CSE 6242/ CX 4242

Graphs / Networks

Centrality measures, algorithms, interactive applications

Duen Horng (Polo) Chau
Georgia Tech

Partly based on materials by
Professors Guy Lebanon, Jeffrey Heer, John Stasko, Christos Faloutsos, Le Song

Recap…

- **Last time:** Basics, how to build graph, store graph, laws, etc.
- **Today:** Centrality measures, algorithms, interactive applications for visualization and recommendation

Centrality

= “Importance”

Why Node Centrality?

What can we do if we can rank all the nodes in a graph (e.g., Facebook, LinkedIn, Twitter)?

- Find **celebrities** or influential people in a social network (Twitter)
- Find "**gatekeepers**" who connect communities (headhunters love to find them on LinkedIn)
- What else?

More generally

Helps **graph analysis, visualization, understanding**, e.g.,

- Let us **rank** nodes, group or study them by centrality
- Only show subgraph formed by the **top 100 nodes**, out of the millions in the full graph
  - **Similar to google search results** (ranked, and they only show you 10 per page)
- Most graph analysis packages already have centrality algorithms implemented. **Use them!**

Can also compute edge centrality.
Here we focus on node centrality.

Degree Centrality (easiest)

Degree = number of neighbors

For directed graphs

- in degree = No. of incoming edges
- out degree = No. of outgoing edges

Algorithms?

- Sequential scan through edge list
- What about for a graph stored in SQLite?

Computing degrees using SQL

Recall simplest way to store a graph in SQLite:

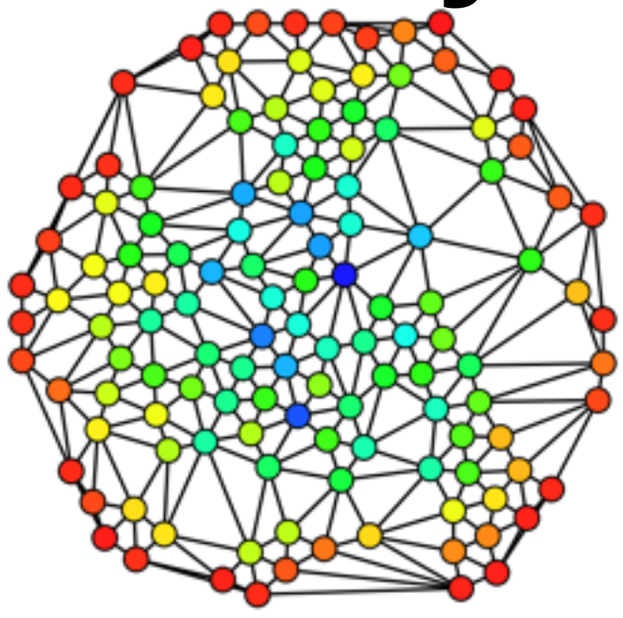
```
edges(source_id, target_id)
```

1. Create index for each column

2. Use **group by** statement to find node degrees

```
select count(*) from edges group by source_id;
```

Betweenness Centrality



High betweenness

= important “gatekeeper” or liaison

Betweenness of a node v

$$= \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

Number of shortest paths between s and t that **goes through v**

Number of shortest paths between s and t

= how often a node serves as the “bridge” that connects two other nodes.

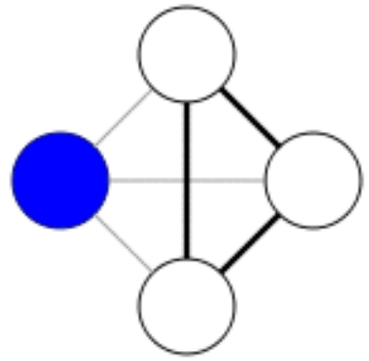
(Local) Clustering Coefficient

A node's clustering coefficient is a measure of how close the node's neighbors are from forming a clique.

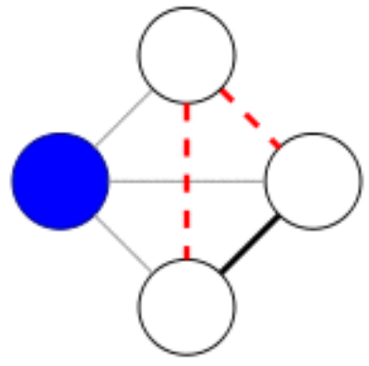
- 1 = neighbors form a clique
- 0 = No edges among neighbors

(Assuming undirected graph)

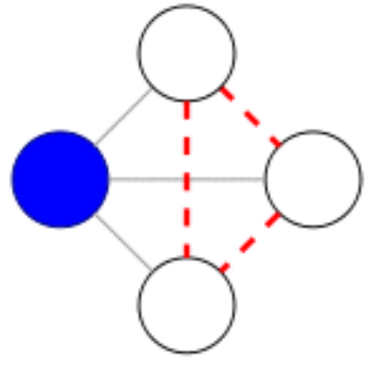
"Local" means it's for a node; can also compute a graph's "global" coefficient



c = 1



c = 1/3



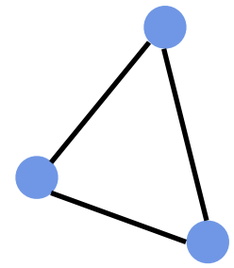
c = 0

Computing Clustering Coefficients...

Requires triangle counting

Real social networks have a lot of triangles

- Friends of friends are friends



But: triangles are expensive to compute

(3-way join; several approx. algos)

Can we do that quickly?

details

Super Fast Triangle Counting
[Tsourakakis ICDM 2008]

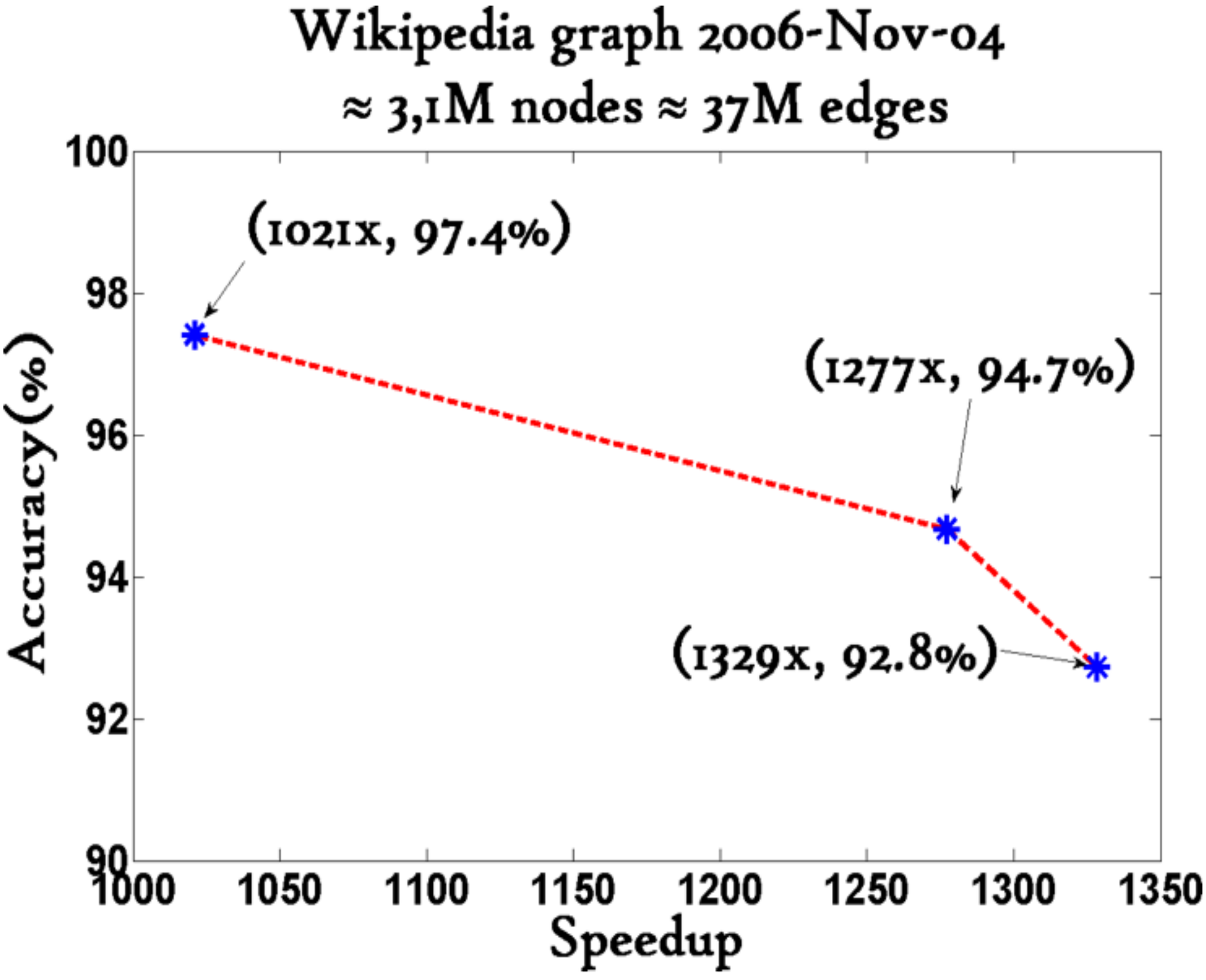
But: triangles are expensive to compute
(3-way join; several approx. algos)

Q: Can we do that quickly?

