Λ14 Διαδικτυακά Κοινωνικά Δίκτυα και Μέσα

Link Prediction

Motivation

- Recommending *new friends* in online social networks.
- Predicting the participation of *actors* in events
- Suggesting *interactions* between the members of a company/organization that are external to the hierarchical structure of the organization itself.
- Predicting *connections* between members of terrorist organizations who have not been directly observed to work together.
- Suggesting *collaborations* between researchers based on co-authorship.
- Overcoming the data-sparsity problem *in recommender systems* using collaborative filtering

Motivation

In social networks:

- Increases user engagement
- Controls the growth of the network

Outline

- Estimating a score for each edge (seminal work of Liben-Nowell&Kleinberg)
- Classification approach
- The who to follow service at Twitter

Problem Definition

Link prediction problem: Given the links in a social network at time *t* (G_{old}), ***predict*** the edges that will be added to the network during the time interval from time *t* to a given future time *t'* (G_{new}).

- Based solely on the *topology* of the network (social proximity) (the more general problem also considers attributes of the nodes and links)

- Different from the problem of *inferring missing* (hidden) links (there is a temporal aspect)

  To save experimental effort in the laboratory or in the field

Problem Formulation (details)

Consider a social network $G = (V, E)$ where each edge $e = <u, v> \in E$ represents an interaction between u and v that took place at a particular time $t(e)$

(multiple interactions between two nodes as parallel edges with different timestamps)

For two times, $t < t'$, let $G[t, t']$ *denote* subgraph of G consisting of all edges with a timestamp between t and t'

- For four times, $t_0 < t'_0 < t_1 < t'_1$, given $G[t_0, t'_0]$, we wish to output a list of edges not in $G[t_0, t'_0]$ that are predicted to appear in $G[t_1, t'_1]$

✓ $[t_0, t'_0]$ training interval
✓ $[t_1, t'_1]$ test interval

Methods for Link Prediction (outline)

- Assign a *connection weight score(x, y)* to each pair of nodes <x, y> based on the input graph
- Produce a ranked list of decreasing order of score

- We can consider all links *incident to a specific node x*, and recommend to *x* the top ones
- If we focus to a specific *x*, the score can be seen as a centrality measure for x

Methods for Link Prediction (outline)

How to assign the score(x, y) between two nodes x and y?

- ✓ Some form of **similarity** or **node proximity**

Methods for Link Prediction:
Neighborhood-based

The larger the *overlap of the neighbors* of two nodes, the more likely the nodes to be linked in the future

Methods for Link Prediction:
Neighborhood-based

Let $\Gamma(x)$ denote the set of neighbors of x in G_{old}

Common neighbors:

$$\text{score}(x, y) = |\Gamma(x) \cap \Gamma(y)|$$

A adjacency matrix

$A_{x,y}^2$:Number of different paths of length 2

Jaccard coefficient:

$$\text{score}(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$$

The probability that both x and y have a feature for a randomly selected feature that either x or y has

Methods for Link Prediction:
Neighborhood-based

Adamic/Adar

$$\mathrm{score}(x, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log |\Gamma(z)|}$$

✓ Assigns large weights to common neighbors *z* of *x* and *y* which themselves have few neighbors (weight rare features more heavily)

- Neighbors who are linked with **2** nodes are assigned weight = 1/log(2)
- Neighbors who are linked with **5** nodes are assigned weight = 1/log(5)

Methods for Link Prediction:

Neighborhood-based

Preferential attachment

Based on the premise that the probability that a new edge has node *x* as its endpoint is proportional to $|\Gamma(x)|$, i.e., nodes like to form ties with 'popular' nodes

$$\mathrm{score}(x, y) = |\Gamma(x)||\Gamma(y)|$$

✓ Researchers found empirical evidence to suggest that co-authorship is correlated with the product of the neighborhood sizes

❖ This depends ***on the degrees*** of the nodes not on their neighbors per se

Methods for Link Prediction: Neighborhood-based

1. Overlap
2. Jaccard
3. Adamic/Adar
4. Preferential attachment

Methods for Link Prediction:
Shortest Path

For $x, y \in V \times V - E_{old}$,
score(x, y) = (negated) length of *shortest path* between x and y

✓ If there are more than *n* pairs of nodes tied for the shortest path length, order them at random.

Methods for Link Prediction: based on the ensemble of all paths

Not just the shortest, but *all* paths between two nodes

Methods for Link Prediction: based on the ensemble of all paths

Katz$_\beta$ measure

$$\text{score}(x, y) := \sum_{\ell=1}^{\infty} \beta^{\ell} \cdot |\text{paths}_{x,y}^{\langle \ell \rangle}|$$

Sum over all paths of length *l*

β > *0* (< 1) is a parameter of the predictor, exponentially damped to count short paths more heavily

✓ *Small β predictions much like common neighbors*

β small, degree, maximal β, eigenvalue

Methods for Link Prediction: based on the ensemble of all paths

Katz$_\beta$ measure

$$\text{score}(x, y) := \sum_{\ell=1}^{\infty} \beta^\ell \cdot |\text{paths}_{x,y}^{\langle \ell \rangle}|$$

$$\sum_{l=1}^{\infty} \beta^l \cdot |\text{paths}_{xy}^{(l)}| = \beta A_{xy} + \beta^2 (A^2)_{xy} + \beta^3 (A^3)_{xy} + \cdots$$

Closed form: $(I - \beta A)^{-1} - I$

- *Unweighted* version, in which $\text{path}_{x,y}^{(1)}$ = **1**, if x and y have collaborated, **0** otherwise
- *Weighted* version, in which $\text{path}_{x,y}^{(1)}$ = **#times** x and y have collaborated

Methods for Link Prediction: based on the ensemble of all paths

Consider a *random walk* on G_{old} that starts at *x* and iteratively moves to a neighbor of *x* chosen uniformly at random from *Γ(x).*

The Hitting Time $H_{x,y}$ from *x* to *y* is the expected number of steps it takes for the random walk starting at *x* to reach *y*.

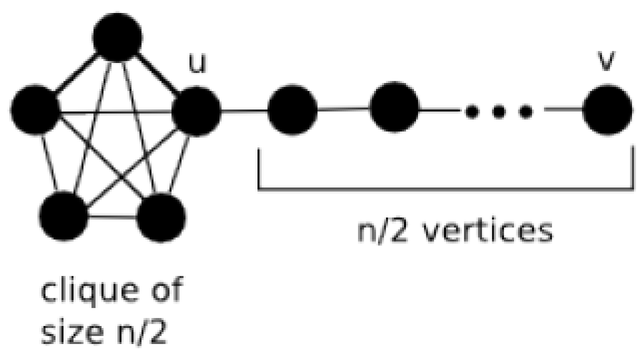
$$\text{score}(x, y) = -H_{x,y}$$

The Commute Time $C_{x,y}$ from *x* to *y* is the expected number of steps to travel from *x* to *y* and from *y* to *x*

$$\text{score}(x, y) = -(H_{x,y} + H_{y,x})$$

Not symmetric, can be shown

$$h_{vu} = \Theta(n^2)$$
$$h_{uv} = \Theta(n^3)$$



Methods for Link Prediction: based on the ensemble of all paths

Example: hit time in a line

Can also consider stationary-normed versions:
(to counteract the fact that $H_{x,y}$ is rather small when y is a node with a large stationary probability)
score(x, y) = $-H_{x,y}\,\pi_y$
score(x, y) = $-(H_{x,y}\,\pi_y + H_{y,x}\,\pi_x)$

Methods for Link Prediction: based on the ensemble of all paths

The hitting time and commute time measures are sensitive to parts of the graph far away from x and y -> periodically *reset the walk*

Random walk with restart: Random walk on G_{old} that starts at *x* and has a probability α of returning to *x* at each step

Rooted PageRank: Starts from *x*, with probability (1 – *a*) moves to a random neighbor and with probability *a* returns to *x*

score(x, y) = stationary probability of y in a rooted PageRank

Methods for Link Prediction: based on the ensemble of all paths

SimRank

Two objects are *similar*, if they are *related to similar objects*

Two objects *x* and *y* are *similar*, if they are related to objects *a* and *b* respectively and *a* and *b* are themselves similar

Average similarity between neighbors of x and neighbors of y

$$\text{similarity}(x, y) := \gamma \cdot \frac{\sum_{a \in \Gamma(x)} \sum_{b \in \Gamma(y)} \text{similarity}(a, b)}{|\Gamma(x)| \cdot |\Gamma(y)|}$$

Base case: similarity(x, x) = 1

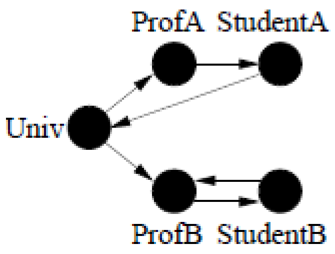
score(x, y) = similarity(x, y)

SimRank

Introduced for directed graphs (similar if referenced by similar objects)

I(x): in-neighbors of x

$$s(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(a), I_j(b))$$



Average similarity between in-neighbors of *a* and in-neighbors of *b*

C a constant between 0 and 1

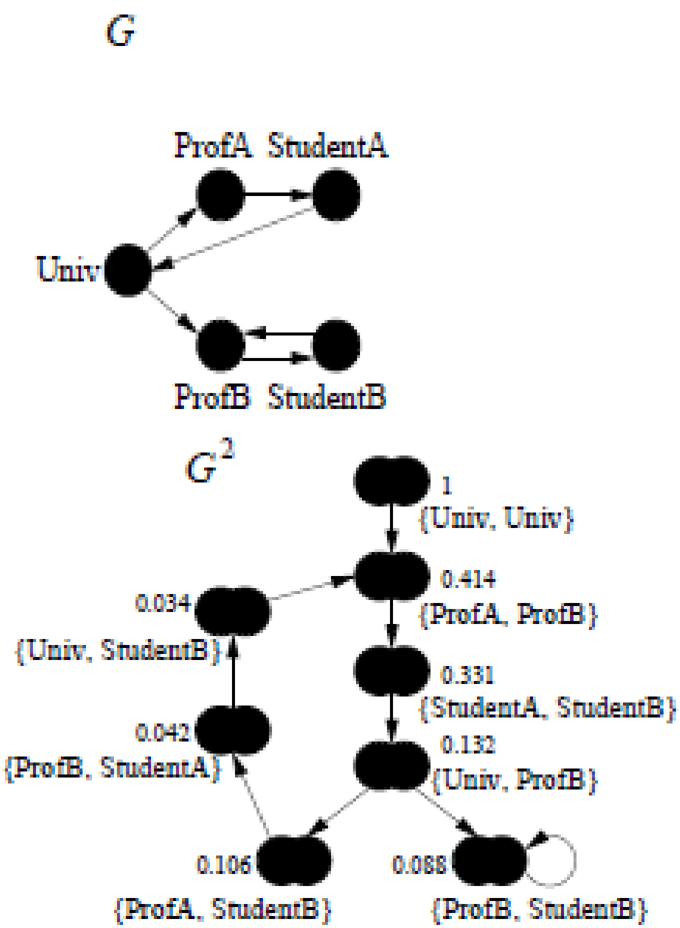
n^2 equations

Iterative computation

$s_0(x, y) = 1$ if $x = y$ and 0 otherwise

s_{k+1} based on the s_k values of its (in-neighbors) computed at iteration k

SimRank as a random walk



Similarity as propagating among pairs

Pair graph G^2:

A node for each pair of nodes

An edge (x, y) → (a, b), if x → a and y → b

Scores *flow* from a node to its neighbors

Computation starts at singleton nodes

C gives the rate of *decay* as similarity flows across edges (*C* = 0.8 in the example)

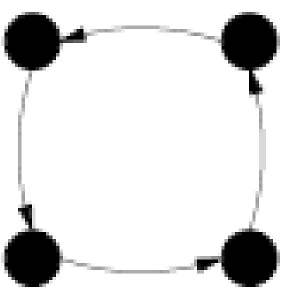
Symmetric pairs (a, b) node same as (b, a) node (with the union of associated edges),

Self-loops

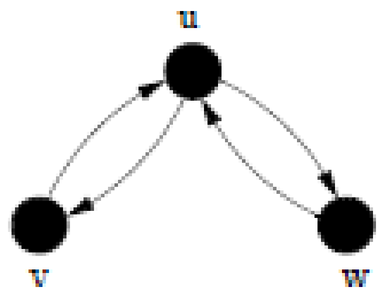
Prune by considering only nodes within a a radius

SimRank

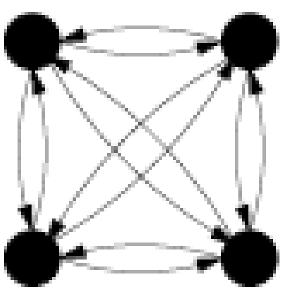
Expected Meeting Distance (EMD): how soon two random surfers are expected to meet at the same node if they started at nodes x and y and randomly walked (in lock step) the graph backwards



$= \infty$



m(u, v) = m(u,w) = ∞, m(v, w) = 1
v and w are much more similar than u is to v or w.



