Online Social Networks and Media

Link Analysis and Web Search

How to Organize the Web

First try: Human curated Web directories Yahoo, DMOZ, LookSmart



Help us improve the Yahoo! Directory - Share your ideas

Baseball, Soccer, Basketball..

<u>World</u>

Shopping

Clothing, Food, Gifts ...

Català, Dansk, Deutsch, Español, Français, Italiano, 日本語, Nederlands, Polski, Русский, Svenska,...

People, Religion, Issues...

Become an Editor Help build the largest human-edited directory of the web

Society



Copyright © 2012 Netscape

How to organize the web

- Second try: Web Search
 - Information Retrieval investigates:
 - Find relevant docs in a small and trusted set e.g., Newspaper articles, Patents, etc. ("needle-in-ahaystack")
 - Limitation of keywords (synonyms, polysemy, etc)
 - But: Web is huge, full of untrusted documents, random things, web spam, etc.
 - Everyone can create a web page of high production value
 - Rich diversity of people issuing queries
 - Dynamic and constantly-changing nature of web content

Size of the Search Index



YAHOO! SEARCH

The size of the World Wide Web: Estimated size of Yahoo Search index



bing

Last Month Last Three Months Last Year Last Two Years Size Bing (Number of webpages) 20 18 16 Size in billion webpages 14 12 Bing 10 8 6 30 Apr 2011 23 Sep 2011 16 Feb 2012 11 Jul 2012 04 Dec 2012 05 Dec 2010

The size of the World Wide Web: Estimated size of Bing index

How to organize the web

- Third try (the Google era): using the web graph
 - Swift from relevance to *authoritativeness*
 - It is not only important that a page is relevant, but that it is also important on the web
- For example, what kind of results would we like to get for the query "greek newspapers"?

Link Analysis

- Not all web pages are equal on the web
- The links act as endorsements:
 - When page *p* links to *q* it endorses the content of the content of *q*

What is the simplest way to measure importance of a page on the web?



Rank by Popularity

 Rank pages according to the number of incoming edges (in-degree, degree centrality)



- 1. Red Page
- 2. Yellow Page
- 3. Blue Page
- 4. Purple Page
- 5. Green Page

Popularity



- It is not important only how many link to you, but also how important are the people that link to you.
- Good authorities are pointed by good authorities
 - Recursive definition of importance

THE PAGERANK ALGORITHM

PageRank

- Good authorities should be pointed by good authorities
 - The value of a node is the value of the nodes that point to it.
- How do we implement that?
 - Assume that we have a unit of authority to distribute to all nodes.
 - Initially each node gets $\frac{1}{n}$ amount of authority
 - Each node distributes the authority value they have to their neighbors
 - The authority value of each node is the sum of the authority fractions it collects from its neighbors.

$$w_v = \sum_{u \to v} \frac{1}{d_{out}(u)} w_u$$

 w_v : the PageRank value of node v

Recursive definition

A simple example



 Solving the system of equations we get the authority values for the nodes

 $- w = \frac{1}{2} w = \frac{1}{4} w = \frac{1}{4}$

A more complex example

$$w_{1} = 1/3 w_{4} + 1/2 w_{5}$$

$$w_{2} = 1/2 w_{1} + w_{3} + 1/3 w_{4}$$

$$w_{3} = 1/2 w_{1} + 1/3 w_{4}$$

$$w_{4} = 1/2 w_{5}$$

$$w_{5} = w_{2}$$

$$w_v = \sum_{u \to v} \frac{1}{d_{out}(u)} w_u$$



Computing PageRank weights

- A simple way to compute the weights is by iteratively updating the weights
- PageRank Algorithm

Initialize all PageRank weights to $\frac{1}{n}$ Repeat: $w_v = \sum_{u \to v} \frac{1}{d_{out}(u)} w_u$ Until the weights do not change

This process converges





Initially, all nodes PageRank 1/8

Step

 $\frac{1}{2}$

А

1/2

3/16

✓ As a kind of "fluid" that circulates through the network
 ✓ The total PageRank in the network remains constant (no need to normalize)



PageRank: equilibrium

- A simple way to check whether an assignment of numbers forms an equilibrium set of PageRank values: check that they sum to 1, and that when apply the Basic PageRank Update Rule, we get the same values back.
- If the network is strongly connected, then there is a unique set of equilibrium values.

