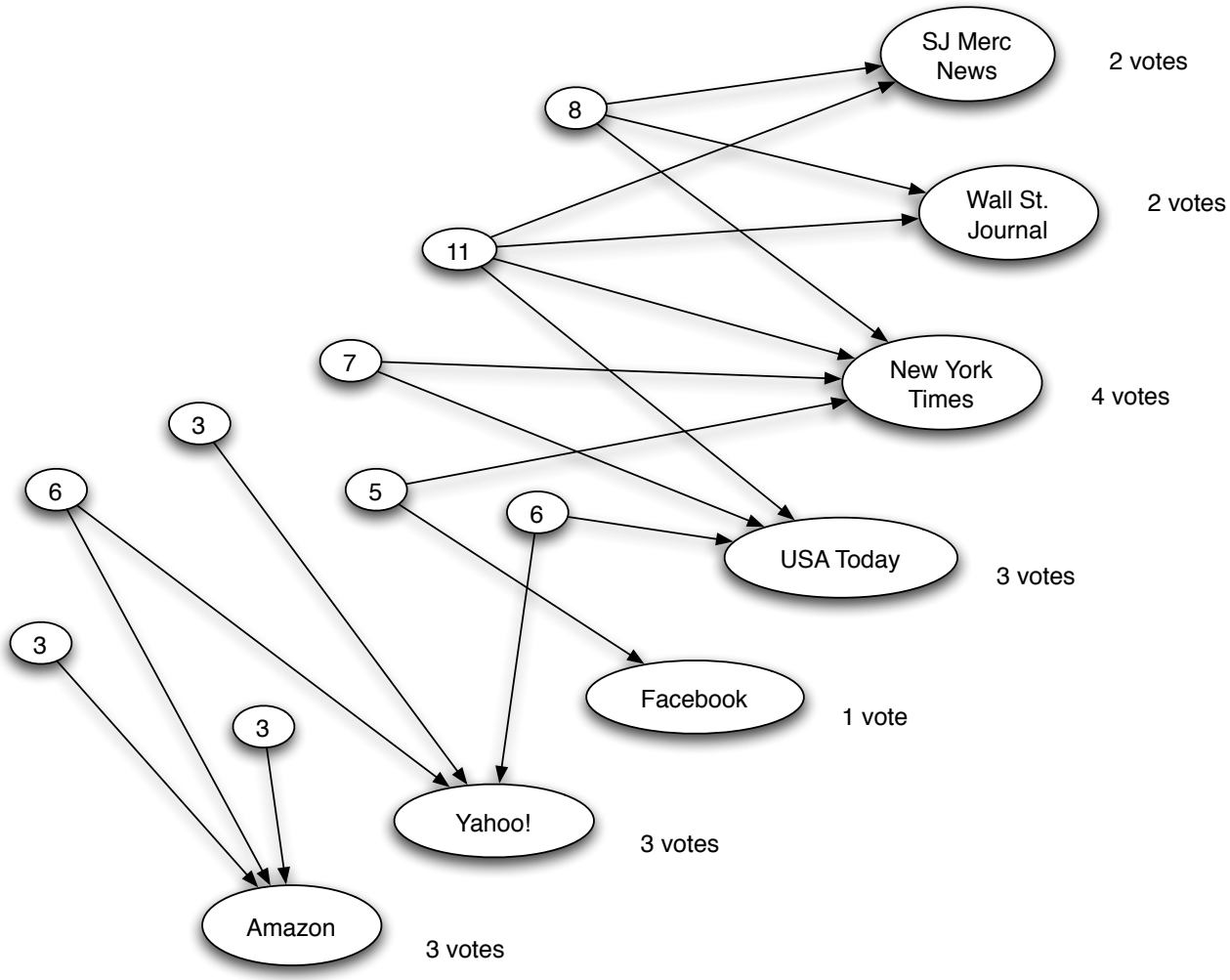
Voting by in-links (Cont.)

- This example is designed to be small enough to try by hand.
- The unlabeled circles represent our sample of pages relevant to the query "newspapers."
- Among the four pages receiving the most votes from them, two are newspapers (New York Times and USA Today) and two are not (Yahoo and Amazon).

Page's value as a list

- In addition to the newspapers themselves, there is another kind of useful answer to our query: pages that compile lists of resources relevant to the topic.
- If we could find good list pages for newspapers, we would have another approach to the problem of finding the newspapers themselves.
- Among the pages casting votes, a few of them in fact voted for many of the pages that received a lot of votes.
- * It would be natural, therefore, to suspect that these pages have some sense where the good answers are, and to score them highly as lists.
- Concretely, we could say that a page's value as a list is equal to the sum of the votes received by all pages that it voted for.