= 3,
a lower similarity than between v and w but higher than between u and v (or u and w).

SimRank

Let us consider G^2
A node (a, b) as a state of the tour in G:
if a moves to c, b moves to d in G,
then (a, b) moves to (c, d) in G^2

A tour in G^2 of length n represents a pair of tours in G where each has length n

What are the states in G^2 that correspond to "meeting" points in G?

SimRank

What are the states in G^2 that correspond to “meeting” points in G?

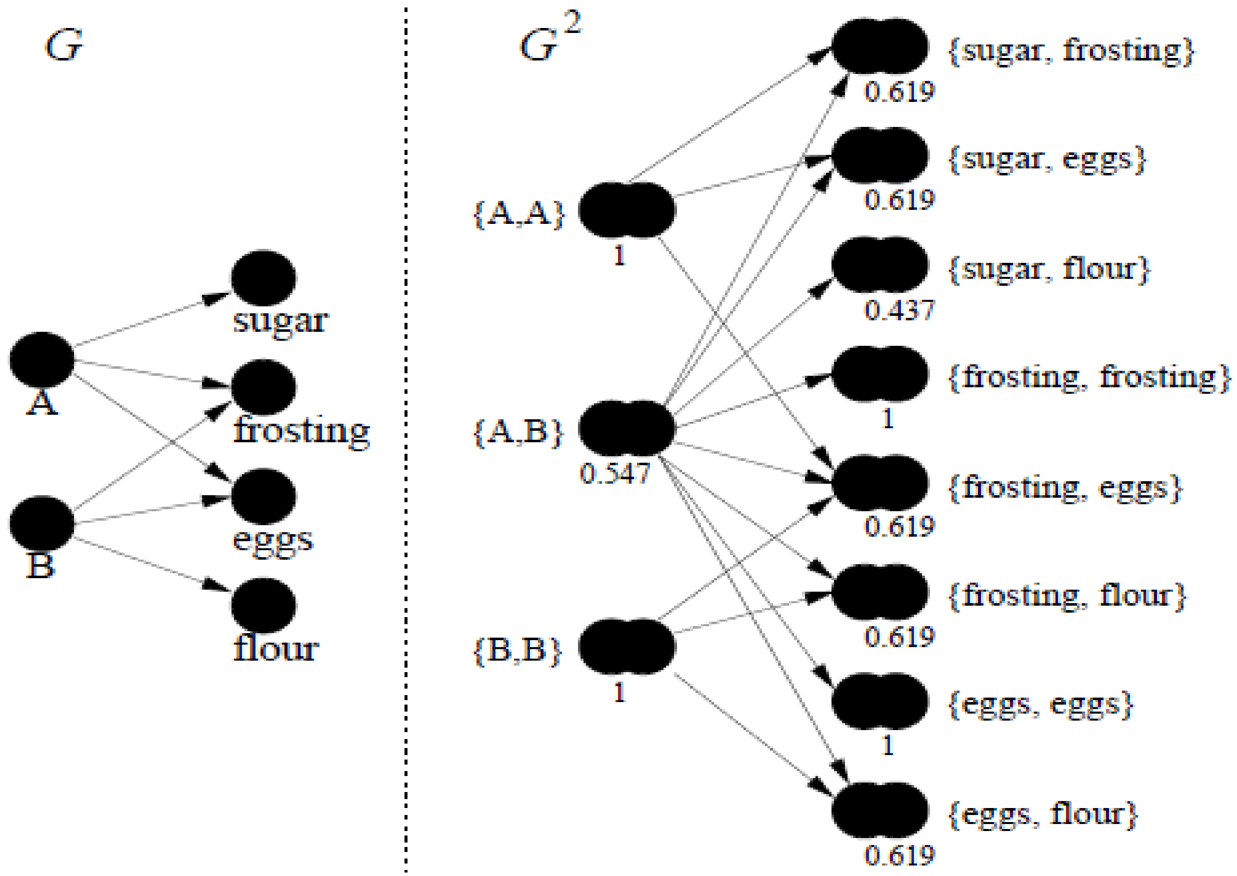
Singleton nodes (common neighbors)

The EMD m(*a*, *b*) is just the expected distance (hitting time) in G^2 between (*a*, *b*) and any singleton node

The sum is taken over all walks that start from (*a*, *b*) and end at a singleton node

This roughly corresponds to the SimRank of (a, b)

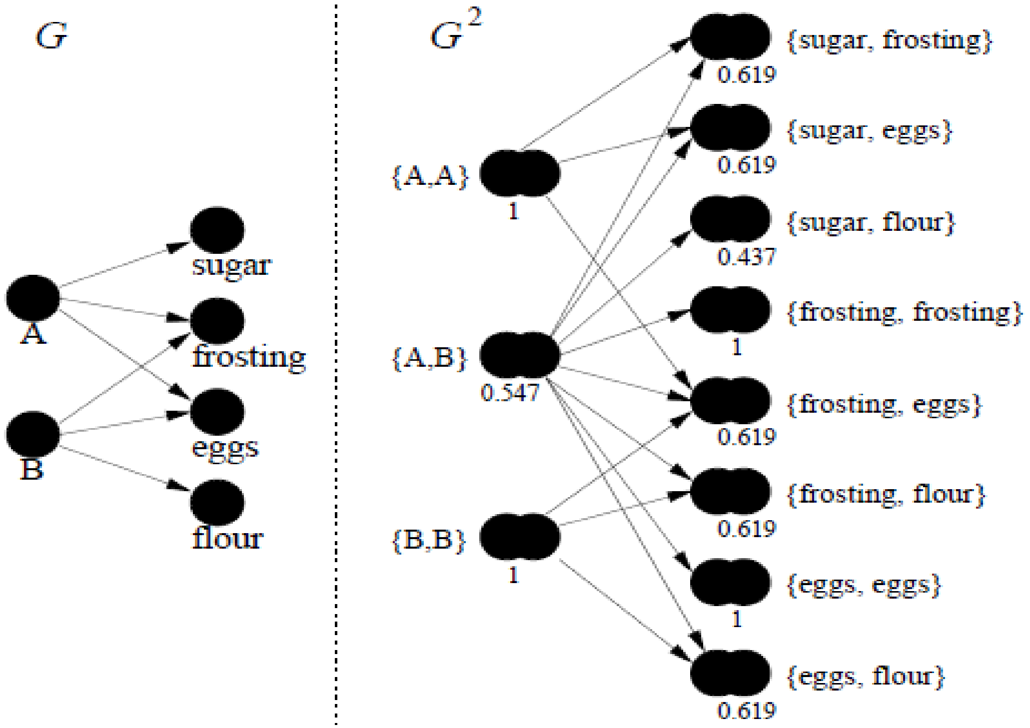
SimRank for bipartite graphs



- People are *similar* if they purchase *similar* items.
- Items are *similar* if they are purchased by *similar* people

Useful for recommendations in general

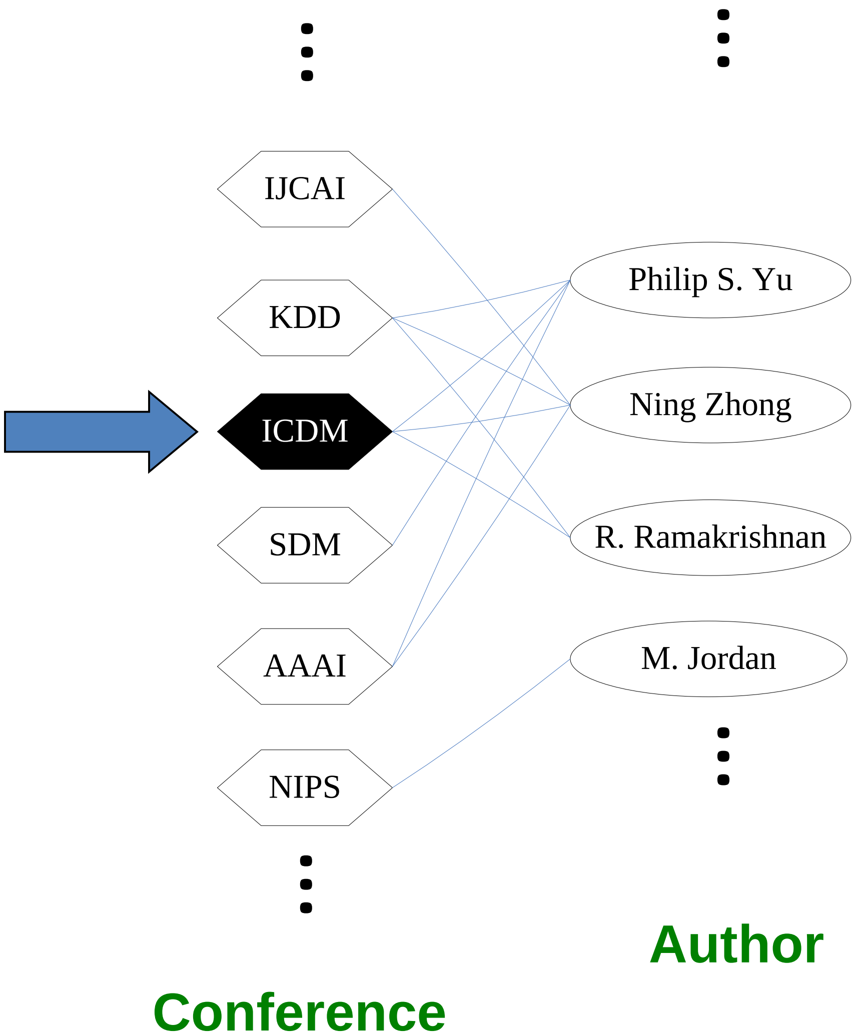
SimRank for bipartite graphs



$$s(A,B) = \frac{C_1}{|O(A)||O(B)|} \sum_{i=1}^{|O(A)|} \sum_{j=1}^{|O(B)|} s(O_i(A), O_j(B))$$