Random Walks on Graphs

- The algorithm defines a random walk on the graph
- Random walk:
 - Start from a node chosen uniformly at random with probability $\frac{1}{n}$.
 - Pick one of the outgoing edges uniformly at random
 - Move to the destination of the edge
 - Repeat.
- The Random Surfer model
 - Users wander on the web, following links.

















• Step 4...



Random walk

Question: what is the probability p^t_i of being at node *i* after *t* steps?





Markov chains

A Markov chain describes a discrete time stochastic process over a set of ۲ states

$$S = \{s_1, s_2, \dots, s_n\}$$

according to a transition probability matrix $P = \{P_{ij}\}$ - P_{ij} = probability of moving to state j when at state i

Matrix P has the property that the entries of all rows sum to 1 ۲

$$\sum_{i} P[i, j] = 1$$

A matrix with this property is called stochastic

- State probability distribution: The vector $p^t = (p_1^t, p_2^t, ..., p_n^t)$ that stores the probability of being at state S_i after t steps
- Memorylessness property: The next state of the chain depends only at the ۲ current state and not on the past of the process (first order MC)
 - Higher order MCs are also possible
- Markov Chain Theory: After infinite steps the state probability vector ٠ converges to a unique distribution if the chain is irreducible (possible to get from any state to any other state) and aperiodic

Random walks

- Random walks on graphs correspond to Markov Chains
 - The set of states S is the set of nodes of the graph
 G
 - The transition probability matrix is the probability that we follow an edge from one node to another $P[i, j] = 1/\deg_{out}(i)$

An example





Node Probability vector

- The vector $p^t = (p_i^t, p_2^t, \dots, p_n^t)$ that stores the probability of being at node v_i at step t
- *p*⁰_i = the probability of starting from state
 i (usually) set to uniform
- We can compute the vector p^t at step t using a vector-matrix multiplication

$$p^t = p^{t-1} P$$

An example

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

$$p_{1}^{t} = \frac{1}{3}p_{4}^{t-1} + \frac{1}{2}p_{5}^{t-1}$$

$$p_{2}^{t} = \frac{1}{2}p_{1}^{t-1} + p_{3}^{t-1} + \frac{1}{3}p_{4}^{t-1}$$

$$p_{3}^{t} = \frac{1}{2}p_{1}^{t-1} + \frac{1}{3}p_{4}^{t-1}$$

$$p_{4}^{t} = \frac{1}{2}p_{5}^{t-1}$$

$$p_{5}^{t} = p_{2}^{t-1}$$



Stationary distribution

- The stationary distribution of a random walk with transition matrix P, is a probability distribution π , such that $\pi = \pi P$
- The stationary distribution is an eigenvector of matrix P

 the principal left eigenvector of P stochastic matrices have maximum eigenvalue 1
- The probability π_i is the fraction of times that we visited state i as $t \to \infty^i$
- Markov Chain Theory: The random walk converges to a unique stationary distribution independent of the initial vector if the graph is strongly connected, and not bipartite.

Computing the stationary distribution

• The Power Method

Initialize q^0 to some distribution Repeat $q^t = q^{t-1}P$ Until convergence

- After many iterations $q^t \rightarrow \pi$ regardless of the initial vector q^0
- Power method because it computes $q^t = q^0 P^t$
- Rate of convergence

- determined by the second eigenvalue $\frac{|\lambda_2|}{|\lambda_1|}$

The stationary distribution

- What is the meaning of the stationary distribution π of a random walk?
- π(i): the probability of being at node i after very large (infinite) number of steps
- $\pi = p_0 P^{\infty}$, where *P* is the transition matrix, p_0 the original vector
 - -P(i, j): probability of going from i to j in one step
 - $-P^{2}(i, j)$: probability of going from i to j in two steps (probability of all paths of length 2)

 $-P^{\infty}(i,j) = \pi(j)$: probability of going from i to j in infinite steps – starting point does not matter.