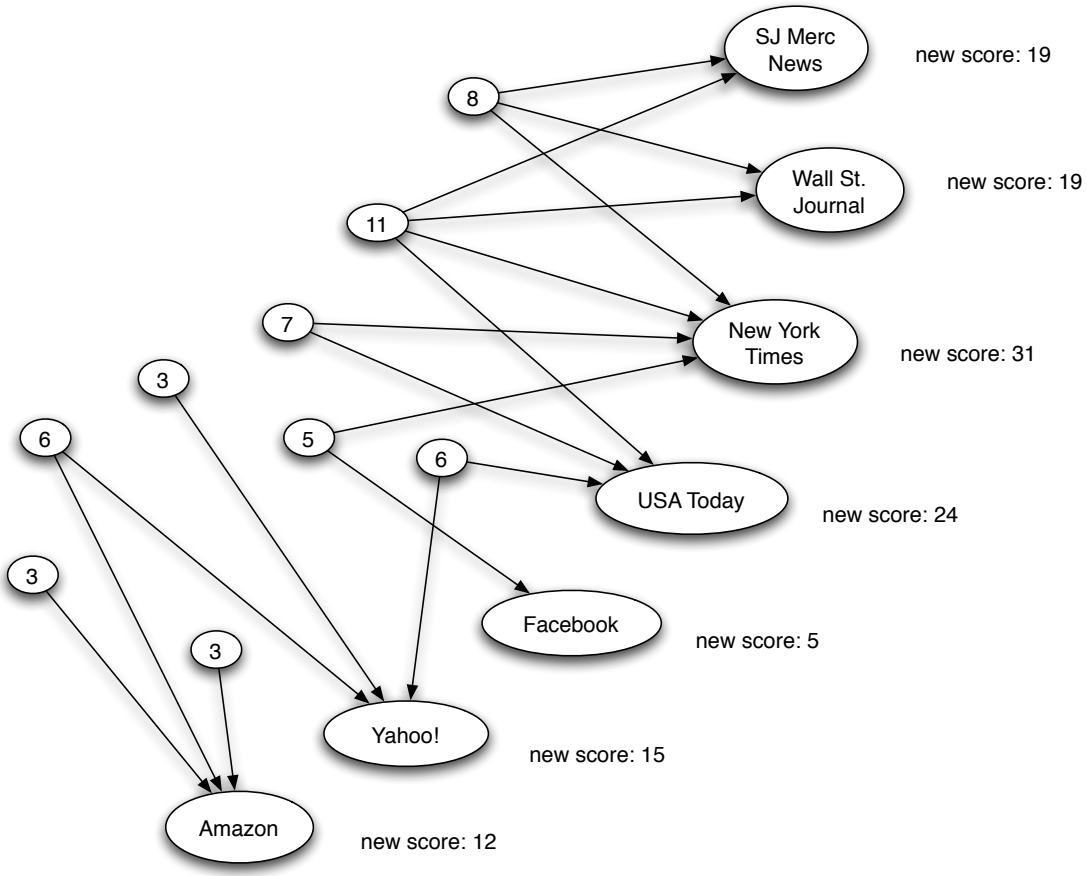
Page's value as a list (Cont.)



Improvement

- If we believe that pages scoring well as lists actually have a better sense for where the good results are, then we should weight their votes more heavily.
- We could tabulate the votes again, but this time giving each page's vote a weight equal to its value as a list.

Improvement (Cont.)



The other newspapers have surpassed the initially high-scoring Yahoo and Amazon, because these other newspapers were endorsed by pages that were estimated to be good lists.

Repeated improvement

- If we have better votes on the right-hand-side of the figure, we can use these to get still more refined values for the quality of the lists on the left-hand-side of the figure.
- With more refined estimates for the high-value lists, we can re-weight the votes that we apply to the right-hand-side once again.
- The process can go back and forth forever: it can be viewed as a Principle of Repeated Improvement, in which each refinement to one side of the figure enables a further refinement to the other.

Hubs and authorities

- The example suggests a ranking procedure which is defined in terms of two kinds of nodes:
  - Authorities (权威): The prominent highly endorsed answers to the queries (nodes that are pointed to by highly ranked nodes).
  - Hubs (中枢): High-value lists (nodes that point to highly ranked nodes).
- For each page p, we are trying to estimate its value as a potential authority and as a potential hub, so we assign it to numerical values $a(p)$ (for authority weight) and $h(p)$ (for hub weight).

Updates

- Let A denote the $n \times n$ adjacency matrix, i.e., $A_{ij} = 1$ if there is a link from node i to node j.
- The authority weights satisfy (authority updates):

$$a(j) = \sum_i A_{ij} \cdot h(i).$$

- The hub weights satisfy (hub updates):

$$h(i) = \sum_j A_{ij} \cdot a(j).$$

- This can be written in matrix-vector notation as:

$$a = A^\intercal h, \quad h = Aa.$$

Algorithm

1. We start with all hub scores and all authority scores equal to 1.
2. We choose a number of steps k.
3. We then perform a sequence of k hub-authority updates. Each update works as follows:
   - First apply the Authority Update Rule to the current set of scores.
   - Then apply the Hub Update Rule to the resulting set of scores.
4. Normalization: We divide down each authority score by the sum of all authority scores, and divide down each hub score by the sum of all hub scores.