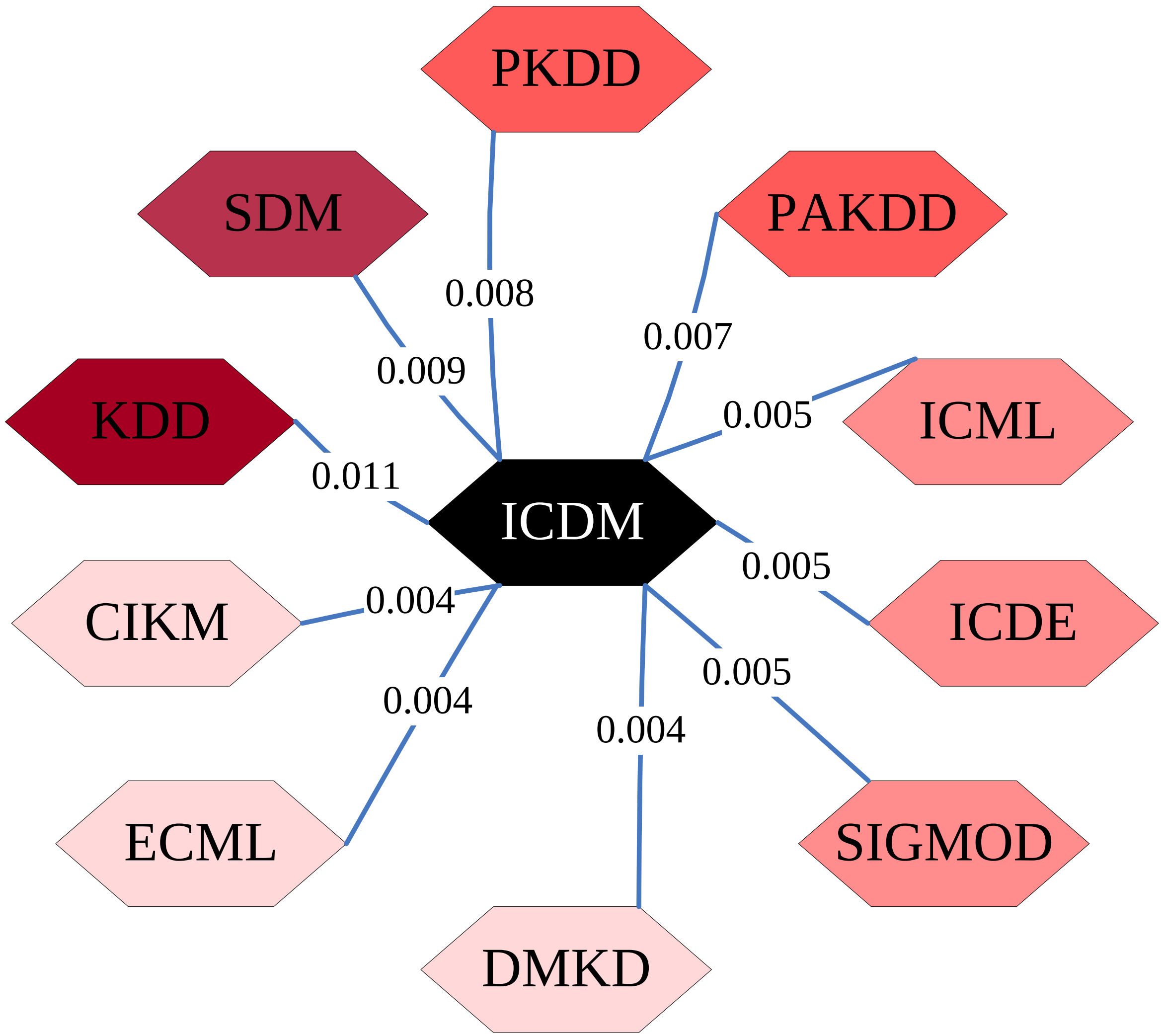
$$s(c,d) = \frac{C_2}{|I(c)||I(d)|} \sum_{i=1}^{|I(c)|} \sum_{j=1}^{|I(d)|} s(I_i(c), I_j(d))$$

SimRank



Q: What is most related conference to ICDM?

SimRank


PKDD
SDM
PAKDD
KDD
ICML
ICDM
CIKM
ICDE
ECML
SIGMOD
DMKD
0.008
0.007
0.009
0.005
0.011
0.005
0.004
0.004
0.005
0.004

Methods for Link Prediction: based on paths

1. Shortest paths
2. Katz
3. Hitting and commute time
4. Rooted PageRank
5. SimRank

Methods for Link Prediction: other

Low rank approximations

M adjacency matrix, represent M with a lower rank matrix M_k

Apply SVD (singular value decomposition)

The *rank-k* matrix that best approximates M

Singular Value Decomposition

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

$$[n \times r]\,[r \times r]\,[r \times n]$$

- **r** : rank of matrix A
- $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r$: singular values (square roots of eigenvals AA^T, A^TA)
- $\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_r$: left singular vectors (eigenvectors of AA^T)
- $\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_r$: right singular vectors (eigenvectors of A^TA)

$$A_r = \sigma_1 \vec{u}_1 \vec{v}_1^T + \sigma_2 \vec{u}_2 \vec{v}_2^T + \cdots + \sigma_r \vec{u}_r \vec{v}_r^T$$

Methods for Link Prediction: other

Unseen Bigrams

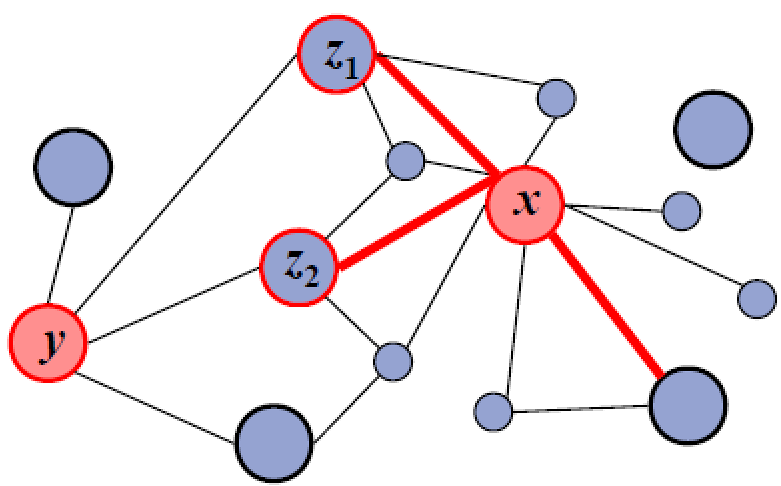
Unseen bigrams: pairs of word that co-occur in a test corpus, but not in the corresponding training corpus

Not just x but also nodes similar to x, similar how?

$S_x^{(\delta)}$ -- δ nodes z with largest score(x, z)

$$\text{score}^*_{unweighted}(x, y) \quad := \quad \left|\{z : z \in \Gamma(y) \cap S_x^{(\delta)}\}\right|$$

$$\text{score}^*_{weighted}(x, y) \quad := \quad \sum_{z \in \Gamma(y) \cap S_x^{(\delta)}} \text{score}(x, z)$$



Methods for Link Prediction: High-level approaches

Clustering

- Compute score(x, y) for al edges in E_{old}
- Delete the (1-p) fraction of the edges whose score is the lowest, for some parameter p
- Recompute score(x, y) for all pairs in the subgraph

Problem Formulation (implementation details)

Prediction for a subset of nodes

Two parameters: $\kappa_{training}$ and κ_{test}

Core: all nodes that are incident to at least $\kappa_{training}$ edges in $G[t_0, t'_0]$, and at least κ_{test} edges in $G[t_1, t'_1]$

❖ *Predict new edges between the nodes in Core*

Example Dataset: co-authorship

	training period			Core		
	authors	papers	collaborations[1]	authors	$\|E_{old}\|$	$\|E_{new}\|$
astro-ph	5343	5816	41852	1561	6178	5751
cond-mat	5469	6700	19881	1253	1899	1150
gr-qc	2122	3287	5724	486	519	400
hep-ph	5414	10254	47806	1790	6654	3294
hep-th	5241	9498	15842	1438	2311	1576

t_0 = 1994, t'_0 = 1996: **training interval -> [1994, 1996]**
t_1 = 1997, t'_1 = 1999: **test interval -> [1997, 1999]**

- G_{collab} = <V, E_{old}> = G[1994, 1996]
- E_{new}: authors in V that co-author a paper during the test interval but not during the training interval

$\kappa_{training}$ = *3*, κ_{test} = 3: **Core** consists of all authors who have written at least 3 papers during the training period and at least 3 papers during the test period

Predict E_{new}

How to Evaluate the Prediction (outline)

Each link predictor p outputs a ranked list L_p of pairs in $V \times V - E_{old}$: predicted new collaborations in decreasing order of confidence

❖ *How many of the top-n (relevant) predictions are correct (precision?)*

Define n as $|E*_{new}|$

$$E*_{new} = E_{new} \cap (\text{Core} \times \text{Core}) = |E*_{new}|$$

Evaluation method: *Size of the intersection* of

- the first n edge predictions from L_p that are in Core × Core, and
- the set $E*_{new}$

Precision at recall

Evaluation: baseline

Baseline: random predictor

Randomly select pairs of authors who did not collaborate in the training interval

Probability that a random prediction is correct:

$$\frac{|E_{new}|}{\binom{|\text{Core}|}{2} - |E_{old}|}$$

In the datasets, from 0.15% (cond-mat) to 0.48% (astro-ph)

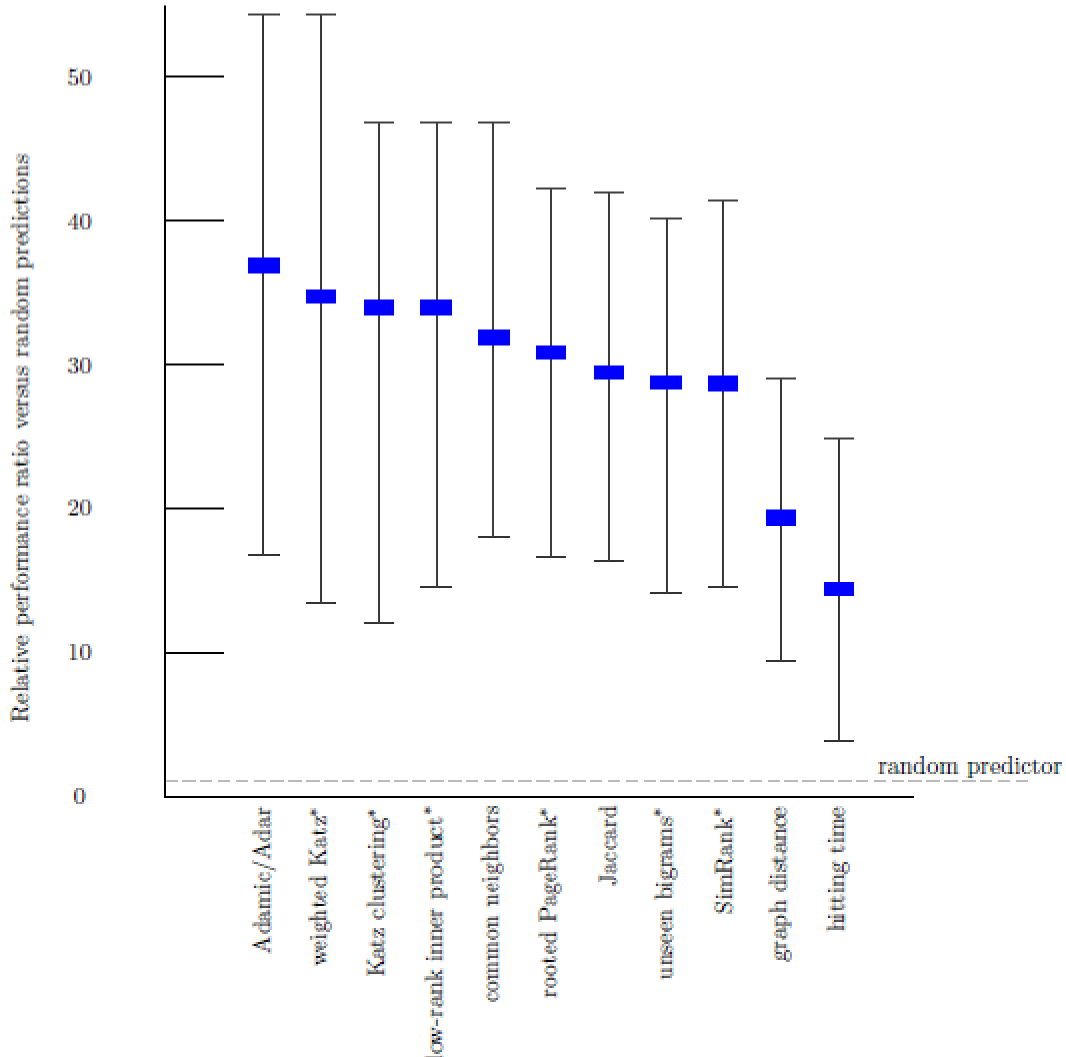
Evaluation: Factor improvement over random

predictor		astro-ph	cond-mat	gr-qc	hep-ph	hep-th
probability that a random prediction is correct		0.475%	0.147%	0.341%	0.207%	0.153%
graph distance (all distance-two pairs)		*9.4*	*25.1*	*21.3*	*12.0*	*29.0*
common neighbors		**18.0**	**40.8**	**27.1**	**26.9**	**46.9**
preferential attachment		4.7	6.0	7.5	*15.2*	7.4
Adamic/Adar		*16.8*	**54.4**	**30.1**	**33.2**	**50.2**
Jaccard		*16.4*	**42.0**	19.8	**27.6**	*41.5*
SimRank	$\gamma = 0.8$	*14.5*	*39.0*	*22.7*	*26.0*	*41.5*
hitting time		6.4	23.7	*24.9*	3.8	13.3
hitting time—normed by stationary distribution		5.3	23.7	11.0	11.3	21.2
commute time		5.2	15.4	**33.0**	*17.0*	23.2
commute time—normed by stationary distribution		5.3	16.0	11.0	11.3	16.2
rooted PageRank	$\alpha = 0.01$	*10.8*	*27.8*	**33.0**	*18.7*	*29.1*
	$\alpha = 0.05$	*13.8*	*39.6*	**35.2**	*24.5*	*41.1*
	$\alpha = 0.15$	*16.6*	**40.8**	**27.1**	**27.5**	*42.3*
	$\alpha = 0.30$	*17.1*	**42.0**	*24.9*	**29.8**	*46.5*
	$\alpha = 0.50$	*16.8*	**40.8**	*24.2*	**30.6**	*46.5*
Katz (weighted)	$\beta = 0.05$	3.0	21.3	19.8	2.4	12.9
	$\beta = 0.005$	*13.4*	**54.4**	**30.1**	*24.0*	**51.9**
	$\beta = 0.0005$	*14.5*	**53.8**	**30.1**	**32.5**	**51.5**
Katz (unweighted)	$\beta = 0.05$	*10.9*	**41.4**	**37.4**	*18.7*	**47.7**
	$\beta = 0.005$	*16.8*	**41.4**	**37.4**	*24.1*	**49.4**
	$\beta = 0.0005$	*16.7*	**41.4**	**37.4**	*24.8*	**49.4**