A: Yes!

#triangles = 1/6 Sum ((λi)^3)

(and, because of skewness,
we only need the top few eigenvalues!

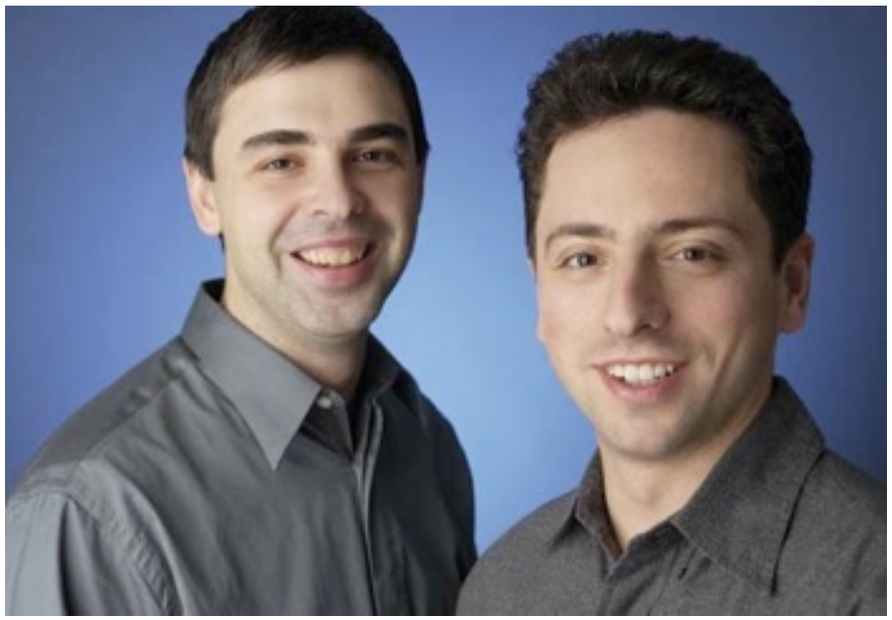


1000x+ speed-up, >90% accuracy

More Centrality Measures…

- Degree
- Betweenness
- Closeness, by computing
  - Shortest paths
  - "**Proximity**" (usually via *random walks*) — **used successfully in a lot of applications**
- Eigenvector
- …

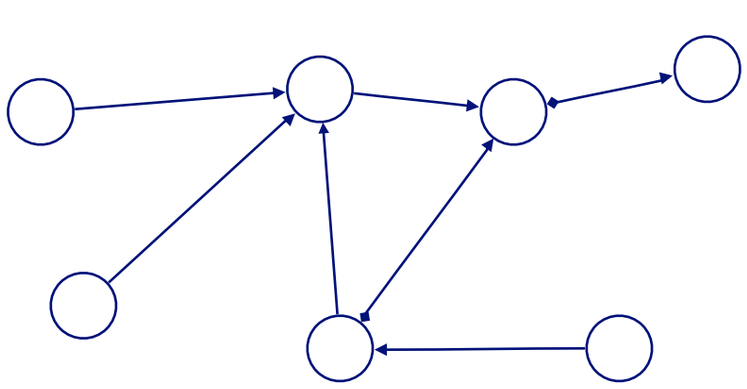
PageRank (Google)

Larry Page  Sergey Brin

Brin, Sergey and Lawrence Page (1998). *Anatomy of a Large-Scale Hypertextual Web Search Engine*. 7th Intl World Wide Web Conf.

PageRank: Problem

Given a directed graph, find its most interesting/central node

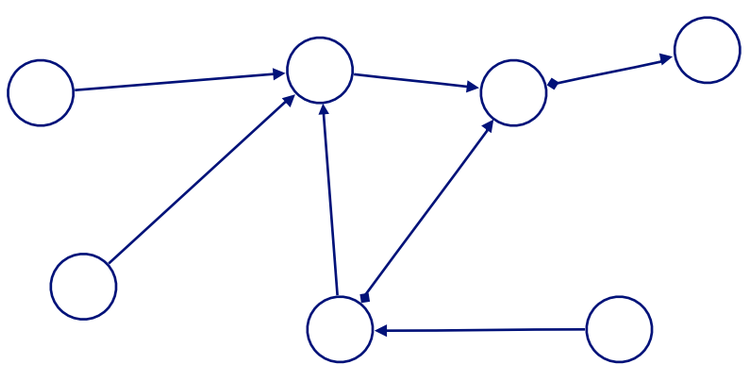


A node is important, if it is connected with important nodes (recursive, but OK!)

PageRank: Solution

Given a directed graph, find its most interesting/central node

Proposed solution:
use **random walk**; spot most 'popular' node (-> steady state probability (ssp))

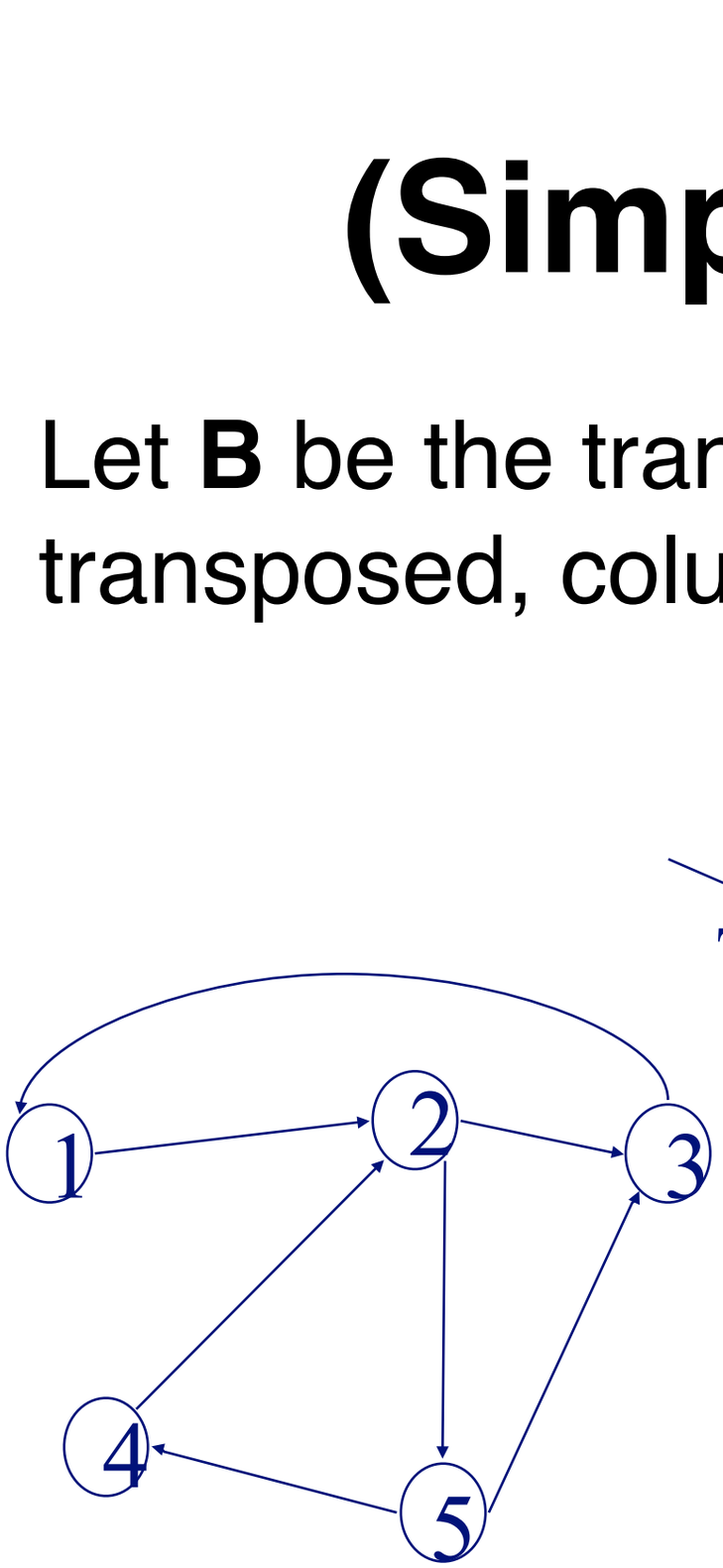


"state" = webpage

A node has high ssp, if it is connected with high ssp nodes (recursive, but OK!)