The PageRank random walk

- Vanilla random walk
 - make the adjacency matrix stochastic and run a random walk

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



The PageRank random walk

- What about sink nodes?
 - what happens when the random walk moves to a node without any outgoing inks?

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


The PageRank random walk

- Replace these row vectors with a vector v
 - typically, the uniform vector

Ρ

$$P' = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$

$$Y = P + dv^{T} \qquad d = \begin{cases} 1 & \text{if i is sink} \\ 0 & \text{otherwise} \end{cases}$$

The PageRank random walk



The PageRank random walk

- Add a random jump to vector v with prob 1-α
 typically, to a uniform vector
- Restarts after 1/(1-α) steps in expectation
 Guarantees irreducibility, convergence

$$\mathsf{P}'' = \alpha \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 \end{bmatrix} + (1 - \alpha) \begin{bmatrix} 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{bmatrix}$$

 $P'' = \alpha P' + (1-\alpha)uv^T$, where u is the vector of all 1s

PageRank algorithm [BP98]

- The Random Surfer model
 - pick a page at random
 - with probability 1- α jump to a random page
 - with probability a follow a random outgoing link
- Rank according to the stationary distribution

$$PR(p) = \alpha \sum_{q \to p} \frac{PR(q)}{|Out(q)|} + (1 - \alpha) \frac{1}{n}$$

 $\alpha = 0.85$ in most cases



- 1. Red Page
- 2. Purple Page
- 3. Yellow Page
- 4. Blue Page
- 5. Green Page

PageRank: Example



Stationary distribution with random jump

• If v is the jump vector

$$p^{0} = v$$

$$p^{1} = \alpha p^{0} P' + (1 - \alpha)v = \alpha v P' + (1 - \alpha)v$$

$$p^{2} = \alpha p^{1} P' + (1 - \alpha)v = \alpha^{2} v P'^{2} + (1 - \alpha)v \alpha P' + (1 - \alpha)v$$

$$\vdots$$

$$p^{\infty} = (1 - \alpha)v + (1 - \alpha)v \alpha P' + (1 - \alpha)v \alpha^{2} P'^{2} + \cdots$$

$$= (1 - \alpha)(I - \alpha P')^{-1}$$

- With the random jump the shorter paths are more important, since the weight decreases exponentially
 - makes sense when thought of as a restart
- If v is not uniform, we can bias the random walk towards the nodes that are close to v
 - Personalized and Topic-Specific Pagerank.

Effects of random jump

- Guarantees convergence to unique distribution
- Motivated by the concept of random surfer
- Offers additional flexibility
 - personalization
 - anti-spam
- Controls the rate of convergence

- the second eigenvalue of matrix P'' is α

Random walks on undirected graphs

- For undirected graphs, the stationary distribution of a random walk is proportional to the degrees of the nodes
 - Thus in this case a random walk is the same as degree popularity
- This is not longer true if we do random jumps
 - Now the short paths play a greater role, and the previous distribution does not hold.

PageRank implementation

- Store the graph in adjacency list, or list of edges
- Keep current pagerank values and new pagerank values
- Go through edges and update the values of the destination nodes.
- Repeat until the difference between the pagerank vectors (L_1 or L_∞ difference) is below some small value ε .

A (Matlab-friendly) PageRank algorithm

 Performing vanilla power method is now too expensive – the matrix is not sparse



PageRank history

- Huge advantage for Google in the early days
 - It gave a way to get an idea for the value of a page, which was useful in many different ways
 - Put an order to the web.
 - After a while it became clear that the anchor text was probably more important for ranking
 - Also, link spam became a new (dark) art
- Flood of research
 - Numerical analysis got rejuvenated
 - Huge number of variations
 - Efficiency became a great issue.
 - Huge number of applications in different fields
 - Random walk is often referred to as PageRank.