Evaluation: Factor improvement over random

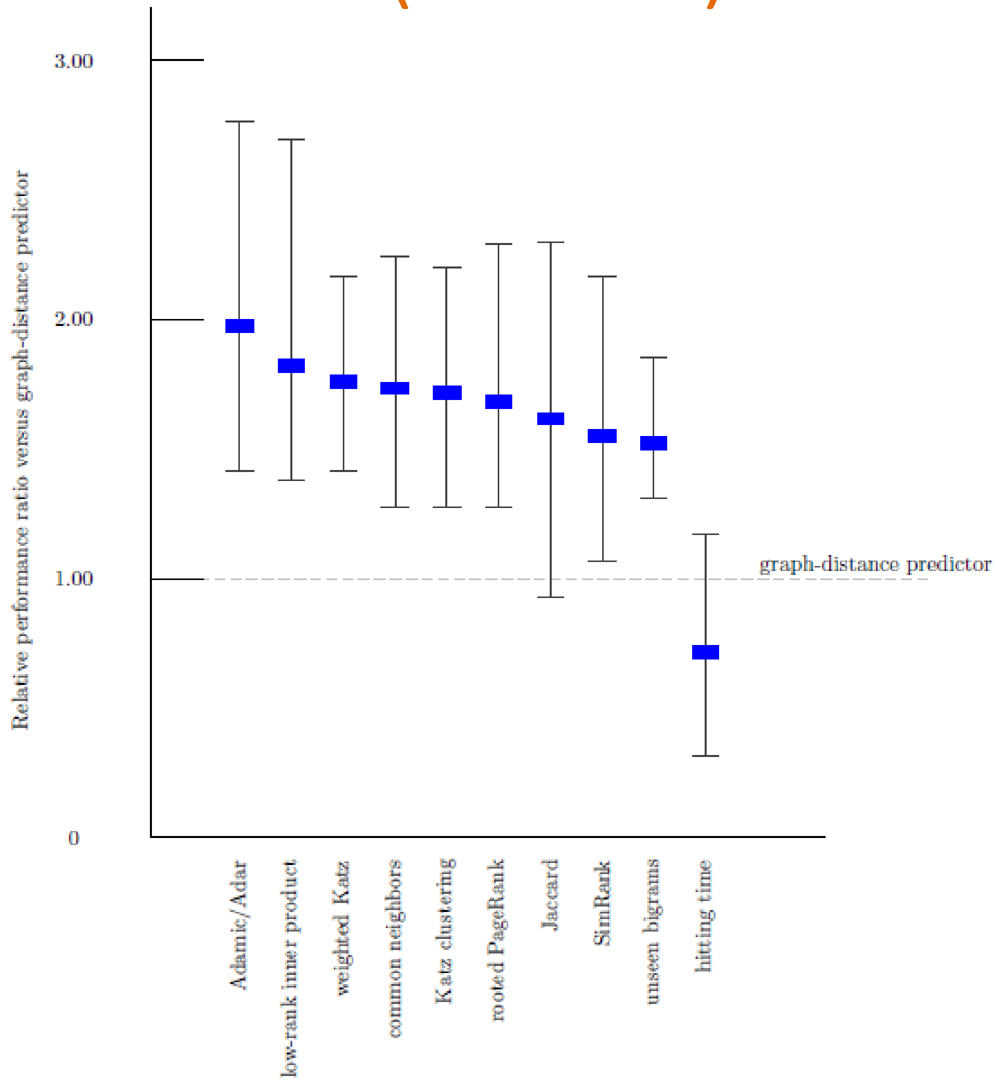
predictor		astro-ph	cond-mat	gr-qc	hep-ph	hep-th
probability that a random prediction is correct		0.475%	0.147%	0.341%	0.207%	0.153%
graph distance (all distance-two pairs)		*9.4*	*25.1*	*21.3*	*12.0*	*29.0*
common neighbors		**18.0**	**40.8**	**27.1**	**26.9**	**46.9**
Low-rank approximation: Inner product	rank $= 1024$	*15.2*	**53.8**	**29.3**	**34.8**	**49.8**
	rank $= 256$	*14.6*	**46.7**	**29.3**	**32.3**	**46.9**
	rank $= 64$	*13.0*	**44.4**	**27.1**	**30.7**	**47.3**
	rank $= 16$	*10.0*	21.3	**31.5**	**27.8**	*35.3*
	rank $= 4$	8.8	15.4	**42.5**	*19.5*	22.8
	rank $= 1$	6.9	5.9	**44.7**	*17.6*	14.5
Low-rank approximation: Matrix entry	rank $= 1024$	8.2	16.6	6.6	*18.5*	21.6
	rank $= 256$	*15.4*	*36.1*	8.1	*26.2*	*37.4*
	rank $= 64$	*13.7*	**46.1**	16.9	**28.1**	*40.7*
	rank $= 16$	9.1	21.3	*26.4*	*23.1*	*34.0*
	rank $= 4$	8.8	15.4	**39.6**	*20.0*	22.4
	rank $= 1$	6.9	5.9	**44.7**	*17.6*	14.5
Low-rank approximation: Katz ($\beta = 0.005$)	rank $= 1024$	*11.4*	*27.2*	**30.1**	**27.0**	*32.0*
	rank $= 256$	*15.4*	**42.0**	11.0	**34.2**	*38.6*
	rank $= 64$	*13.1*	**45.0**	19.1	**32.2**	*41.1*
	rank $= 16$	9.2	21.3	**27.1**	*24.8*	*34.9*
	rank $= 4$	7.0	15.4	**41.1**	*19.7*	22.8
	rank $= 1$	0.4	5.9	**44.7**	*17.6*	14.5
unseen bigrams (weighted)	common neighbors, $\delta = 8$	*13.5*	*36.7*	**30.1**	*15.6*	**46.9**
	common neighbors, $\delta = 16$	*13.4*	*39.6*	**38.9**	*18.5*	**48.6**
	Katz ($\beta = 0.005$), $\delta = 8$	*16.8*	*37.9*	*24.9*	*24.1*	**51.1**
	Katz ($\beta = 0.005$), $\delta = 16$	*16.5*	*39.6*	**35.2**	*24.7*	**50.6**
unseen bigrams (unweighted)	common neighbors, $\delta = 8$	*14.1*	*40.2*	**27.9**	*22.2*	*39.4*
	common neighbors, $\delta = 16$	*15.3*	*39.0*	**42.5**	*22.0*	*42.3*
	Katz ($\beta = 0.005$), $\delta = 8$	*13.1*	*36.7*	**32.3**	*21.6*	*37.8*
	Katz ($\beta = 0.005$), $\delta = 16$	*10.3*	*29.6*	**41.8**	*12.2*	*37.8*
clustering: Katz ($\beta_1 = 0.001, \beta_2 = 0.1$)	$\rho = 0.10$	7.4	*37.3*	**46.9**	**32.9**	*37.8*
	$\rho = 0.15$	*12.0*	**46.1**	**46.9**	*21.0*	*44.0*
	$\rho = 0.20$	4.6	*34.3*	19.8	*21.2*	*35.7*
	$\rho = 0.25$	3.3	*27.2*	20.5	*19.4*	17.4

Evaluation: Average relevance performance

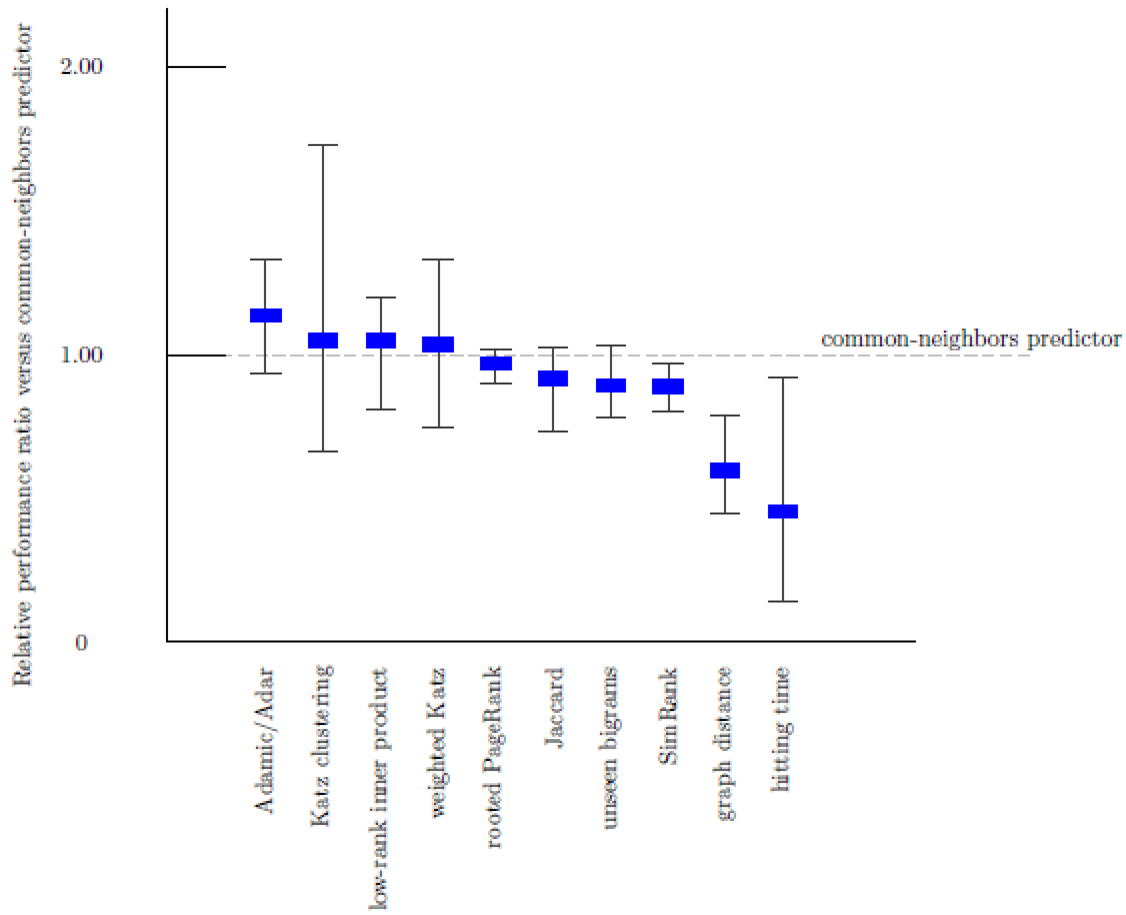


- average ratio over the five datasets of the given predictor's performance *versus a baseline* predictor's performance.
- the error bars indicate the minimum and maximum of this ratio over the five datasets.
- the parameters for the starred predictors are: (1) for weighted Katz, β= 0.005; (2) for Katz clustering, β1 = 0.001; ρ = 0.15; β2 = 0.1; (3) for low-rank inner product, rank = 256; (4) for rooted Pagerank, α = 0.15; (5) for unseen bigrams, unweighted, common neighbors with δ = 8; and (6) for SimRank, C (γ) = 0.8.