(Simplified) PageRank

Let **B** be the transition matrix:
transposed, column-normalized



From / To

B

		1		
1			1	
	1/2			1/2
				1/2
	1/2			

p1
p2
p3
p4
p5

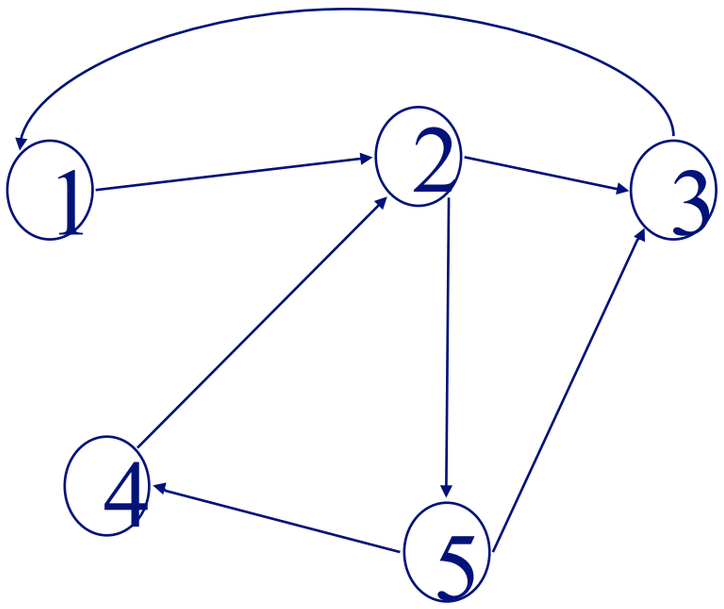
=

p1
p2
p3
p4
p5

(Simplified) PageRank

B p = p

B p = p



		1		
1			1	
	1/2			1/2
				1/2
	1/2			

p1
p2
p3
p4
p5

=

p1
p2
p3
p4
p5

(Simplified) PageRank

- $\mathbf{B}\ \mathbf{p} = 1\ ^{*}\ \mathbf{p}$
- thus, $\mathbf{p}$ is the **eigenvector** that corresponds to the highest eigenvalue (=1, since the matrix is column-normalized)
- Why does such a $\mathbf{p}$ exist?
  - $\mathbf{p}$ exists if $\mathbf{B}$ is nxn, nonnegative, irreducible [Perron–Frobenius theorem]

(Simplified) PageRank

- In short: imagine a particle randomly moving along the edges
- compute its steady-state probability (ssp)

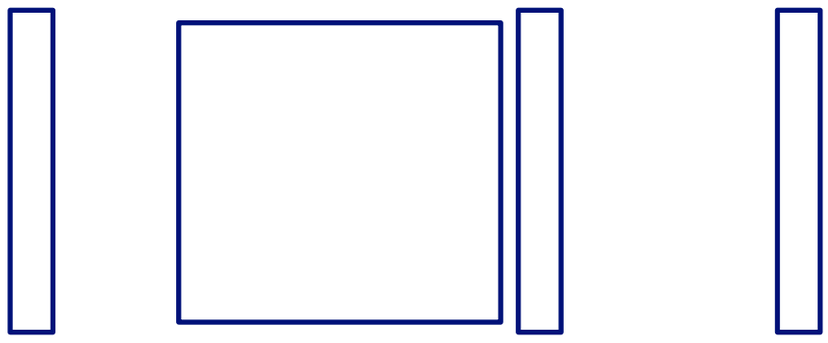
Full version of algorithm:
with occasional random jumps

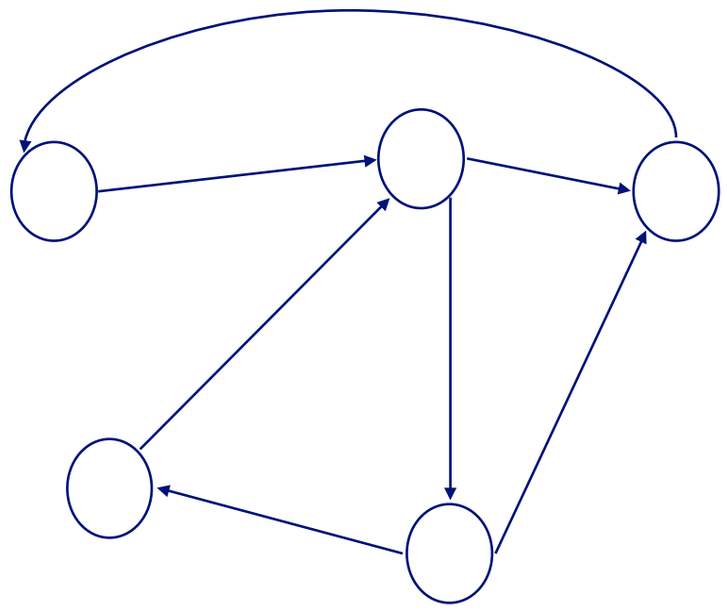
Why? To make the matrix irreducible

Full Algorithm

- With probability 1-c, fly-out to a random node
- Then, we have

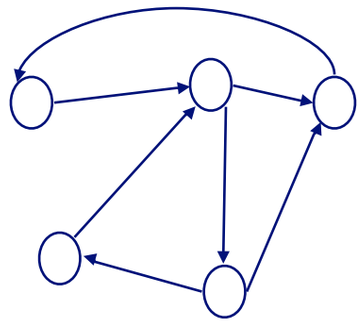
$$\mathbf{p} = c\,\mathbf{B}\,\mathbf{p} + (1-c)/n\,\mathbf{1} \Rightarrow$$





Full Algorithm

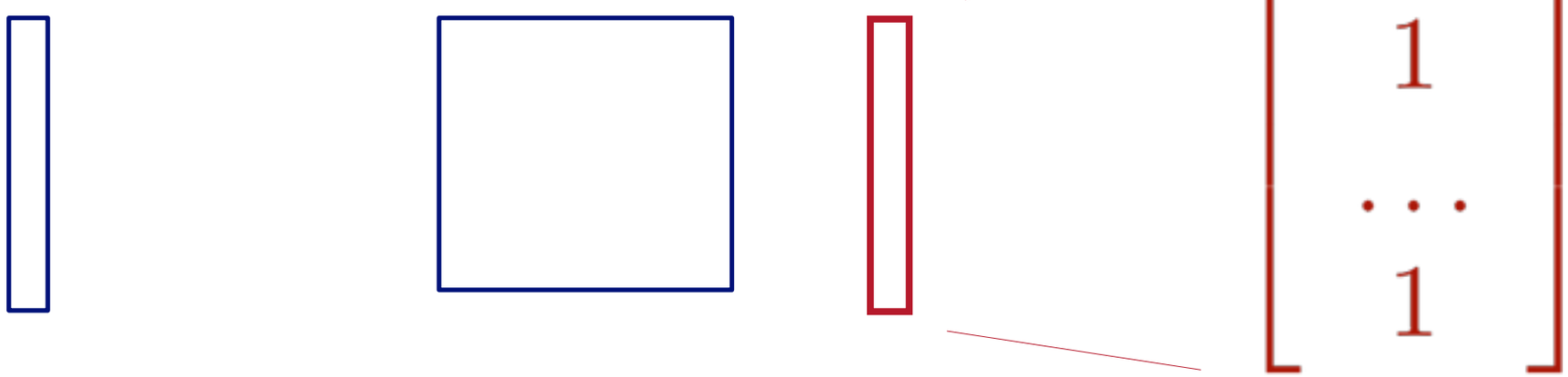
- With probability 1-c, fly-out to a random node