THE HITS ALGORITHM

The HITS algorithm

- Another algorithm proposed around the same time as PageRank for using the hyperlinks to rank pages
 - Kleinberg: then an intern at IBM Almaden
 - IBM never made anything out of it

Root set obtained from a text-only search engine









Hubs and Authorities [K98]

- Authority is not necessarily transferred directly between authorities
- Pages have double identity
 - hub identity
 - authority identity
- Good hubs point to good authorities
- Good authorities are pointed by good hubs



Hubs and Authorities

- Two kind of weights:
 - Hub weight
 - Authority weight
- The hub weight is the sum of the authority weights of the authorities pointed to by the hub
- The authority weight is the sum of the hub weights that point to this authority.

HITS Algorithm

- Initialize all weights to 1.
- Repeat until convergence
 - O operation : hubs collect the weight of the authorities

$$h_i = \sum_{i:i \to i} a_j$$

- I operation: authorities collect the weight of the hubs

$$a_i = \sum_{j: j \to i} h_j$$

Normalize weights under some norm

HITS and eigenvectors

- The HITS algorithm is a power-method eigenvector computation
- In vector terms
 - $-a^{t} = A^{T}h^{t-1}$ and $h^{t} = Aa^{t-1}$
 - $-a^t = A^T A a^{t-1}$ and $h^t = A A^T h^{t-1}$
 - Repeated iterations will converge to the eigenvectors
- The authority weight vector *a* is the eigenvector of *A^TA* and the hub weight vector *h* is the eigenvector of *AA^T*
- The vectors *a* and *h* are called the singular vectors of the matrix A

Singular Value Decomposition

$$A = U \Sigma V^{\mathsf{T}} = \begin{bmatrix} \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_r \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & & \sigma_r \end{bmatrix} \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vdots \\ \vec{v}_r \end{bmatrix}$$

- **r** : rank of matrix A
- $\sigma_1 \ge \sigma_2 \ge \dots \ge \sigma_r$: singular values (square roots of eig-vals AA^T, A^TA)
- $\vec{u}_1, \vec{u}_2, \dots, \vec{u}_r$: left singular vectors (eig-vectors of AA^T)
- $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_r$: right singular vectors (eig-vectors of $A^T A$) $A = \sigma_1 \vec{u}_1 \vec{v}_1^T + \sigma_2 \vec{u}_2 \vec{v}_2^T + \dots + \sigma_r \vec{u}_r \vec{v}_r^T$

Why does the Power Method work?

- If a matrix R is real and symmetric, it has real eigenvalues and eigenvectors: $(\lambda_1, w_1), (\lambda_2, w_2), \dots, (\lambda_r, w_r)$
 - r is the rank of the matrix

 $- \ |\lambda_1| \ge |\lambda_2| \ge \cdots \ge |\lambda_r|$

 For any matrix R, the eigenvectors w₁, w₂, ..., w_r of R define a basis of the vector space

- For any vector x, $Rx = \alpha_1 w_1 + \alpha_2 w_2 + \dots + \alpha_r w_r$

- After t multiplications we have: $- R^{t}x = \lambda_{1}^{t-1}\alpha_{1}w_{1} + \lambda_{2}^{t-1}a_{2}w_{2} + \dots + \lambda_{r}^{t-1}a_{r}w_{r}$
- Normalizing (divide by λ_1^{t-1}) leaves only the term w_1 .

Initialize



Step 1: O operation



Step 1: I operation



Step 1: Normalization (Max norm)



Step 2: O step



Step 2: I step



Step 2: Normalization



Convergence



 Perform a random walk on the bipartite graph of hubs and authorities alternating between the two



- Start from an authority chosen uniformly at random
 - e.g. the red authority



- Start from an authority chosen uniformly at random
 - e.g. the red authority
- Choose one of the in-coming links uniformly at random and move to a hub
 - e.g. move to the yellow authority with probability 1/3

	→□
hubs	authorities

- Start from an authority chosen uniformly at random
 - e.g. the red authority
- Choose one of the in-coming links uniformly at random and move to a hub
 - e.g. move to the yellow authority with probability 1/3
- Choose one of the out-going links uniformly at random and move to an authority
 - e.g. move to the blue authority with probability 1/2