What happens when we perform the k-step hub-authority computation for some large value of k?

k-step hub-authority computation

- We start with initial vectors of authority and hub scores that we denote $a^{[0]}$ and $h^{[0]}$, each of them equal to the vector all of whose coordinates are 1.
- Let $a^{[k]}$ and $h^{[k]}$ denote the vectors of authority and hub scores after k applications of updating.
- Then we have 1-step hub-authority computation:

$$a^{[1]} = A^\intercal h^{[0]}, \quad h^{[1]} = Aa^{[1]} = AA^\intercal h^{[0]}.$$

- In the second step, we therefore get

$$a^{[2]} = A^\intercal h^{[1]} = A^\intercal AA^\intercal h^{[0]}, \quad h^{[2]} = Aa^{[2]} = AA^\intercal AA^\intercal h^{[0]}.$$

- In the last step (step k),

$$a^{[k]} = (A^\intercal A)^{k-1} A^\intercal h^{[0]}, \quad h^{[k]} = (AA^\intercal)^k h^{[0]}.$$

Normalization

- Since the actual magnitude of the hub and authority values tend to grow with each update, they will only converge when we take normalization into account.
- Concretely, what we will show is that there are constants c and d so that the sequences of vectors $\frac{h^{[k]}}{c^k}$ and $\frac{a^{[k]}}{d^k}$ converge to limits as k goes to infinity.
- Let

$$\frac{h^{[k]}}{c^k} = \frac{(AA^\intercal)^k h^{[0]}}{c^k} \to h^*.$$

- Then

$$h^* = \lim_{k\to\infty} \frac{(AA^\intercal)^{k+1} h^{[0]}}{c^{k+1}} = \frac{AA^\intercal}{c} \lim_{k\to\infty} \frac{(AA^\intercal)^k h^{[0]}}{c^k} = \frac{AA^\intercal}{c} h^*.$$

- So we expect that h^* should be an eigenvector of the matrix $AA^\intercal$, with c a corresponding eigenvalue.

Symmetric matrix

- $AA^\intercal$ is a symmetric matrix.

$\Rightarrow$ All its eigenvalues are real and it has a set of n eigenvectors that are all unit vectors and all mutually orthogonal.

- We write the resulting eigenvectors as $z_1, z_2, \ldots, z_n$, with corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ respectively; and let's order the eigenvalues so that $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$.

* For simplification, let's suppose that $|\lambda_1| > |\lambda_2|$.

Convergence

- Let $h^{[0]} = q_1 z_1 + q_2 z_2 + \cdots + q_n z_n$.
- Then

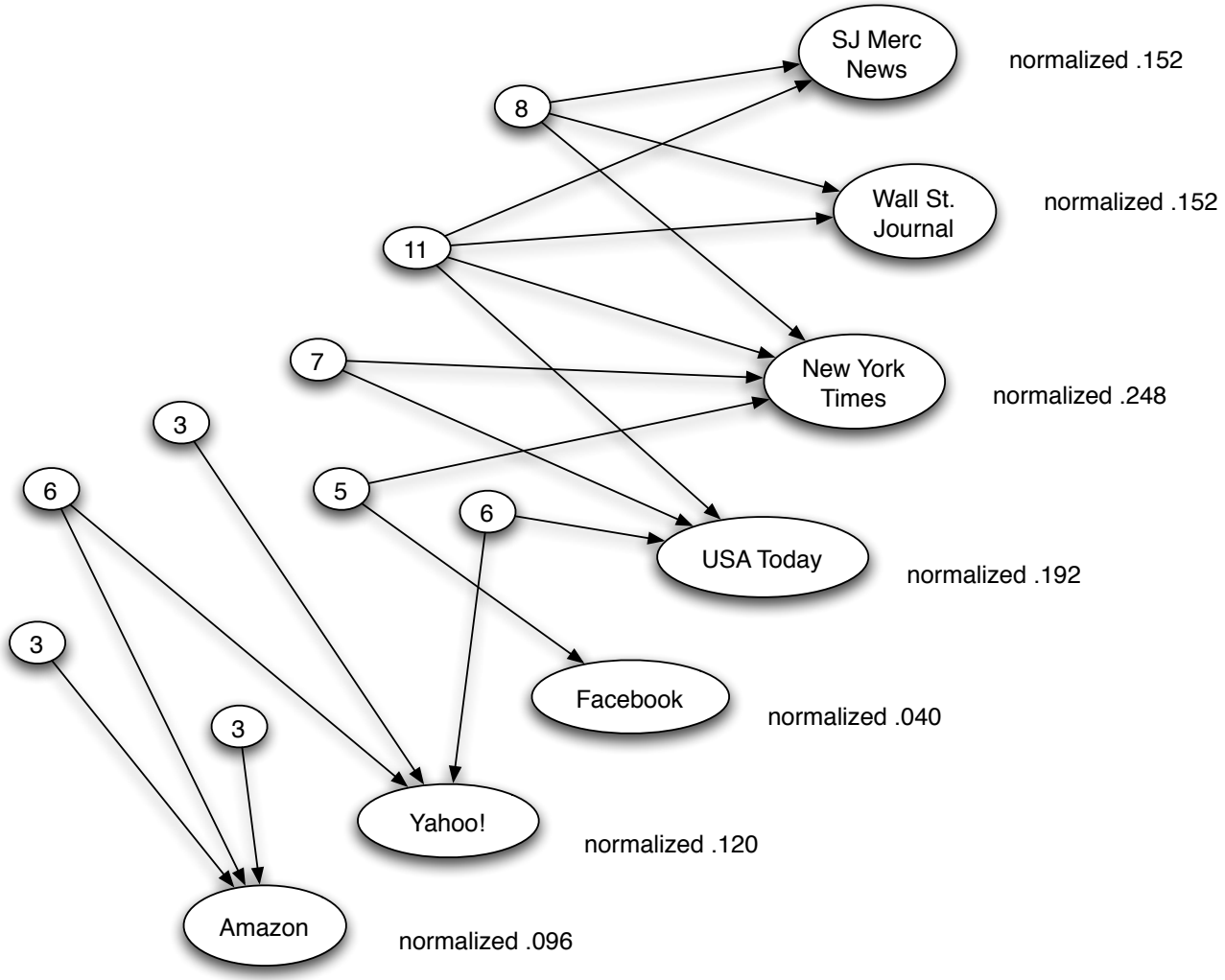
$$\begin{aligned} h^{[k]} &= (AA^\intercal)^k h^{[0]} = (AA^\intercal)^k (q_1 z_1 + q_2 z_2 + \cdots + q_n z_n) \\ &= \lambda_1^k q_1 z_1 + \lambda_2^k q_2 z_2 + \cdots + \lambda_n^k q_n z_n. \end{aligned}$$