- Then, we have

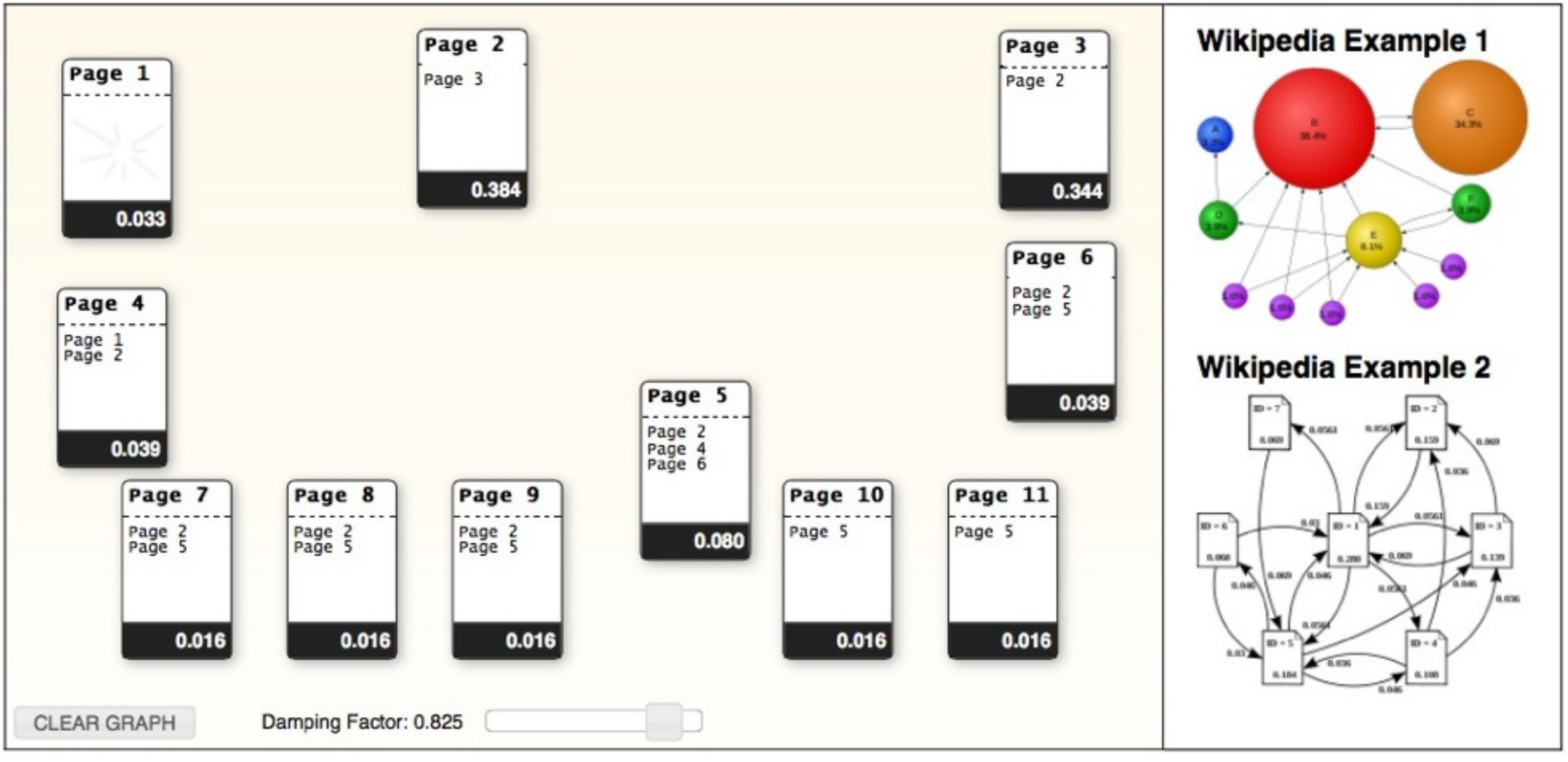
$$\mathbf{p} = c\,\mathbf{B}\,\mathbf{p} + (1-c)/n\,\mathbf{1} \Rightarrow$$

$$\mathbf{p} = (1-c)/n\,\left[\mathbf{I} - c\,\mathbf{B}\right]^{-1}\,\mathbf{1}$$



http://williamcotton.com/pagerank-explained-with-javascript

PageRank Explained with Javascript



PageRank for graphs (generally)

You can compute PageRank for **any graphs**

Should be in your algorithm “toolbox”

- Better than simple centrality measure (e.g., degree)
- Fast to compute for large graphs (O(E))

But can be “misled” (Google Bomb)

- How?

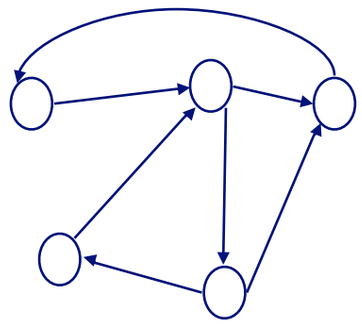
Personalized PageRank

Make one small variation of PageRank

- Intuition: not all pages are equal, some more relevant to a person's specific needs
- How?

“Personalizing” PageRank

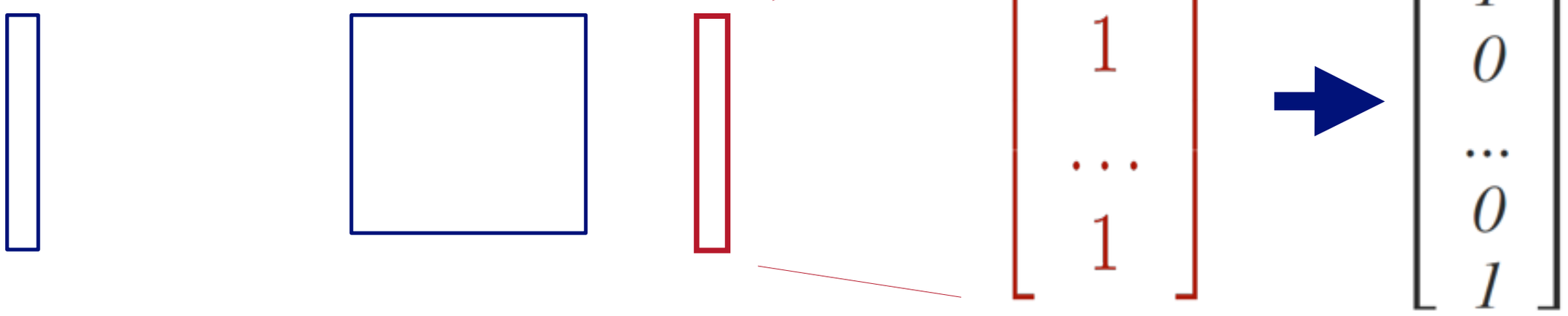
- With probability 1-c, fly-out to ~~a random node~~ **some preferred nodes**



- Then, we have

$$\mathbf{p} = c\,\mathbf{B}\,\mathbf{p} + (1-c)/n\,\mathbf{1} \Rightarrow$$

$$\mathbf{p} = (1-c)/n\,\left[\mathbf{I} - c\,\mathbf{B}\right]^{-1}\,\mathbf{1}$$



Why learn Personalized PageRank?

Can be used for **recommendation**, e.g.,

- If I like this webpage, what would I also be interested?
- If I like this product, what other products I also like? (in a user-product bipartite graph)

- Also helps with **visualizing large graphs**
  - Instead of visualizing every single nodes, visualize the **most important ones**

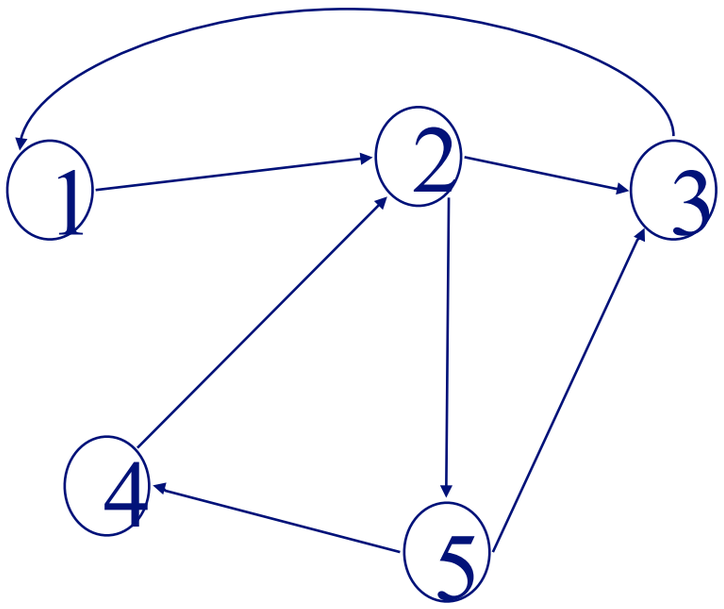
Again, very flexible. Can be run on any graph.

How to compute (Simplified) PageRank for huge matrix?

Use the power iteration method

http://en.wikipedia.org/wiki/Power_iteration

B p -> p'



		1		
1			1	
	1/2			1/2
				1/2
	1/2			

p1
p2
p3
p4
p5

->

p1
p2
p3
p4
p5

Can initialize this vector to any non-zero vector, e.g., all "1"s

Building an interactive application

Will show you an example application (Apolo) that uses a **"diffusion-based" algorithm** to perform recommendation on a large graph

- **Personalized PageRank**
  (= Random Walk with Restart)
- Belief Propagation
  (powerful inference algorithm, for fraud detection, image segmentation, error-correcting codes, etc.)
- "Spreading activation" or "degree of interest" in Human-Computer Interaction (HCI)
- Guilt-by-association techniques

Building an interactive application

Why diffusion-based algorithms are widely used?