Evaluation: Average relevance performance (distance)



Evaluation: Average relevance performance (neighbors)



Evaluation: prediction overlap

	Adamic/Adar	Katz clustering	common neighbors	hitting time	Jaccard's coefficient	weighted Katz	low-rank inner product	rooted Pagerank	SimRank	unseen bigrams
Adamic/Adar	1150	638	520	193	442	1011	905	528	372	486
Katz clustering		1150	411	182	285	630	623	347	245	389
common neighbors			1150	135	506	494	467	305	332	489
hitting time				1150	87	191	192	247	130	156
Jaccard's coefficient					1150	414	382	504	845	458
weighted Katz						1150	1013	488	344	474
low-rank inner product							1150	453	320	448
rooted Pagerank								1150	678	461
SimRank									1150	423
unseen bigrams										1150

❖ How much similar are the predictions made by the different methods?

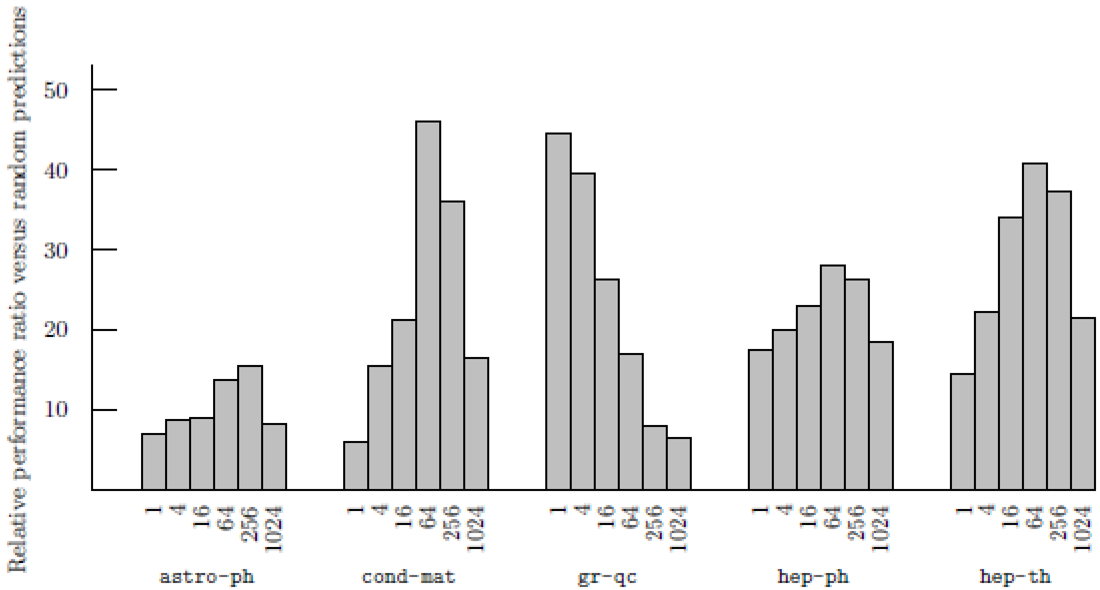
Why?

correct

	Adamic/Adar	Katz clustering	common neighbors	hitting time	Jaccard's coefficient	weighted Katz	low-rank inner product	rooted Pagerank	SimRank	unseen bigrams
Adamic/Adar	92	65	53	22	43	87	72	44	36	49
Katz clustering		78	41	20	29	66	60	31	22	37
common neighbors			69	13	43	52	43	27	26	40
hitting time				40	8	22	19	17	9	15
Jaccard's coefficient					71	41	32	39	51	43
weighted Katz						92	75	44	32	51
low-rank inner product							79	39	26	46
rooted Pagerank								69	48	39
SimRank									66	34
unseen bigrams										68

Evaluation: datasets

❖ How much does the performance of the different methods depends on the dataset?



- (rank) On 4 of the 5 datasets best at an intermediate rank
  On qr-qc, best at rank 1, does it have a "simpler" structure"?
- On hep-ph, preferential attachment the best
- Why is astro-ph "difficult"?

The culture of physicists and physics collaboration

Evaluation: small world

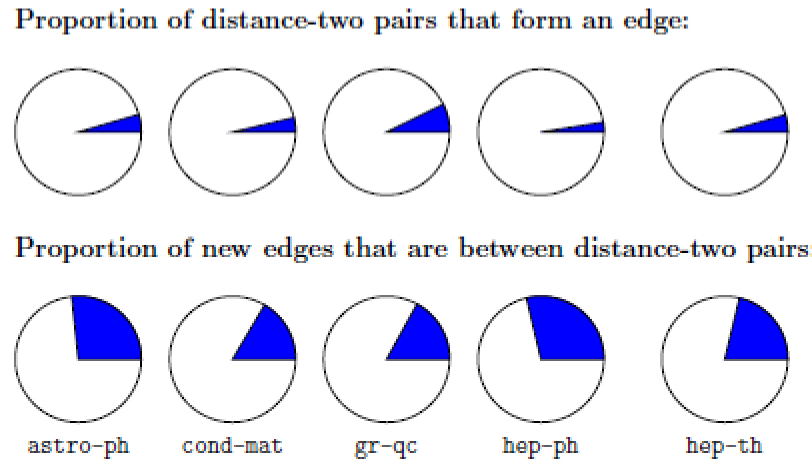
The shortest path even in unrelated disciplines is often very short

Basic classifier on graph distances does not work

Evaluation: restricting to distance three

Many pairs of authors separated by a graph distance of 2 will not collaborate and
Many pairs who collaborate are at distance greater than 2

Disregard all distance 2 pairs (do not just "close" triangles)



	astro-ph	cond-mat	gr-qc	hep-ph	hep-th
# pairs at distance two	33862	5145	935	37687	7545
# new collaborations at distance two	1533	190	68	945	335
# new collaborations	5751	1150	400	3294	1576

predictor		astro-ph	cond-mat	gr-qc	hep-ph	hep-th
graph distance (all distance-three pairs)		2.8	5.4	7.7	4.0	8.6
preferential attachment		3.2	2.6	8.6	4.7	1.4
SimRank	$\gamma = 0.8$	5.9	14.3	10.6	7.6	21.9
hitting time		4.4	10.1	13.7	4.5	4.7
hitting time—normed by stationary distribution		2.0	2.5	0.0	2.5	6.6
commute time		3.8	5.9	21.1	5.9	6.6
commute time—normed by stationary distribution		2.6	0.8	1.1	4.8	4.7
rooted PageRank	$\alpha = 0.01$	4.6	12.7	21.1	6.5	12.6
	$\alpha = 0.05$	5.3	13.5	21.1	8.7	16.6
	$\alpha = 0.15$	5.4	11.8	18.0	10.7	19.9
	$\alpha = 0.30$	5.8	13.5	8.4	11.6	19.9
	$\alpha = 0.50$	6.3	15.2	7.4	12.7	19.9
Katz (weighted)	$\beta = 0.05$	1.5	5.9	11.6	2.3	2.7
	$\beta = 0.005$	5.5	14.3	28.5	4.2	12.6
	$\beta = 0.0005$	6.2	13.5	27.5	4.2	12.6
Katz (unweighted)	$\beta = 0.05$	2.3	12.7	30.6	9.0	12.6
	$\beta = 0.005$	9.1	11.8	30.6	5.1	17.9
	$\beta = 0.0005$	9.2	11.8	30.6	5.1	17.9
Low-rank approximation: Inner product	rank = 1024	2.3	2.5	9.5	4.0	6.0
	rank = 256	4.8	5.9	5.3	9.9	10.6
	rank = 64	3.8	12.7	5.3	7.1	11.3
	rank = 16	5.3	6.7	6.3	6.8	15.3
	rank = 4	5.1	6.7	32.7	2.0	4.7
	rank = 1	6.1	2.5	32.7	4.2	8.0
Low-rank approximation: Matrix entry	rank = 1024	4.1	6.7	6.3	5.9	13.3
	rank = 256	3.8	8.4	3.2	8.5	19.9
	rank = 64	2.9	11.8	2.1	4.0	10.0
	rank = 16	4.4	8.4	4.2	5.9	16.6
	rank = 4	4.9	6.7	27.5	2.0	4.7
	rank = 1	6.1	2.5	32.7	4.2	8.0
Low-rank approximation: Katz ($\beta = 0.005$)	rank = 1024	4.3	6.7	28.5	5.9	13.3
	rank = 256	3.6	8.4	3.2	8.5	20.6
	rank = 64	2.8	11.8	2.1	4.2	10.6
	rank = 16	5.0	8.4	5.3	5.9	15.9
	rank = 4	5.2	6.7	28.5	2.0	4.7
	rank = 1	0.3	2.5	32.7	4.2	8.0
unseen bigrams (weighted)	common neighbors, $\delta = 8$	5.8	6.7	14.8	4.2	23.9
	common neighbors, $\delta = 16$	7.9	9.3	28.5	5.1	19.3
	Katz ($\beta = 0.005$), $\delta = 8$	5.2	10.1	22.2	2.8	17.9
	Katz ($\beta = 0.005$), $\delta = 16$	6.6	10.1	29.6	3.7	15.3
unseen bigrams (unweighted)	common neighbors, $\delta = 8$	5.4	5.1	13.7	4.5	21.3
	common neighbors, $\delta = 16$	6.3	8.4	25.3	4.8	21.9
	Katz ($\beta = 0.005$), $\delta = 8$	4.1	7.6	22.2	2.0	17.3
	Katz ($\beta = 0.005$), $\delta = 16$	4.3	4.2	28.5	3.1	16.6
clustering: Katz ($\beta_1 = 0.001, \beta_2 = 0.1$)	$\rho = 0.10$	3.2	4.2	31.7	7.1	8.6
	$\rho = 0.15$	4.6	4.2	32.7	7.6	6.6
	$\rho = 0.20$	2.3	5.9	7.4	4.5	8.0
	$\rho = 0.25$	2.0	11.8	6.3	6.8	5.3

Evaluation: the breadth of data

Three additional datasets

1. Proceedings of STOC and FOCS
2. Papers for Citeseer
3. All five of the arXiv sections

STOC/FOCS	arXiv sections	combined arXiv sections	Citeseer
6.1	18.0—46.9	71.2	147.0

Common neighbors vs Random

✓ Suggests that is easier to predict links *within communities*

Extensions

- ❖ Improve performance. Even the best (Katz clustering on gr-qc) correct on only about 16% of its prediction

- ❖ Improve efficiency on very large networks (approximation of distances)

- ❖ Treat more recent collaborations as more important

- ❖ Additional information (paper titles, author institutions, etc)
To some extent latently present in the graph

Summary

Problem definition
Compute score(u, v)
Neighborhood-based
- common neighbors
- Jaccard coefficient
- Adamic/Adar
- preferential attachment

Path-based
- shortest path
- Katz
- hitting time, commute time – normed by stationary distribution
- rooted Page Rank
- SimRank

Summary (SimRank)

Two objects are as *similar*, as their neighbors

$$\text{similarity}(x, y) := \gamma \cdot \frac{\sum_{a \in \Gamma(x)} \sum_{b \in \Gamma(y)} \text{similarity}(a, b)}{|\Gamma(x)| \cdot |\Gamma(y)|}$$

score(x, y) = similarity(x, y)

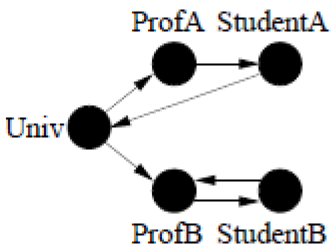
Base case: similarity(x, x) = 1

Average similarity between neighbors of x and neighbors of y

Summary (SimRank)

Introduced for directed graphs (similar if referenced by similar objects)