- Divide both sides by λ_1^k,

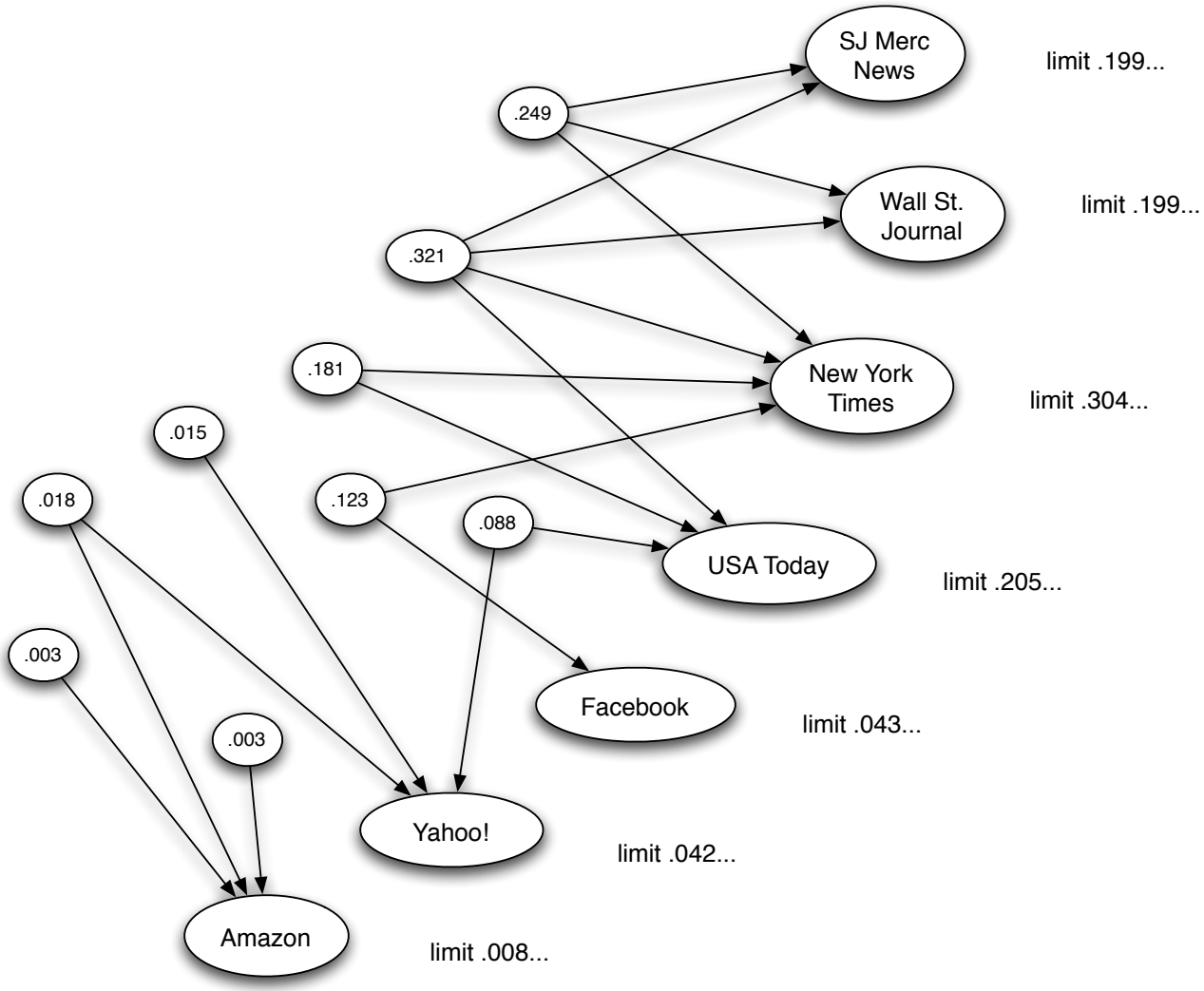
$$\frac{h^{[k]}}{\lambda_1^k} = q_1 z_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^k q_2 z_2 + \cdots + \left(\frac{\lambda_n}{\lambda_1}\right)^k q_n z_n.$$