- **Intuitive to interpret**
  uses “network effect”, homophily, etc.
- **Easy to implement**
  Math is relatively simple
- **Fast**
  run time linear to #edges, or better
- **Probabilistic** meaning

Human-In-The-Loop Graph Mining

Apolo:

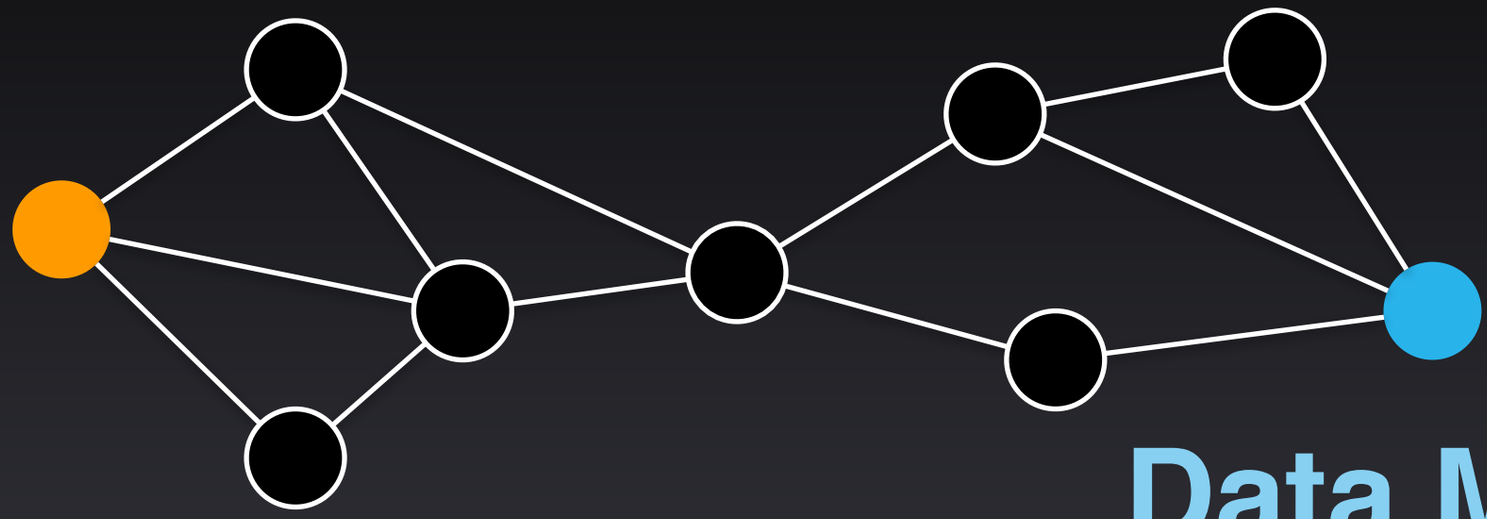
Machine Learning + Visualization

CHI 2011

Apolo: Making Sense of Large Network Data by Combining Rich User Interaction and Machine Learning

Finding More Relevant Nodes

HCI
Paper



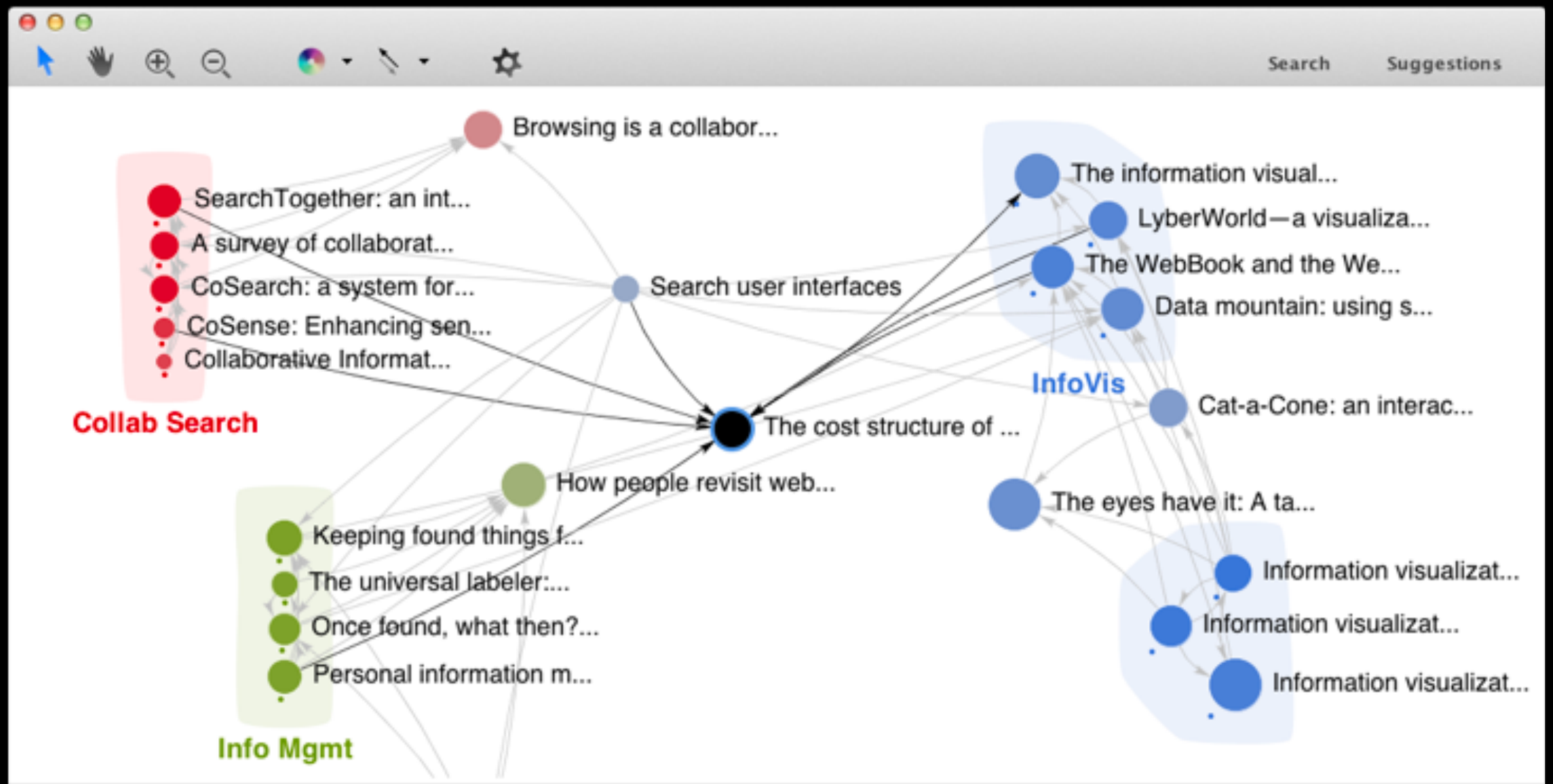
Data Mining
Paper

Citation network

Apolo uses **guilt-by-association**
(Belief Propagation, similar to personalized PageRank)

Demo: Mapping the Sensemaking Literature

Nodes: 80k papers from Google Scholar (node size: #citation)
Edges: 150k citations



Search | Suggestions

For **The cost structure of sensemaking**

The information visualizer, an inf...	1991	
Card, S.K. and Robertson, G.G. and Macki...	532	
The WebBook and the Web Forag...	1996	
Card, S.K. and Robertson, G.G. and York, W.	403	
LyberWorld—a visualization user...	1994	
Hemmje, M. and Kunkel, C. and Willett, A.	223	
The structure of the information...	1997	
Card, S.K. and Mackinlay, J.	198	
Information visualization	2009	
Card, S. and Mackinlay, JD and Shneiderm...	180	
"I"ll get that off the audio": a cas...	1997	
Moran, T.P. and Palen, L. and Harrison, S....	143	
An organic user interface for sear...	1995	
Mackinlay, J.D. and Rao, R. and Card, S.K.	123	
Using a landscape metaphor to re...	1993	
Chalmers, M.	122	
Personal information management	2007	
Jones, W.P. and Teevan, J.	109	
SearchTogether: an interface for c...	2007	
Morris, M.R. and Horvitz, E.	108	
Information foraging theory: Ada...	2007	
Pirolli, P.	107	
Investigating behavioral variabilit...	2007	
White, R.W. and Drucker, S.M.	79	
Jigsaw: Supporting investigative...	2008	
Stasko, J. and Görg, C. and Liu, Z.	71	
The cost-of-knowledge character...	1994	
Card, S.K. and Pirolli, P. and Mackinlay, J.D.	54	
Collaborative conceptual design:...	1996	
Potts, C. and Catledge, L.	45	

The cost structure of sen...

The cost structure of sensemaking PDF 1993

Russell, D.M. and Stefik, M.J. and Pirolli, P. and Card, S.K.