- Formally we have probabilities:
 - $-a_i$: probability of being at authority *i*
 - $-h_j$: probability of being at hub j
- The probability of being at authority i is computed as:

$$a_i = \sum_{j \in N_{in}(i)} \frac{1}{d_{out}(j)} h_j$$

• The probability of being at hub *j* is computed as

$$h_j = \sum_{i \in N_{out}(j)} \frac{1}{d_{in}(i)} a_i$$

• Repeated computation converges
The SALSA algorithm [LM00]

- In matrix terms
 - A_c = the matrix A where columns are normalized to sum to 1
 - A_r = the matrix A where rows are normalized to sum to 1
- The hub computation $-h = A_c a$
- The authority computation

 $-a = A_r^T h = A_r^T A_c a$

• In MC terms the transition matrix

 $- P = A_r A_c^T$

hubs

 $h_2 = 1/3 a_1 + 1/2 a_2$

 $a_1 = h_1 + 1/2 h_2 + 1/3 h_3$

ABSORBING RANDOM WALKS LABEL PROPAGATION OPINION FORMATION ON SOCIAL NETWORKS

Random walk with absorbing nodes

• What happens if we do a random walk on this graph? What is the stationary distribution?



All the probability mass on the red sink node:
 The red node is an absorbing node

Random walk with absorbing nodes

• What happens if we do a random walk on this graph? What is the stationary distribution?



- There are two absorbing nodes: the red and the blue.
- The probability mass will be divided between the two

- If there are more than one absorbing nodes in the graph a random walk that starts from a non-absorbing node will be absorbed in one of them with some probability
 - The probability of absorption gives an estimate of how close the node is to red or blue



- Computing the probability of being absorbed:
 - The absorbing nodes have probability 1 of being absorbed in themselves and zero of being absorbed in another node.
 - For the non-absorbing nodes, take the (weighted) average of the absorption probabilities of your neighbors
 - if one of the neighbors is the absorbing node, it has probability 1
 - Repeat until convergence (= very small change in probs)

$$P(Red|Pink) = \frac{2}{3}P(Red|Yellow) + \frac{1}{3}P(Red|Green)$$
$$P(Red|Green) = \frac{1}{4}P(Red|Yellow) + \frac{1}{4}$$
$$P(Red|Yellow) = \frac{2}{3}$$



- Computing the probability of being absorbed:
 - The absorbing nodes have probability 1 of being absorbed in themselves and zero of being absorbed in another node.
 - For the non-absorbing nodes, take the (weighted) average of the absorption probabilities of your neighbors
 - if one of the neighbors is the absorbing node, it has probability 1

2

2

Repeat until convergence (= very small change in probs)

$$P(Blue|Pink) = \frac{2}{3}P(Blue|Yellow) + \frac{1}{3}P(Blue|Green)$$

$$P(Blue|Green) = \frac{1}{4}P(Blue|Yellow) + \frac{1}{2}$$

$$P(Blue|Yellow) = \frac{1}{3}$$

Why do we care?

- Why do we care to compute the absorbtion probability to sink nodes?
- Given a graph (directed or undirected) we can choose to make some nodes absorbing.
 - Simply direct all edges incident on the chosen nodes towards them.
- The absorbing random walk provides a measure of proximity of non-absorbing nodes to the chosen nodes.
 - Useful for understanding proximity in graphs
 - Useful for propagation in the graph
 - E.g, on a social network some nodes have high income, some have low income, to which income class is a non-absorbing node closer?