- Recalling our assumption that $|\lambda_1| > |\lambda_2|$, as k goes to infinity, every term on RHS but the first is going to 0.
- As a result, $\lim_{k \to \infty} \frac{h^{[k]}}{\lambda_1^k} = q_1 z_1$.

Normalization: Illustration



Limit: Illustration



Remark

- We will find that in fact a limit in the direction of z_1 is reached essentially regardless of our choice of starting hub scores $h^{[0]}$.
  - The limiting hub weights are really a function of the network structure, not the starting estimates.
- For $h^{[0]} = (1, 1, \ldots, 1)^\intercal$, the coefficient q_1 is not zero, so as to be able to ensure so that the limit $q_1 z_1$ is in fact a non-zero vector in the direction of z_1.

Remark (Cont.)

- When $|\lambda_1| = |\lambda_2|$, we still have convergence, but the limit to which the sequence converges might now depend on the choice of the initial vector $h^{[0]}$.
  - We should emphasize, though, that in practice, with real and sufficiently large hyperlink structures, one essentially always gets a matrix A with the property that $AA^\intercal$ has $|\lambda_1| > |\lambda_2|$.
- Implementation of this algorithm requires "global knowledge," therefore it is implemented in a "query-dependent manner."

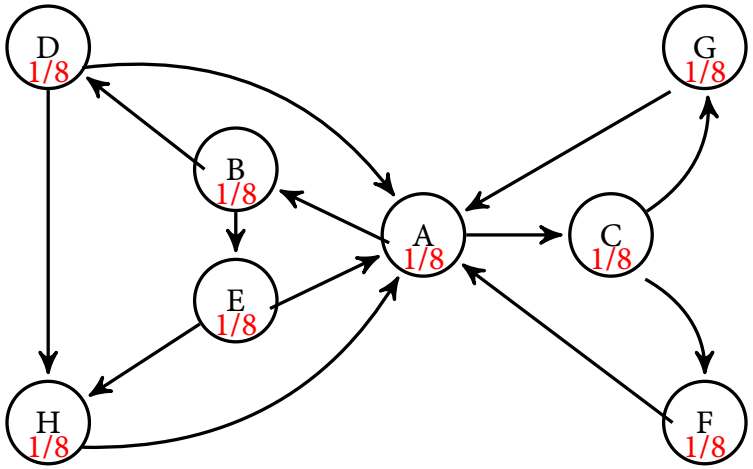
- The intuition behind hubs and authorities is based on the idea that pages play multiple roles in the network.
  - In particular, pages can play a powerful endorsement role without themselves being heavily endorsed.
- In other settings on the Web, however, endorsement is best viewed as passing directly from one prominent page to another.
  - In other words, a page is important if it is cited by other important pages.