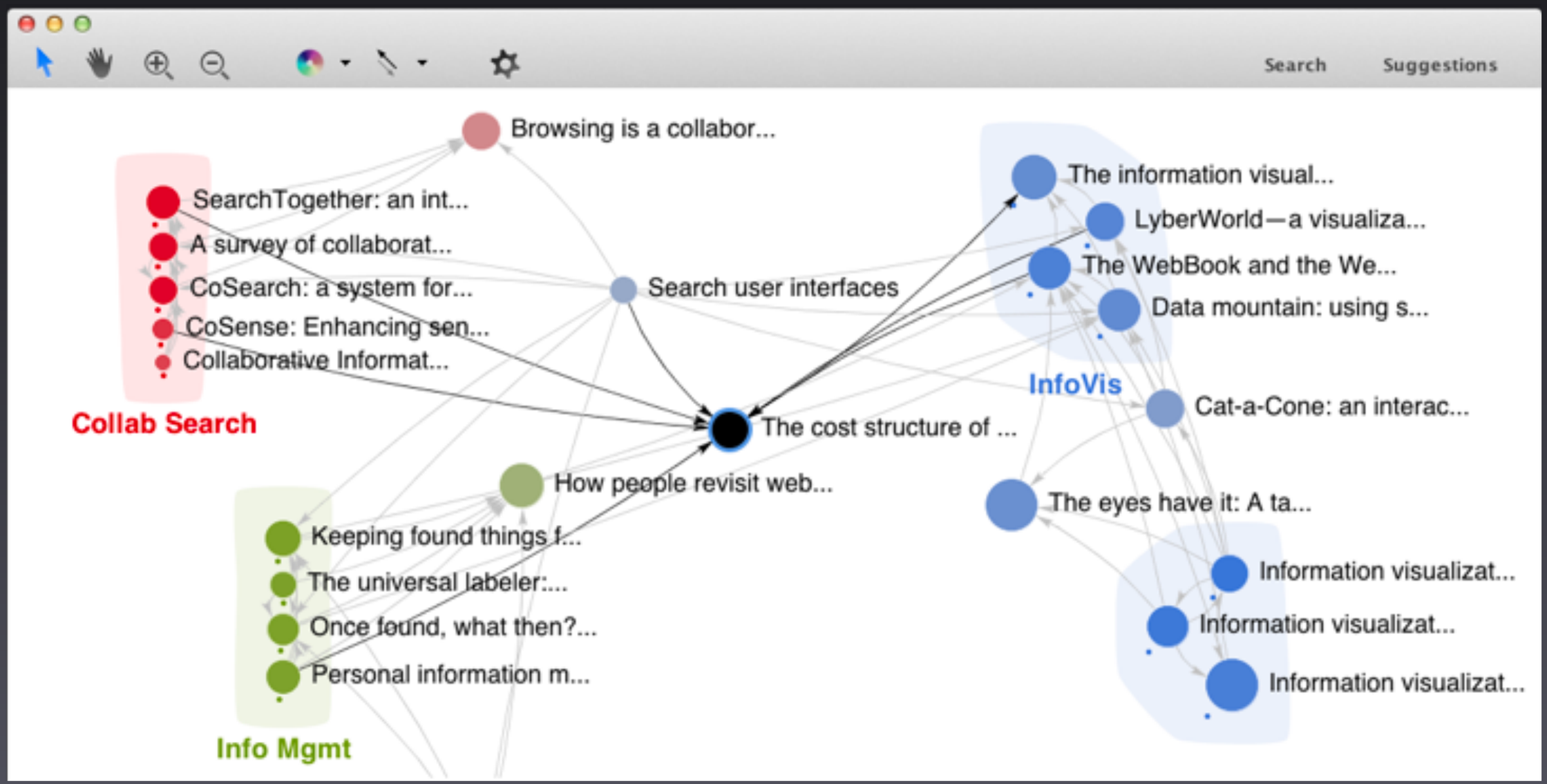
245 citations 8 versions

googlescholar.db

Key Ideas (Recap)

Specify exemplars

Find other relevant nodes (BP)



Apolo's Contributions

1 Human + Machine

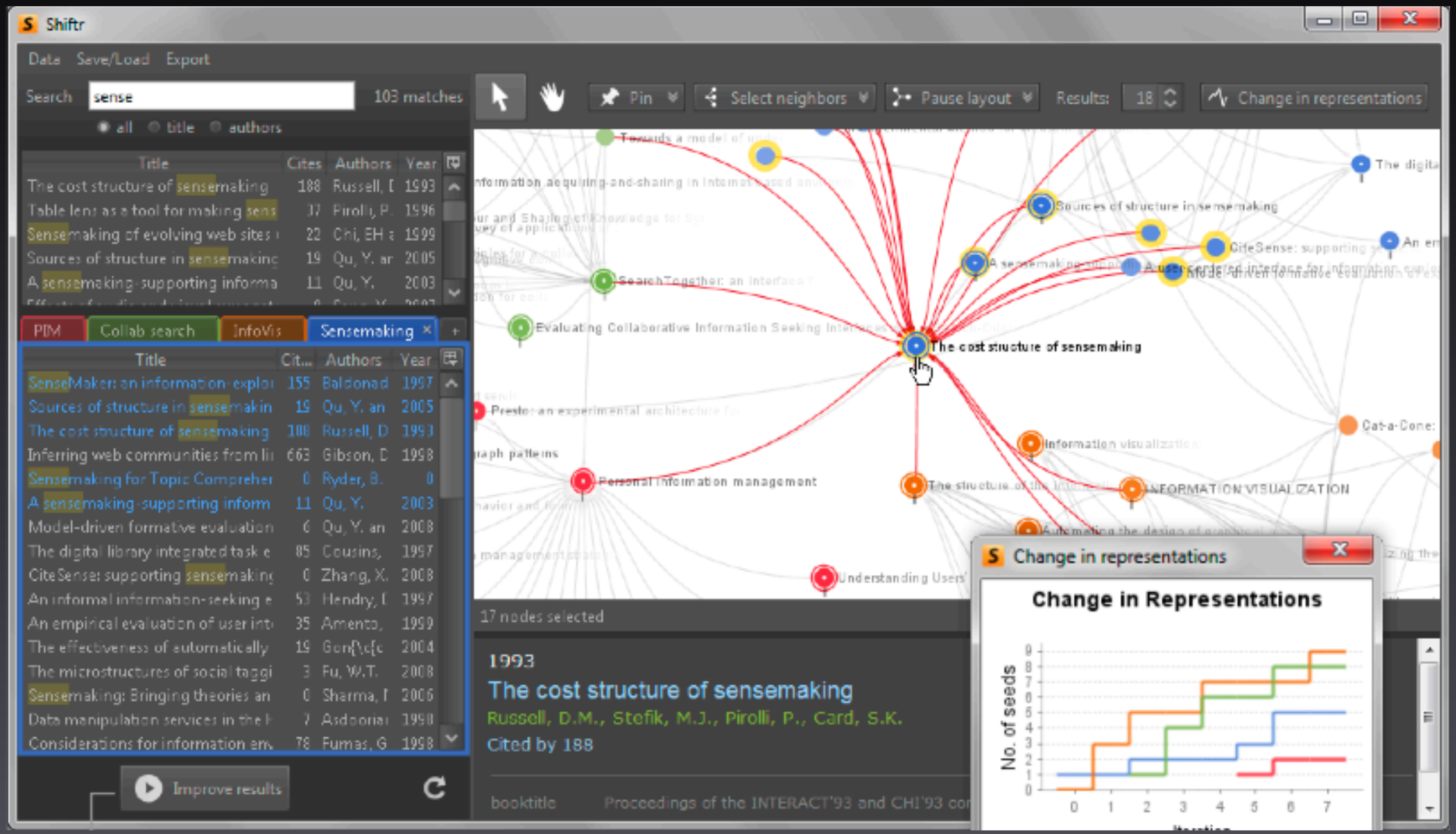
It was like having a **partnership** with the machine.