• In this undirected graph we want to learn the proximity of nodes to the red and blue nodes



• Make the nodes absorbing



 Compute the absorbtion probabilities for red and blue

$$P(Red|Pink) = \frac{2}{3}P(Red|Yellow) + \frac{1}{3}P(Red|Green)$$

$$P(Red|Green) = \frac{1}{5}P(Red|Yellow) + \frac{1}{5}P(Red|Pink) + \frac{1}{5}$$

$$P(Red|Yellow) = \frac{1}{6}P(Red|Green) + \frac{1}{3}P(Red|Pink) + \frac{1}{3}$$

$$P(Blue|Pink) = 1 - P(Red|Pink)$$

$$P(Blue|Green) = 1 - P(Red|Green)$$

P(Blue|Yellow) = 1 - P(Red|Yellow)



Penalizing long paths

• The orange node has the same probability of reaching red and blue as the yellow one

P(Red|Orange) = P(Red|Yellow)

P(Blue|Orange) = P(Blue|Yellow)

• Intuitively though it is further away



Penalizing long paths

 Add an universal absorbing node to which each node gets absorbed with probability α.

With probability α the random walk dies

With probability $(1-\alpha)$ the random walk continues as before

The longer the path from a node to an absorbing node the more likely the random walk dies along the way, the lower the absorbtion probability

$$P(\underline{Red}|Green) = (1 - \alpha) \left(\frac{1}{5}P(\underline{Red}|\underline{Yellow}) + \frac{1}{5}P(\underline{Red}|\underline{Pink}) + \frac{1}{5}\right)$$



Propagating values

Assume that Red has a positive value and Blue a negative value

Positive/Negative class, Positive/Negative opinion

 We can compute a value for all the other nodes in the same way

This is the expected value for the node

$$V(Pink) = \frac{2}{3}V(Yellow) + \frac{1}{3}V(Green)$$
$$V(Green) = \frac{1}{5}V(Yellow) + \frac{1}{5}V(Pink) + \frac{1}{5} - \frac{2}{5}$$
$$V(Yellow) = \frac{1}{6}V(Green) + \frac{1}{3}V(Pink) + \frac{1}{3} - \frac{1}{6}$$



Electrical networks and random walks

- Our graph corresponds to an electrical network
- There is a positive voltage of +1 at the Red node, and a negative voltage -1 at the Blue node
- There are resistances on the edges inversely proportional to the weights (or conductance proportional to the weights)
- The computed values are the voltages at the nodes

$$V(Pink) = \frac{2}{3}V(Yellow) + \frac{1}{3}V(Green)$$
$$V(Green) = \frac{1}{5}V(Yellow) + \frac{1}{5}V(Pink) + \frac{1}{5} - \frac{2}{5}$$
$$V(Yellow) = \frac{1}{6}V(Green) + \frac{1}{3}V(Pink) + \frac{1}{3} - \frac{1}{6}$$



Opinion formation

- The value propagation can be used as a model of opinion formation.
- Model:
 - Opinions are values in [-1,1]
 - Every user u has an internal opinion s_u , and expressed opinion z_u .
 - The expressed opinion minimizes the personal cost of user u:

$$c(z_u) = (s_u - z_u)^2 + \sum_{v:v \text{ is a friend of } u} w_u (z_u - z_v)^2$$

- Minimize deviation from your beliefs and conflicts with the society
- If every user tries independently (selfishly) to minimize their personal cost then the best thing to do is to set z_u to the average of all opinions:

$$z_u = \frac{s_u + \sum_{v:v \text{ is a friend of } u} w_u z_u}{1 + \sum_{v:v \text{ is a friend of } u} w_u}$$

• This is the same as the value propagation we described before!

• Social network with internal opinions



s = +0.5





Transductive learning

- If we have a graph of relationships and some labels on some nodes we can propagate them to the remaining nodes
 - Make the labeled nodes to be absorbing and compute the probability for the rest of the graph
 - E.g., a social network where some people are tagged as spammers
 - E.g., the movie-actor graph where some movies are tagged as action or comedy.
- This is a form of semi-supervised learning
 - We make use of the unlabeled data, and the relationships
- It is also called transductive learning because it does not produce a model, but just labels the unlabeled data that is at hand.
 - Contrast to inductive learning that learns a model and can label any new example

Implementation details

- Implementation is in many ways similar to the PageRank implementation
 - For an edge (u, v)instead of updating the value of
 v we update the value of u.
 - The value of a node is the average of its neighbors
 - We need to check for the case that a node u is absorbing, in which case the value of the node is not updated.
 - Repeat the updates until the change in values is very small.