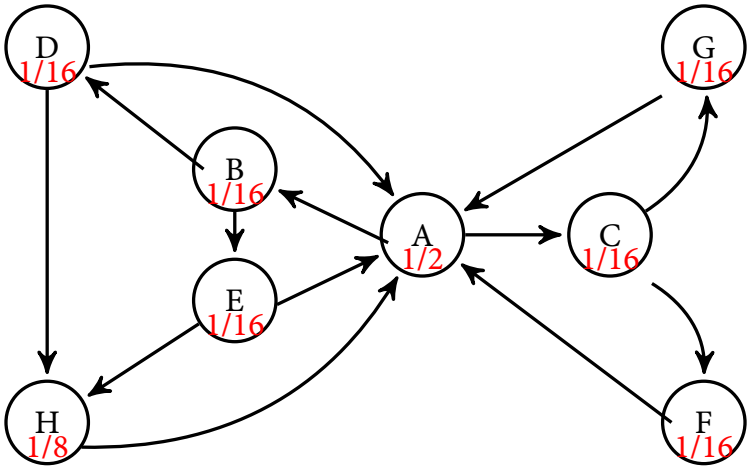
PageRank

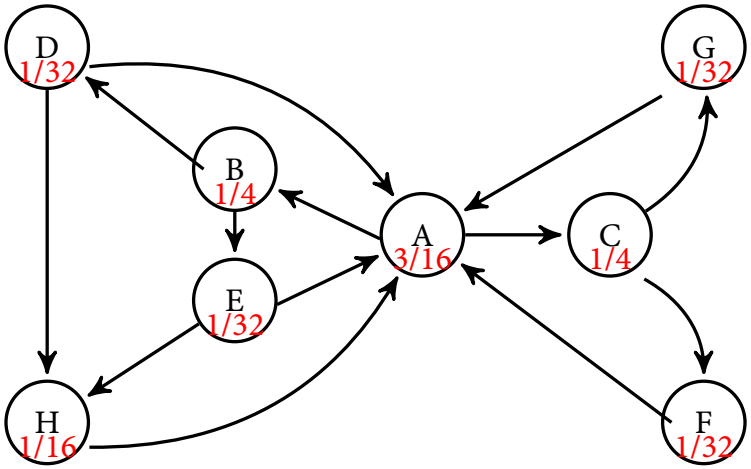
PageRank is computed as follows.

1. In a network with n nodes, we assign all nodes the same initial PageRank, set to be $\frac{1}{n}$.
2. We choose a number of steps k.
3. We then perform a sequence of k updates to the PageRank values, using the following rule for each update:
   - Each page divides its current PageRank equally across its out-going links, and passes these equal shares to the pages it points to. (If a page has no out-going links, it passes all its current PageRank to itself.)
   - Each page updates its new PageRank to be the sum of the shares it receives.

PageRank: Illustration







PageRank

- Each node j has a single weight (PageRank value) $r(j)$ which is a function of the weights of its (incoming) neighbors:

$$r(j) = \sum_i \frac{r(i)}{d_{\text{out}}(i)} A_{ij} = \sum_i \frac{A_{ij}}{d_{\text{out}}(i)} r(i),$$

  where $d_{\text{out}}(i)$ is the out-degree of node i.
- We can express this in vector-matrix notation as:

$$r = \tilde{A}^\intercal r,$$

  where $\tilde{A}_{ij} = \frac{A_{ij}}{d_{\text{out}}(i)}$.
- Note that $\sum_j \tilde{A}_{ij} = 1$, i.e., $\sum_i \tilde{A}^\intercal_{ij} = 1$.

PageRank (Cont.)

- Starting from an initial PageRank vector $r^{[0]}$, we produce a sequence of vectors $r^{[1]}$, $r^{[2]}$, ..., where each is obtained from the previous via multiplication by $\tilde{A}^\intercal$.

$$r^{[k]} = (\tilde{A}^\intercal)^k r^{[0]}.$$

- Since PageRank is conserved as it is updated (the sum of the PageRanks at all nodes remains constant through the application of the scaled update rule), we don't have to worry about normalizing these vectors as we proceed.

PageRank (Cont.)

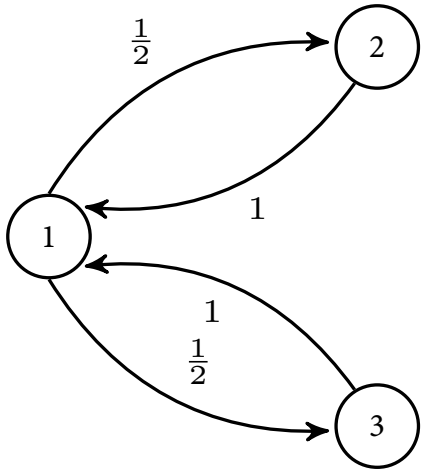
- Similarly as before, one expects that if the PageRank Update Rule converges to a limiting vector r^*, this limit should satisfy

$$\tilde{A}^\intercal r^* = r^*,$$

  that is, we should expect r^* to be an eigenvector of $\tilde{A}^\intercal$ with corresponding eigenvalue 1.
- Note that $\sum_i \tilde{A}^\intercal_{ij} = 1$ (each column summing to 1).
- Perron's Theorem: If a matrix is column-stochastic, then 1 is an eigenvalue and $1 \geq |\lambda'|$ for all other eigenvalues λ'.

PageRank: Remark

- The corresponding eigenvector could be complex.
- We only knows the eigenvector corresponding to largest eigenvalue may be a good answer. However, the convergence process could be problematic.