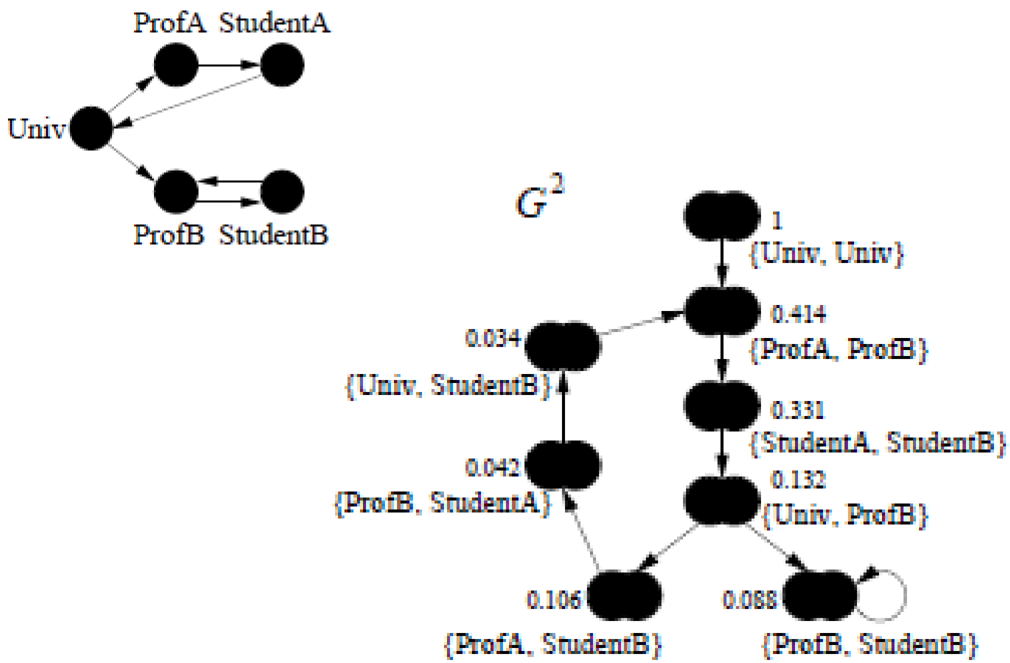
I(x): in-neighbors of x



$$s(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(a), I_j(b))$$

Expected meeting point of two random surfers that move backwards in lock step

Summary (SimRank)



Pair graph G^2:
A node for each pair of nodes
An edge (x, y) → (a, b), if x → a and y → b

Scores *flow* from a node to its neighbors
Computation starts at singleton nodes
C gives the rate of *decay* as similarity flows across edges (*C* = 0.8 in the example)

A tour in G^2 of length n represents a pair of tours in G where each has length n

The EMD m(*a*, *b*) is just the hitting time in G^2 between (*a*, *b*) and any singleton node

Summary (Evaluation)

Output

a list L_p of pairs in V × V − E_{old} ranked by score
(predicted new links in decreasing order of confidence)

Precision at recall

- How many of the top-*n* predictions are correct where *n* = $|E_{new}|$

Improvement over baseline

Baseline: random predictor

Probability that a random prediction is correct: $$\frac{|E_{new}|}{\binom{|V|}{2} - |E_{old}|}$$

Preprocessing:

- Core
- Low Rank Approximation, ignore low score, add friends

Outline

- Estimating a score for each edge (seminal work of Liben-Nowell&Kleinberg
  - Neighbors measures, Distance measures, Other methods
  - Evaluation
- Classification approach
- Twitter

Using Supervised Learning

Given a collection of records (*training set*)

Each record contains
a set of *attributes (features) +* the *class attribute*.

Find a *model* for the class attribute as a function of the values of other attributes.

Goal: previously unseen records should be assigned a class as accurately as possible.

A test set is used to determine the accuracy of the model.

Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

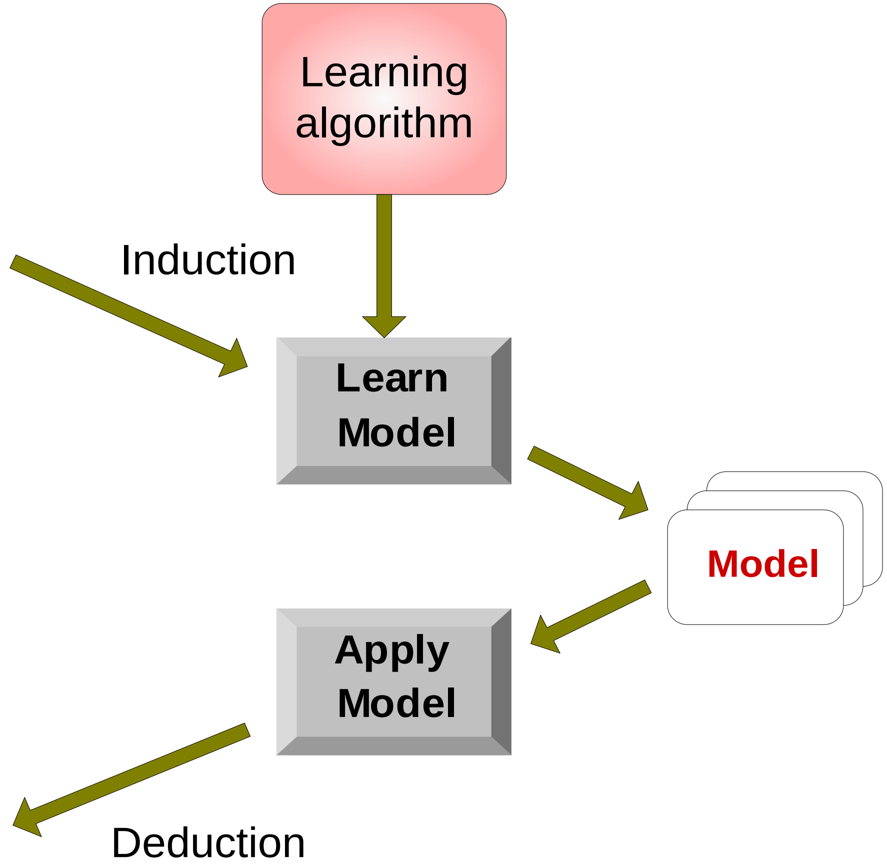
Illustrating the Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Classification Techniques

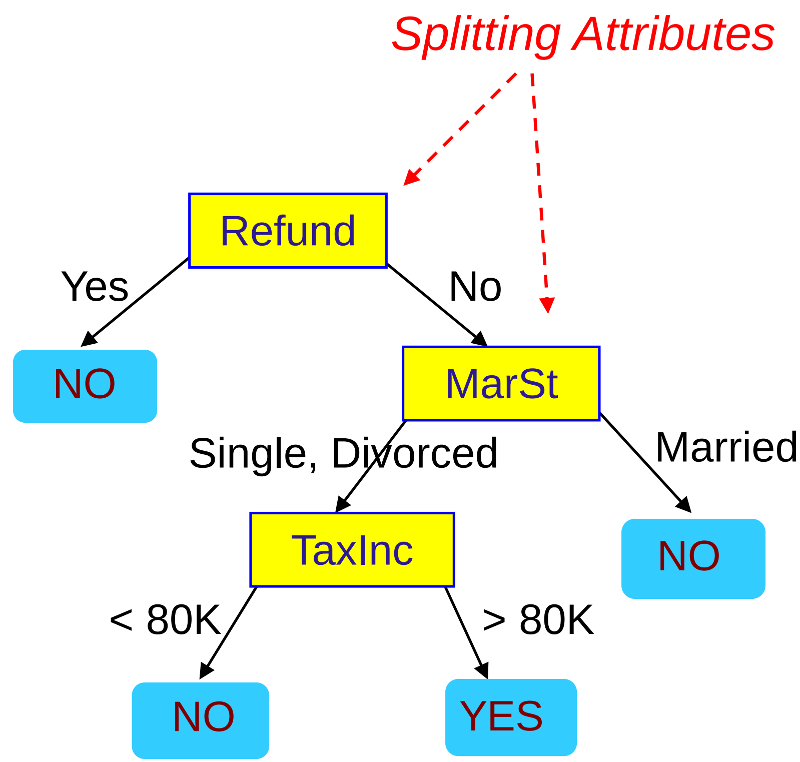
- Decision Tree based methods
- Rule-based methods
- Memory based reasoning
- Neural networks
- Naïve Bayes and Bayesian Belief networks
- Support vector machines
- Logistic regression

Example of a Decision Tree

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



Model: Decision Tree

Classification for Link Prediction

Class?
Features (predictors)?

Name	Parameters	HPLP	HPLP+
In-Degree(i)	-	✓	✓
In-Volume(i)	-	✓	✓
In-Degree(j)	-	✓	✓
In-Volume(j)	-	✓	✓
Out-Degree(i)	-	✓	✓
Out-Volume(i)	-	✓	✓
Out-Degree(j)	-	✓	✓
Out-Volume(j)	-	✓	✓
Common Nbrs(i,j)	-	✓	✓
Max. Flow(i,j)	$l = 5$	✓	✓
Shortest Paths(i,j)	$l = 5$	✓	✓
PropFlow(i,j)	$l = 5$	✓	✓
Adamic/Adar(i,j)	-		✓
Jaccard's Coef(i,j)	-		✓
Katz(i,j)	$l = 5,\ \beta = 0.005$		✓
Pref Attach(i,j)	-		✓

PropFlow: corresponds to the probability that a restricted random walk starting at *x* ends at *y* in *l* steps or fewer using link weights as transition probabilities (stops in *l* steps or if revisits a node)

How to construct the training set

When to extract features and when to determine class?

Two time instances τ_x and τ_y

- From t_0 to τ_x construct graph and extract features (G_{old})
- From $\tau_x + 1$ to τ_y examine if a link appears (determine class value)

What are good values

- Large τ_x better topological features (as the network reaches saturation)
- Large τ_y larger number of positives (size of positive class)
- Should also match the real-world prediction interval

How to construct the training set

Unsupervised

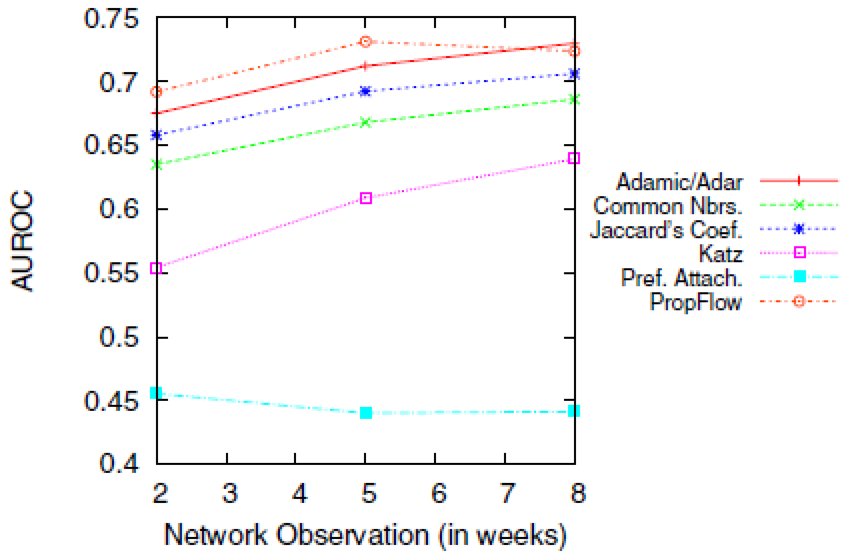


Figure 1: Performance in the second-degree neighborhood as a function of τ_x.

Datasets

712 million cellular phone calls

- weighted, directed networks, weights correspond to the number of calls
- use the first 5 weeks of data (5.5M nodes, 19.7M links) for extracting

features and the sixth week (4.4M nodes, 8.5M links) for obtaining ground truth.

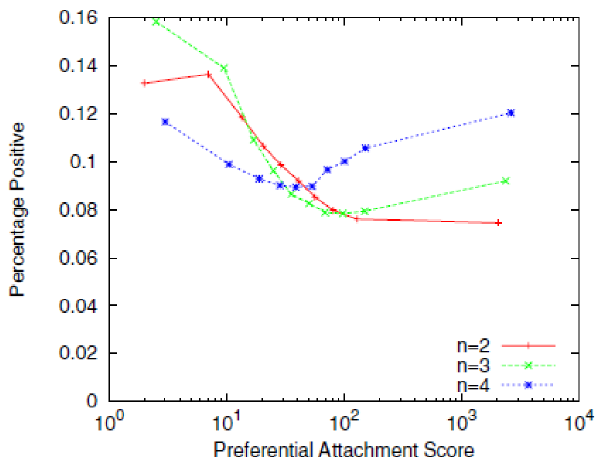
19,464 condensed matter physics collaborations from 1995 to 2000.

- weighted, undirected networks, weights correspond to the number of collaborations two authors share.
- use the years 1995 to 1999 (13.9K nodes, 80.6K links) for extracting features and the year 2000 (8.5K nodes, 41.0K links) for obtaining ground truth.