2 Personalized Landscape

Apolo 2009

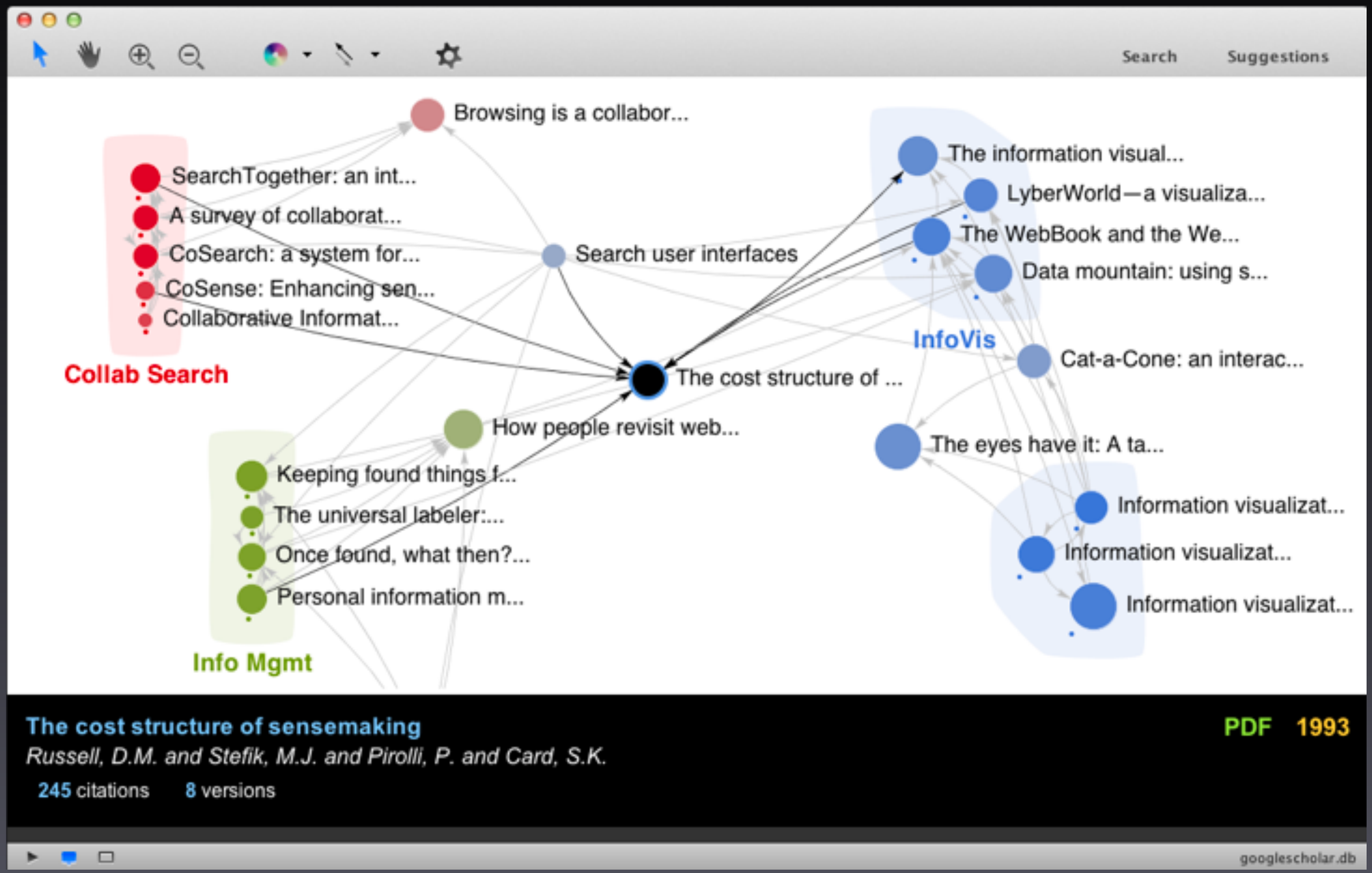

Cluster Data
Add Group
Recommendations:
End User Programming
End users creating effective softw...
End user software engineering: chi...
Invited research overview: end-us...
Brad A. Myers
Margaret M. Burnett
Mary Beth Rosson
Andrew Jensen Ko
Alan F. Blackwell
Show: All
Not Interested
Automatically generating user inte...
Decision-Theoretic User Interface ...
Daniel S. Weld
Krzysztof Z. Gajos
Automatically generating
Exploring the design spac
Predictability and accuracy
Planning-Based Control of
Brad
Brad A. Myers
The garnet user interface developm...
Using HCI Techniques to Design a M...
Creating charts by demonstration.
The Amulet User Interface Developm...
Easily Adding Animations to Interfac...
Simplifying video editng using metad...
SILVER: simplifying video editing wit...
Text Entry
In-stroke word completion.
Integrating isometric joysticks into...
Eyes on the road, hands on the whe...
An alternative to push, press, and t...
Maximizing the guessability of symb...
Few-key text entry revisited: mnem...
Text entry from power wheelchairs: ...
Joystick text entry with date stamp, ...
Show: Papers
Interface Generation
Huddle: automatically generating i...
UNIFORM: automatically generatin...
Demonstrating the viability of auto...
Jeffrey Nichols
Brandon Rothrock
Duen Horng Chau

Apolo 2010


Shiftr
Data
Save/Load
Export
Search
sense
103 matches
all
title
authors
Title
Cites
Authors
Year
The cost structure of sensemaking 188 Russell, D 1993
Table lens as a tool for making sens 37 Pirolli, P 1996
Sensemaking of evolving web sites 22 Chi, EH a 1999
Sources of structure in sensemaking 19 Qu, Y. ar 2005
A sensemaking-supporting informa 11 Qu, Y. 2003
PIM
Collab search
InfoVis
Sensemaking
Title
Cit...
Authors
Year
SenseMaker: an information-explor 155 Baldonad 1997
Sources of structure in sensemakin 19 Qu, Y. an 2005
The cost structure of sensemaking 188 Russell, D 1993
Inferring web communities from li 663 Gibson, D 1998
Sensemaking for Topic Compreher 0 Ryder, B. 0
A sensemaking-supporting inform 11 Qu, Y. 2003
Model-driven formative evaluation 6 Qu, Y. an 2008
The digital library integrated task e 85 Cousins, 1997
CiteSense: supporting sensemaking 0 Zhang, X. 2008
An informal information-seeking e 53 Hendry, D 1997
An empirical evaluation of user int 35 Amento, 1999
The effectiveness of automatically 19 Gon{\c{c 2004
The microstructures of social taggi 3 Fu, W.T. 2008
Sensemaking: Bringing theories an 0 Sharma, P 2006
Data manipulation services in the I 7 Asdoorian 1998
Considerations for information en 78 Fumas, G 1998
Improve results
Pin
Select neighbors
Pause layout
Results:
18
Change in representations
Towards a model of
The digita
Sources of structure in sensemaking
CiteSense: supporting
An em
SearchTogether: an interface
Evaluating Collaborative Information Seeking Interfaces
The cost structure of sensemaking
Presto: an experimental architecture for
Cat-a-Cone:
Information visualization
Personal Information management
INFORMATION VISUALIZATION
Understanding Users'
17 nodes selected
1993
The cost structure of sensemaking
Russell, D.M., Stefik, M.J., Pirolli, P., Card, S.K.
Cited by 188
booktitle
Proceedings of the INTERACT'93 and CHI'93 con
Change in representations
Change in Representations
No. of seeds
0
1
2
3
4
5
6
7
8
9