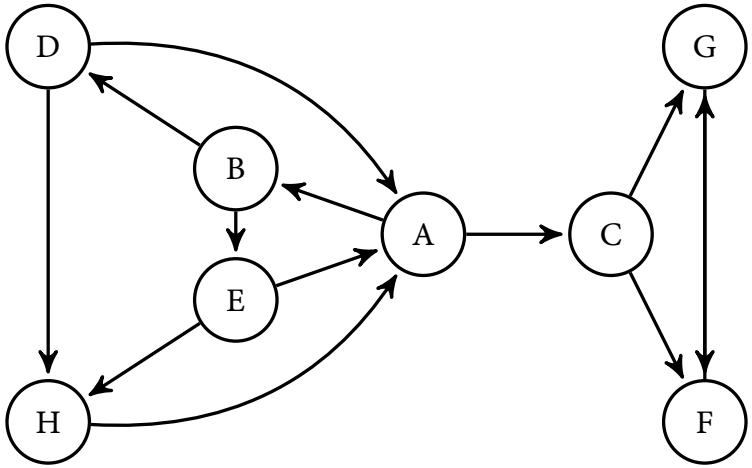


$$\Longleftrightarrow \tilde{A}^{\intercal} = \begin{bmatrix} 0 & 1 & 1 \\ \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & 0 \end{bmatrix}$$

$$r^{[0]} = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix} \to \begin{pmatrix} 2/3 \\ 1/6 \\ 1/6 \end{pmatrix} \to \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix} \to \begin{pmatrix} 2/3 \\ 1/6 \\ 1/6 \end{pmatrix} \to \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix} \to \cdots$$

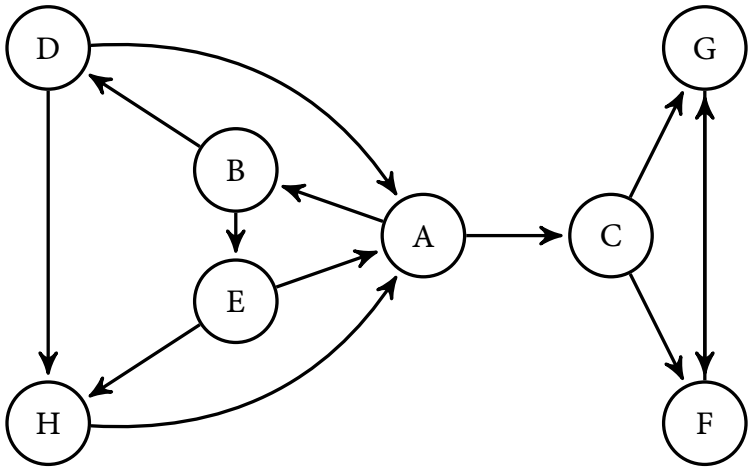
Dangling ends

- There is a difficulty with the basic definition of PageRank, however: in many networks, the "wrong" nodes can end up with all the PageRank—dangling ends.
- PageRank that flows from C to F and G can never circulate back into the rest of the network.
- The links out of C function as a kind of "slow leak" that eventually causes all the PageRank to end up at F and G.



Dangling ends (Cont.)

- We can indeed check that by repeatedly running the Basic PageRank Update Rule, we converge to PageRank values of $\frac{1}{2}$ for each of F and G, and 0 for all other nodes.



Scaled PageRank: Intuition

- Why all the water on earth doesn't inexorably run downhill and reside exclusively at the lowest points?
- It's because there's a counter-balancing process at work: water also evaporates and gets rained back down at higher elevations.

Scaled PageRank

- We pick a scaling factor s that should be strictly between 0 and 1. We then replace the Basic PageRank Update Rule with the following:
- Scaled PageRank Update Rule:
  - First apply the Basic PageRank Update Rule.
  - Then scale down all PageRank values by a factor of s. This means that the total PageRank in the network has shrunk from 1 to s.
  - We divide the residual $(1 - s)$ units of PageRank equally over all nodes, giving $\frac{1-s}{n}$ to each.
- This rule also preserves the total PageRank in the network, since it is just based on redistribution according to a different "water cycle" that evaporates $1 - s$ units of PageRank in each step and rains it down uniformly across all nodes.

Scaled PageRank (Cont.)

- Each node j has a single weight (PageRank value) $r(j)$ which is a function of the weights of its (incoming) neighbors:

$$r(j) = s \sum_i \frac{r(i)}{d_{\text{out}}(i)} A_{ij} + \frac{1-s}{n} = \sum_i \tilde{A}_{ij}^{\intercal} r(i),$$

  where $\tilde{A}_{ij} = s \frac{A_{ij}}{d_{\text{out}}(i)} + \frac{1-s}{n}$.
- We can express this in vector-matrix notation as:

$$r = \tilde{A}^{\intercal} r.$$

Scaled PageRank (Cont.)

- Starting from an initial PageRank vector $r^{[0]}$, we produce a sequence of vectors $r^{[1]}$, $r^{[2]}$, …, where each is obtained from the previous via multiplication by $\tilde{A}^\intercal$.

$$r^{[k]} = (\tilde{A}^\intercal)^k r^{[0]}.$$

- Similarly as before, one expects that if the PageRank Update Rule converges to a limiting vector r^*, this limit should satisfy

$$\tilde{A}^\intercal r^* = r^*,$$

that is, we should expect r^* to be an eigenvector of $\tilde{A}^\intercal$ with corresponding eigenvalue 1.