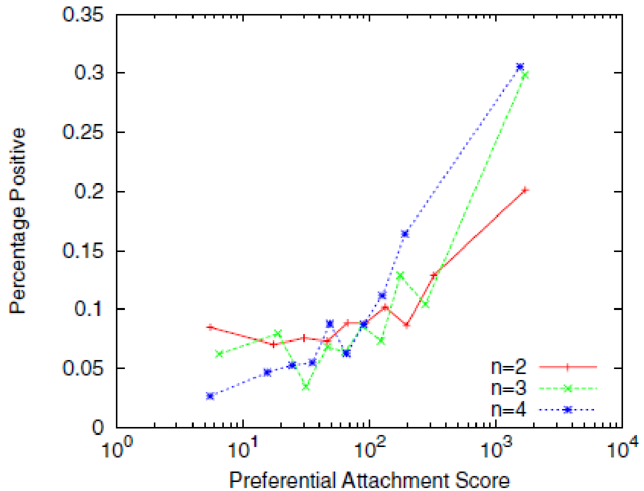
Table 1: Network Characteristics

	phone	condmat
Assortativity Coef.	0.293	0.177
Average Clustering Coef.	0.187	0.642
Mean Degree	3.88	6.42
Median Degree	3	4
Number of SCCs	1,023,044	652
Largest SCC	4,293,751	15,081
Largest SCC Diameter	25	19

Using Supervised Learning: why?



(a) phone

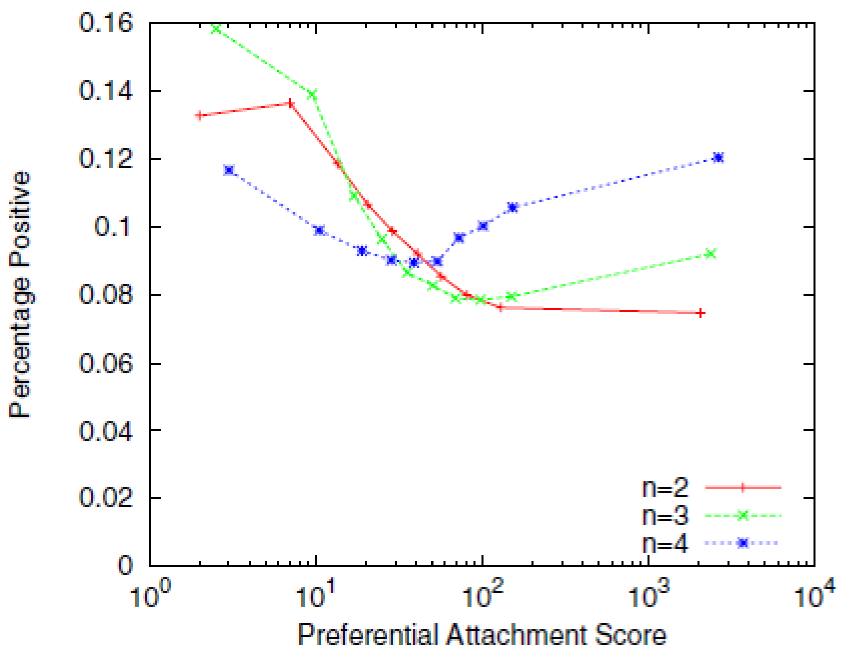


(b) condmat

A different prediction model for each distance

- Predictors that work well in one network not in another
- Should increase with the score (not in phone)
- Preferential attachment increase with distance (when other may fail)

Using Supervised Learning: why?



- Even training on a single feature may outperform ranking (if no clear bound on score)
- Dependencies between features – use an ensemble of features

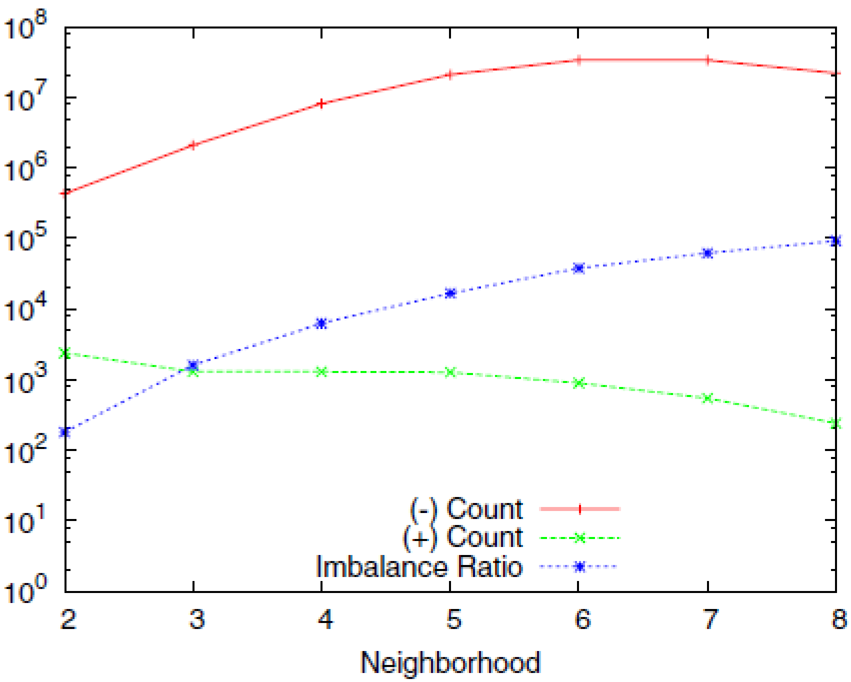
Imbalance

- Sparse networks: |E| = k |V| for constant k << |V|

The class imbalance ratio for link prediction in a sparse network is Ω(|V|/1) when at most |V| nodes are added

Missing links is $|V|^2$
Positives V

n-neigborhood exactly n hops way
Treat each neighborhood as a separate problem



Metrics for Performance Evaluation

Confusion Matrix:

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	TP	FN
	Class=No	FP	TN

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

ROC Curve

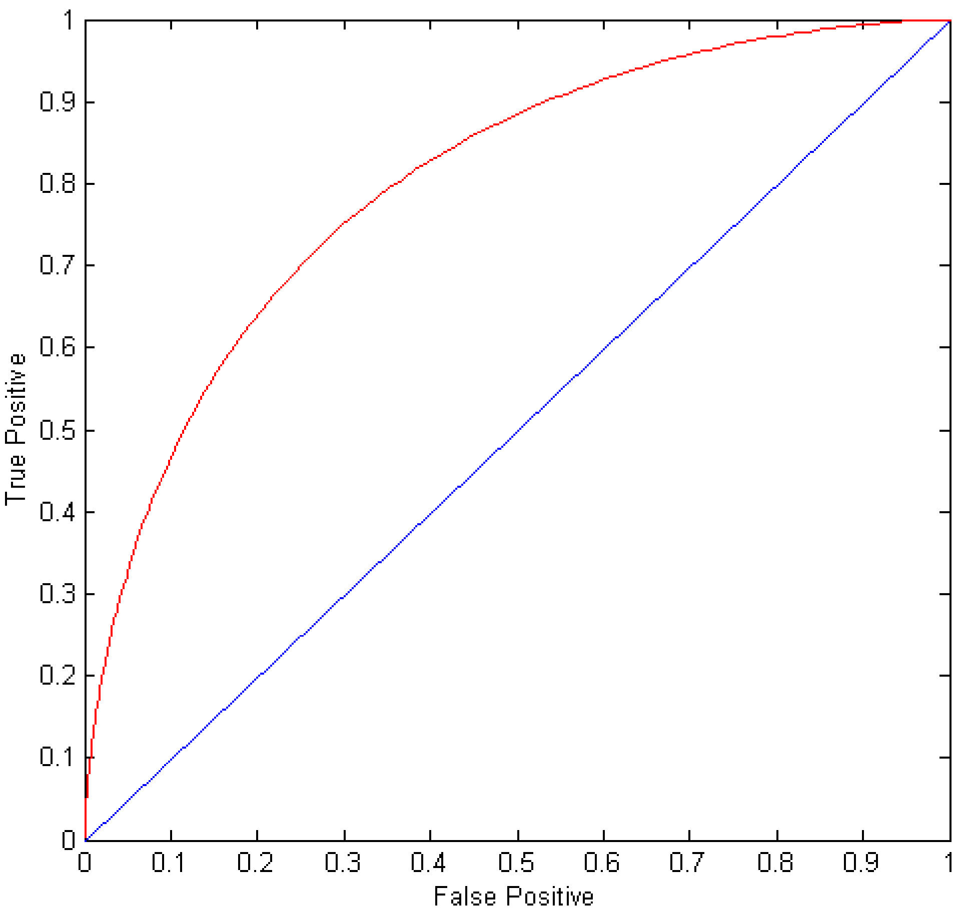
TPR (sensitivity)=TP/(TP+FN) (percentage of positive classified as positive)

FPR = FP/(TN+FP) (percentage of negative classified as positive)

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (0,1): ideal

Diagonal line: Random guessing

Below diagonal line: prediction is opposite of the true class



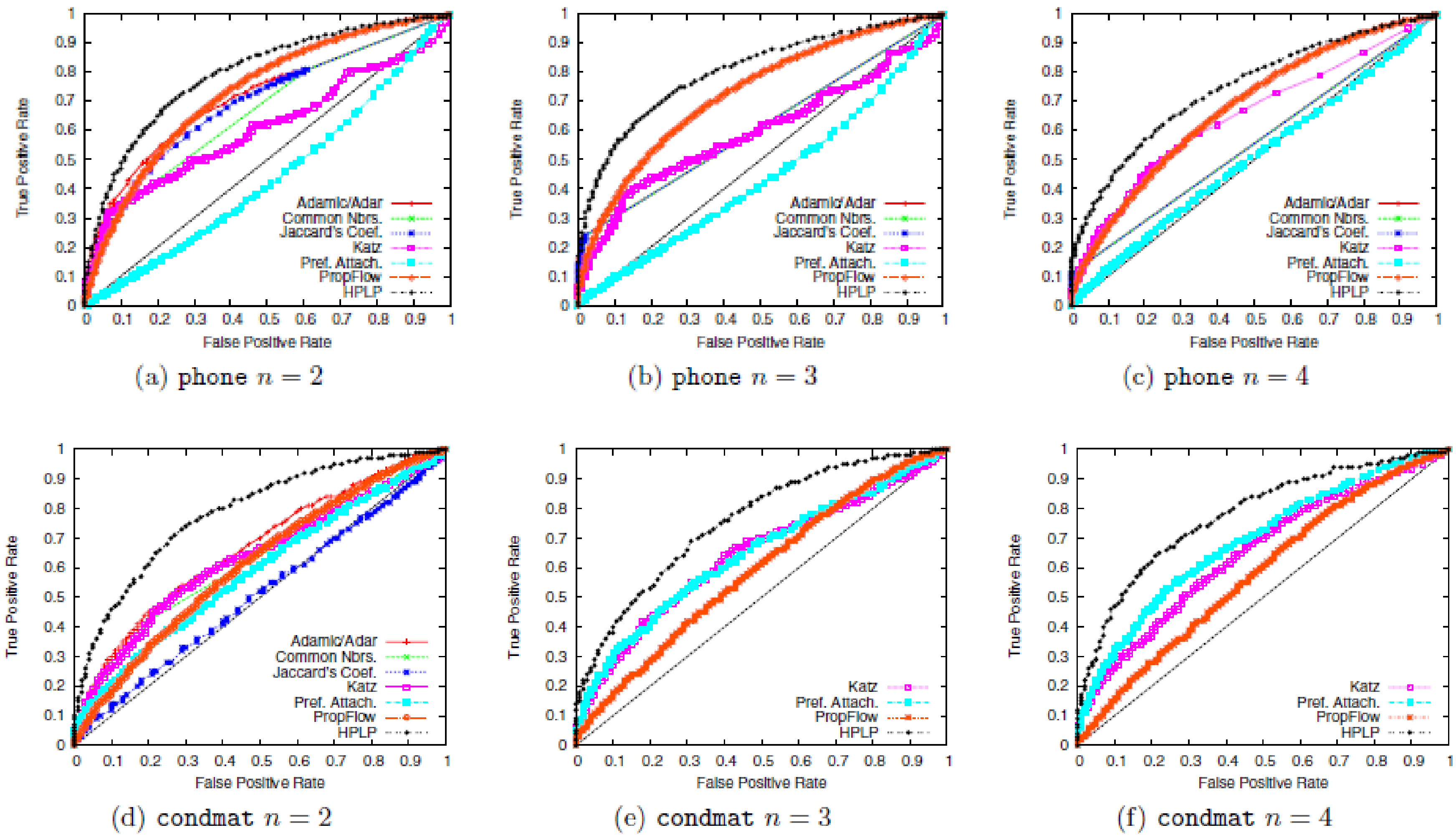
AUC: area under the ROC

Results

Ensemble of classifiers: Random Forest

Random forest: Ensemble classifier
constructs a multitude of decision trees at training time
output the class that is the mode (most frequent) of the classes (classification) or mean prediction (regression) of the individual trees.