- $\tilde{A}^\intercal$ is column-stochastic.

$\Rightarrow$ 1 is an eigenvalue and $1 \geq |\lambda'|$ for all other eigenvalues λ'.

Perron's Theorem

Perron's Theorem

If a matrix M is positive, then

- It has a real eigenvalue λ such that $\lambda > |\lambda'|$ for all other eigenvalues λ'.
- There is an eigenvector y with positive real coordinates corresponding to the largest eigenvalue λ, and y is unique up to multiplication by a constant.
- If the largest eigenvalue λ is equal to 1, then for any starting vector $x \neq 0$ with nonnegative coordinates, the sequence of vectors $M^k x$ converges to a vector in the direction of y as k goes to infinity.

Convergence

- Perron's Theorem tells us that
  - there is a unique vector y that remains fixed under the application of the scaled update rule,
  - and that repeated application of the update rule from any starting point will converge to y.
- This vector y thus corresponds to the limiting PageRank values we have been seeking.

Remark

- This is the version of PageRank that is used in practice, with a scaling factor s that is usually chosen to be between 0.8 and 0.9.
- The use of the scaling factor also turns out to make the PageRank measure less sensitive to the addition or deletion of small numbers of nodes or links.

Random walks

- Consider someone who is randomly browsing a network of Web pages. (random walk)
- They start by choosing a page at random, picking each page with equal probability.
- They then follow links for a sequence of k steps: in each step, they pick a random out-going link from their current page, and follow it to where it leads.

  If their current page has no out-going links, they just stay where they are.
- This is not meant to be an accurate model of an actual person exploring the Web; rather, it is a thought experiment that leads to a particular definition.

Equivalence

Result

The probability of being at a page X after k steps of this random walk is precisely the PageRank of X after k applications of the Basic PageRank Update Rule.

- The PageRank of a page X is the limiting probability that a random walk across hyperlinks will end up at X, as we run the walk for larger and larger numbers of steps.

Equivalence: Remark

- This equivalent definition using random walks also provides a new and sometimes useful perspective.
- * For example, the "leakage" of PageRank to nodes F and G has a natural interpretation in terms of the random walk on the network:
  - in the limit, as the walk runs for more and more steps, the probability of the walk reaching F or G is converging to 1;
  - and once it reaches either F or G, it is stuck at these two nodes forever.
  - Thus, the limiting probabilities of being at F and G are converging to $\frac{1}{2}$ each, and the limiting probabilities are converging to 0 for all other nodes.

Proof

If $b_1^{[k]}, b_2^{[k]}, \ldots, b_n^{[k]}$ denote the probabilities of the walk being at nodes 1, 2, …, n respectively in step k, what is the probability it will be at node j in the next step?

- For each node i that links to j, if we are given that the walk is currently at node i, then there is a $\frac{1}{d_{\text{out}}(i)}$ chance that it moves from i to j in the next step.
- The walk has to actually be at node i for this to happen, so node i contributes $b_i^{[k]} \cdot \frac{1}{d_{\text{out}}(i)}$ to the probability of being at j in the next step.
- Therefore, summing $\frac{b_i^{[k]}}{d_{\text{out}}(i)}$ over all nodes i that link to j gives the probability the walk is at $b_j^{[k+1]}$ in the next step.

$$b_j^{[k+1]} = \sum_i A_{ij} \frac{b_i^{[k]}}{d_{\text{out}}(i)}.$$

Proof (Cont.)

- We can express this in vector-matrix notation as:

$$b^{[k+1]} = \tilde{A}^{\top} b^{[k]},$$

  where $\tilde{A}_{ij} = \frac{A_{ij}}{d_{\text{out}}(i)}$.
- Since both PageRank values and random-walk probabilities start out the same (they are initially $\frac{1}{n}$ for all nodes), and they then evolve according to exactly the same rule, they remain the same forever.

Scaled random walks

- With probability s, the walk follows a random edge as before; and with probability $1 - s$ it jumps to a node chosen uniformly at random.
- Then

$$b_j^{[k+1]} = s \sum_i \frac{b_i^{[k]}}{d_{\text{out}}(i)} A_{ij} + \frac{1-s}{n} = \sum_i \tilde{A}_{ij}^{\intercal} b_i^{[k]},$$

- This is the same as the update rule for the scaled PageRank values.
- The random-walk probabilities and the scaled PageRank values start at the same initial values, and then evolve according to the same update, so they remain the same forever.

Modern web search

- One can integrate information from both network structure and textual content in order to produce the highest-quality search results.
  - the analysis of anchor text.
- In addition to text and links, search engines use many other features as well.
  - users' "feedback".
- People increasingly began modifying their Web-page authoring styles to score highly in search engine rankings.
  - for search engines, the "perfect" ranking function will always be a moving target.
  - search engines are incredibly secretive about the internals of their ranking functions—not just to prevent competing search engines from finding out what they're doing, but also to prevent designers of Web sites from finding out.