Results



(a) phone $n = 2$ (b) phone $n = 3$ (c) phone $n = 4$

(d) condmat $n = 2$ (e) condmat $n = 3$ (f) condmat $n = 4$

Results

- Mechanism by which links arise different both across networks and geodesic distances.
- Local vs Global (preferential attachment)
  - Better in condmat network,
  - Improves with distance
- HPLP achieves performance levels as much as 30% higher than the best unsupervised methods

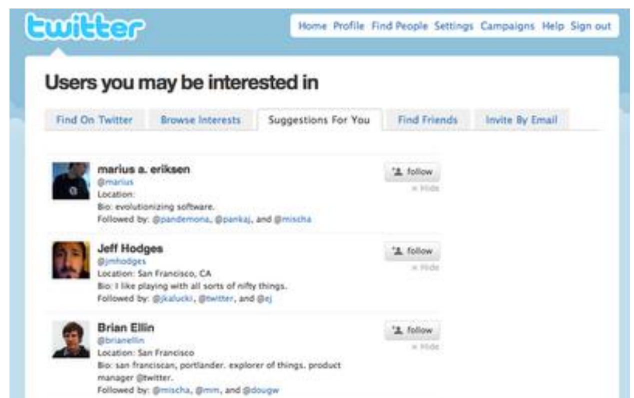
Outline

- Estimating a score for each edge (seminal work of Liben-Nowell&Kleinberg
  - Neighbors measures, distance measures, other methods
  - Evaluation
- Classification approach
  - Brief background on classification
  - Issues
- The who to follow service at Twitter

Introduction

Wtf (“Who to Follow"): *the Twitter user recommendation service*

- Twitter: 200 million users, 400 million tweets every day (as of early 2013)
http://www.internetlivestats.com/twitter-statistics/
- Twitter needs to help existing and new users to discover connections to sustain and grow
- Also used for search relevance, discovery, promoted products, etc.



History of WTF

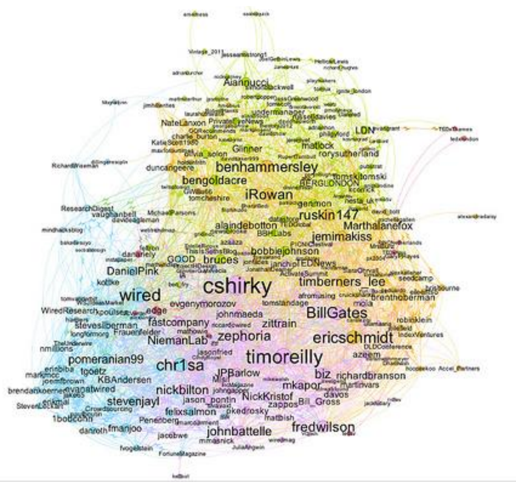
3 engineers, project started in spring 2010, product delivered in summer 2010

Basic assumption: the whole graph fits into memory of a single server

The Twitter graph

- Node: user (directed) edge: follows

- Statistics (August 2012)
  - over *20 billion edges* (only active users)
  - power law distributions of in-degrees and out-degrees.
  - over 1000 with more than 1 million followers,
  - 25 users with more than 10 million followers.



http://blog.ouseful.info/2011/07/07/visualising-twitter-friend-connections-using-gephi-an-example-using-wireduk-friends-network/

Introduction

Difference between:

- *Interested in*
- Similar to

Example (follow @espn but not similar to it)

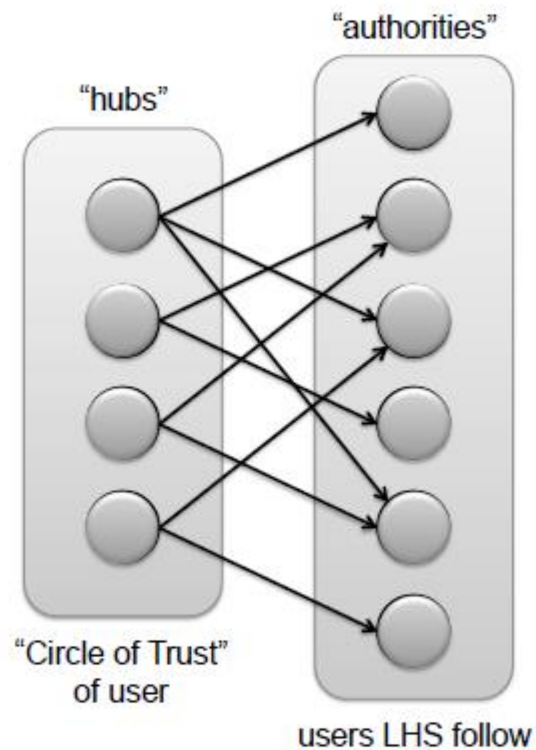
Is it a “social” network as Facebook?

Algorithms

- Asymmetric nature of the follow relationship (other social networks e.g., Facebook or LinkedIn require the consent of both participating members)

- Directed edge case is similar to the user-item recommendations problem where the “item” is also a user.

Bipartite graph



Hubs: 500 top-ranked nodes from the user's circle of trust
Authorities: users that the hubs follow.

Algorithms: Circle of trust

Circle of trust: the result of an egocentric random walk (similar to personalized (rooted) PageRank)

- Computed in an online fashion (from scratch each time) given a set of parameters (# of random walk steps, reset probability, pruning settings to discard low probability vertices, parameters to control sampling of outgoing edges at vertices with large out-degrees, etc.)

- Used in a variety of Twitter products, e.g., in search and discovery, content from users in one's circle of trust upweighted

Algorithms: SALSA

SALSA (Stochastic Approach for Link-Structure Analysis)

a variation of HITS

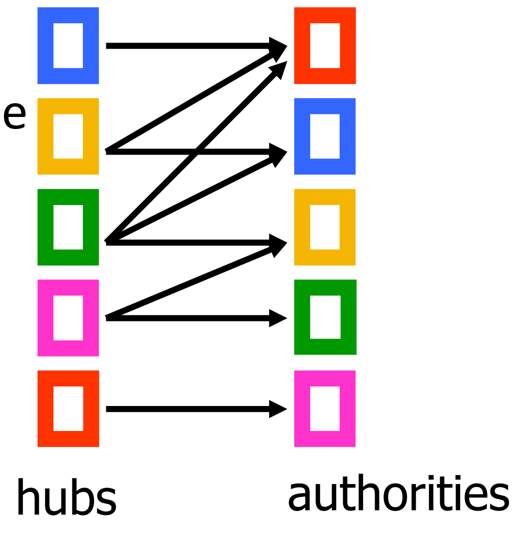
As in HITS

hubs

authorities

HITS

- Good hubs point to good authorities
- Good authorities are pointed by good hubs



hub weight = sum of the authority weights of the authorities pointed to by the hub

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

authority weight = sum of the hub weights that point to this authority.

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

Algorithms: SALSA

Random walks to rank hubs and authorities

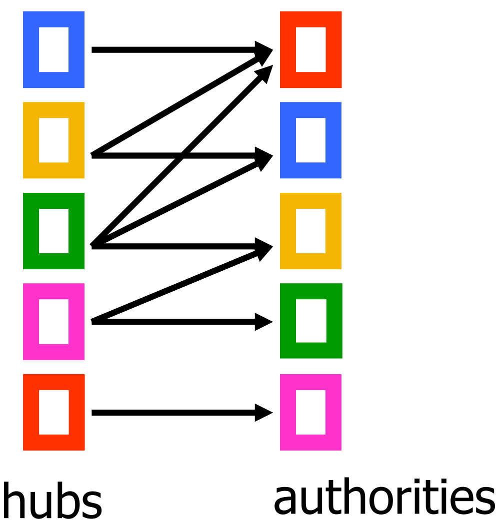
- Two different random walks (Markov chains): a chain of hubs and a chain of authorities
- Each walk traverses nodes only in one side by traversing two links in each step h->a->h, a->h->a

Transition matrices of each Markov chain: H and A

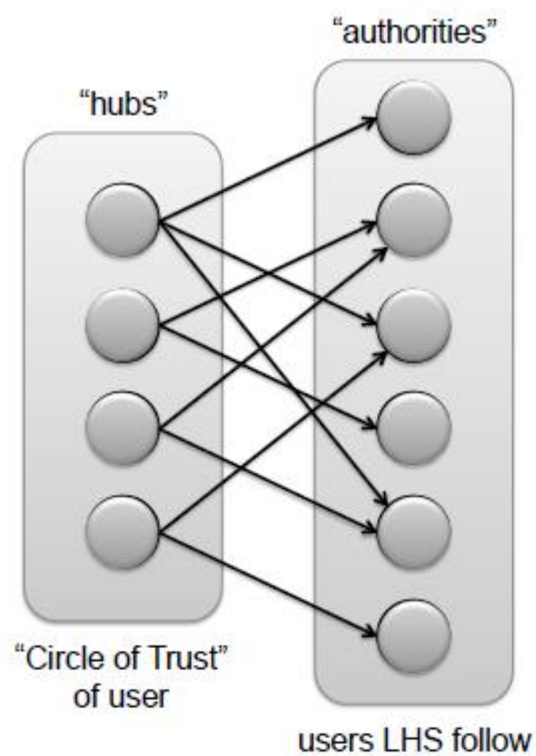
W: the adjacency of the directed graph
W_r: divide each entry by the sum of its row
W_c: divide each entry by the sum of its column

$H = W_r W_c^T$
$A = W_c^T W_r$

Proportional to the degree



Algorithms: SALSA



Hubs: 500 top-ranked nodes from the user's circle of trust
Authorities: users that the hubs follow
Use SALSA assign scores to both sides, recommend best in the RHS

Hub vertices: user similarity (based on homophily, also useful)
Authority vertices : "interested in" user recommendations.

Algorithms: SALSA

How it works

SALSA mimics the recursive nature of the problem:

- A user u is likely to follow those who are followed by users that are similar to u.
- A user is similar to u if the user follow the same (or similar) users.

I. SALSA provides *similar users* to u on the LHS and *similar followings* of those on the RHS.
II. The random walk ensures equitable distribution of scores in both directions
III. Similar users are selected from the circle of trust of the user through personalized PageRank.

Evaluation

- Offline experiments on retrospective data
- Online A/B testing on live traffic

Various parameters may interfere:

- How the results are rendered (e.g., explanations)
- Platform (mobile, etc.)
- New vs old users

Evaluation: metrics

Follow-through rate (FTR) (precision)

- Does not capture recall
- Does not capture lifecycle effects (newer users more receptive, etc.)
- Does not measure the quality of the recommendations: all follow edges are not equal

Engagement per impression (EPI):
After a recommendation is accepted, the amount of engagement by the user on that recommendation in a specified time interval called the observation interval.

Extensions

- Add *metadata to vertices* (e.g., user profile information) *and edges* (e.g., edge weights, timestamp, etc.)

- Consider *interaction graphs* (e.g., graphs defined in terms of retweets, favorites, replies, etc.)

Extensions

Two phase algorithm

- *Candidate generation*: produce a list of promising recommendations for each user, using any algorithm
- *Rescoring*: apply a machine-learned model to the candidates, binary classification problem (logistic regression)

First phase: recall + diversity
Second phase: precision + maintain diversity

References

D. Liben-Nowell, and J. Kleinberg, *The link-prediction problem for social networks.* Journal of the American Society for Information Science and Technology, 58(7) 1019–1031 (2007)

R. Lichtenwalter, J. T. Lussier, N. V. Chawla: *New perspectives and methods in link prediction*. KDD 2010: 243-252

G. Jeh, J. Widom: *SimRank: a measure of structural-context similarity*. KDD 2002: 538-543

P-N Tan, . Steinbach, V. Kumar. Introduction to Data Mining (Chapter 4)

P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, R.Zadeh. *WTF: The Who to Follow Service at Twitter,* WWW 2013

R. Lempel, S. Moran: *SALSA: the stochastic approach for link-structure analysis*. ACM Trans. Inf. Syst. 19(2): 131-160 (2001)