Apolo 2011

22,000 lines of code. Java 1.6. Swing.
Uses SQLite3 to store graph on disk

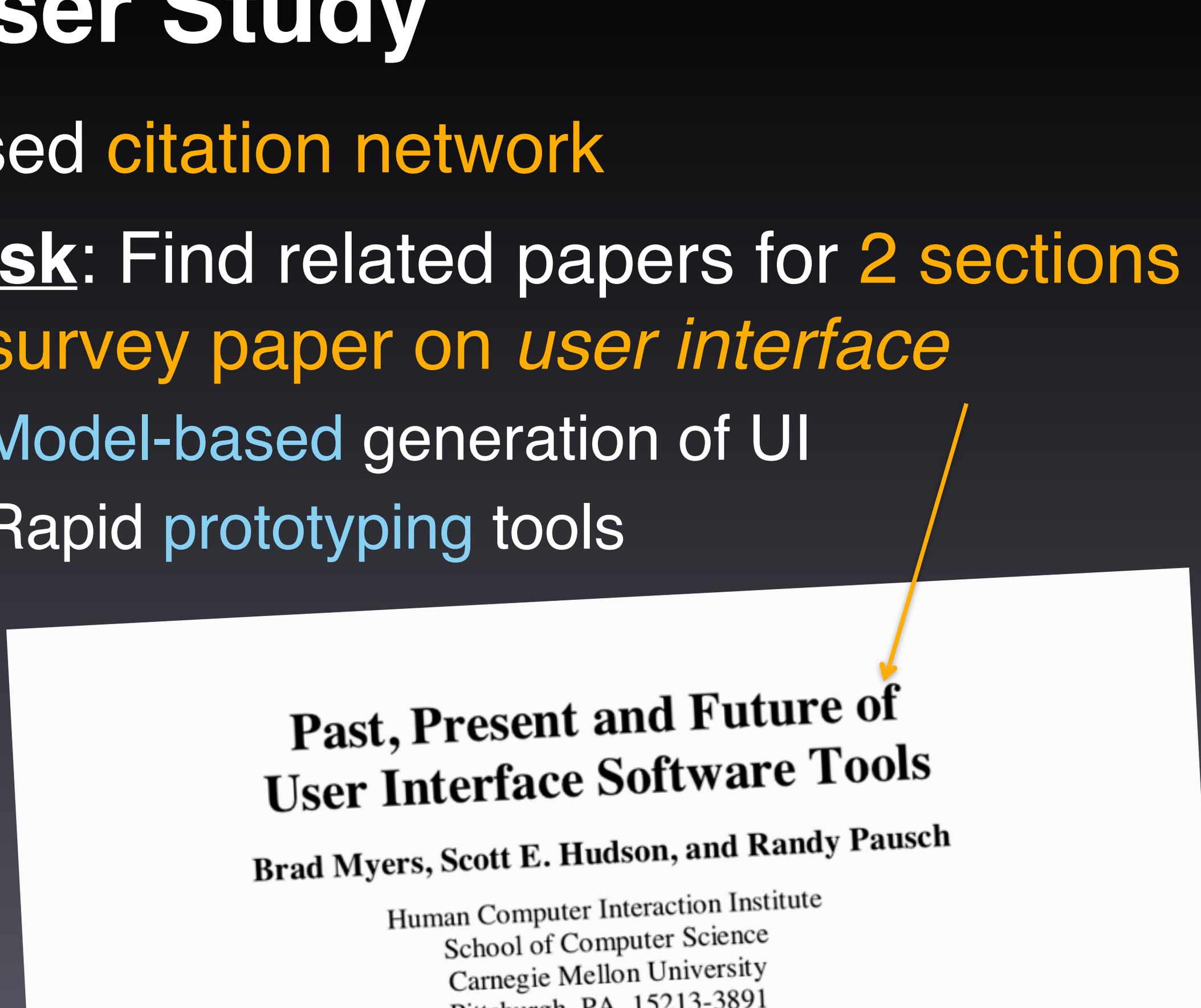


User Study

Used citation network

Task: Find related papers for 2 sections in a survey paper on *user interface*

- Model-based generation of UI
- Rapid prototyping tools



Past, Present and Future of User Interface Software Tools

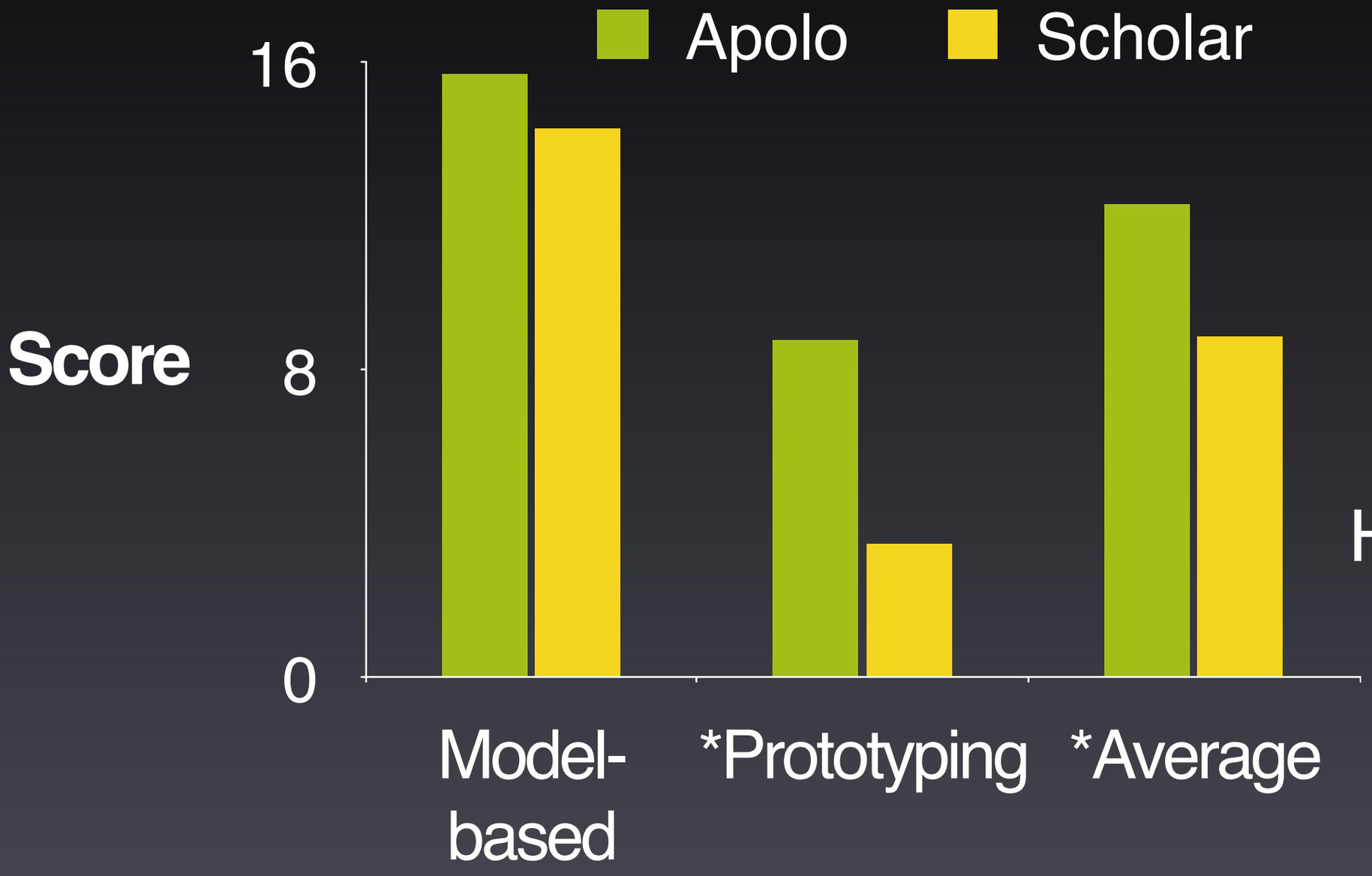
Brad Myers, Scott E. Hudson, and Randy Pausch

Human Computer Interaction Institute
School of Computer Science
Carnegie Mellon University

Between subjects design

Participants: **grad student** or **research staff**

Judges' Scores



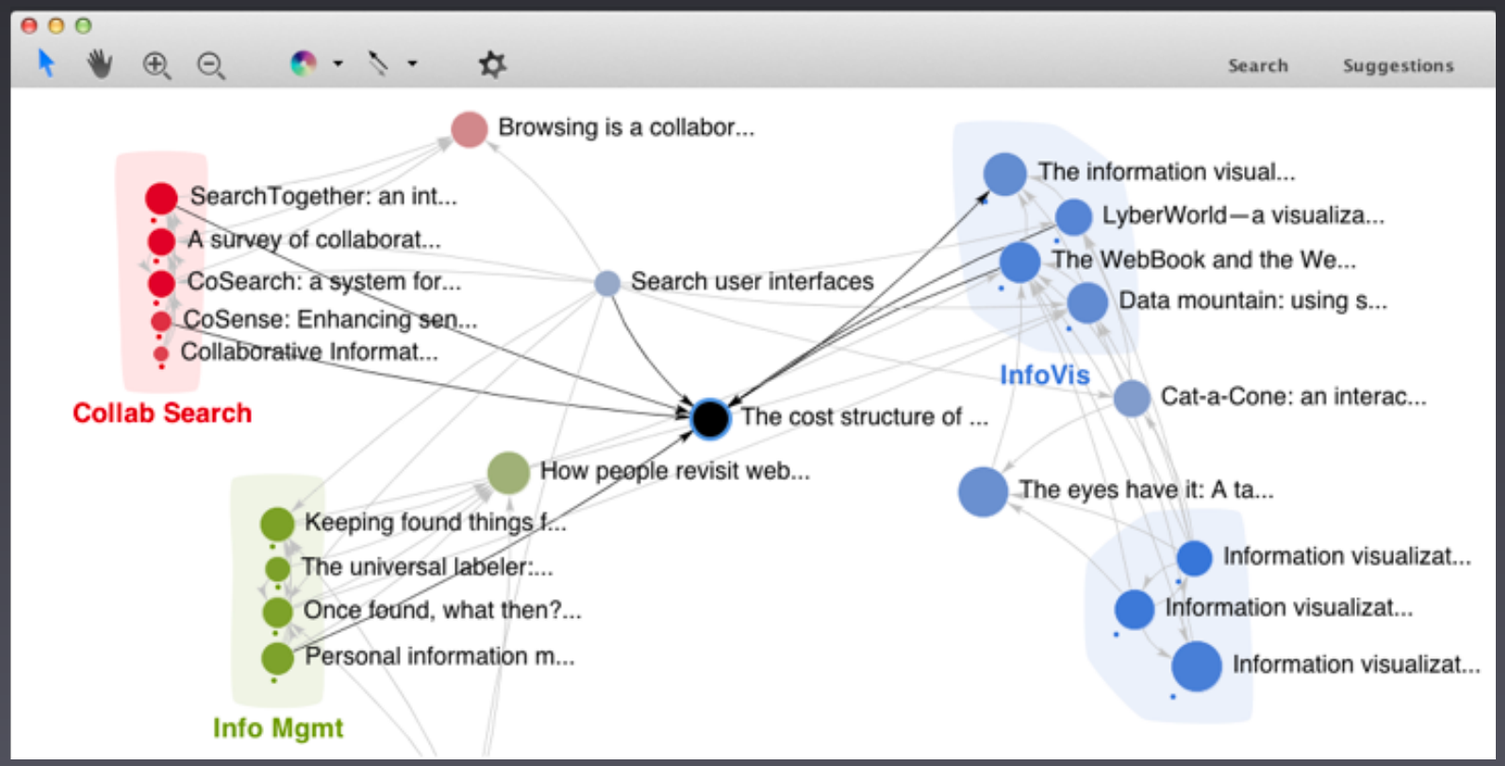
Higher is better.
Apolo wins.

** Statistically significant, by two-tailed t test, p <0.05*

Apolo: Recap

A mixed-initiative approach for exploring and creating personalized landscape for large network data

Apolo = ML + Visualization + Interaction



Practitioners' guide to building (interactive) applications

Think about scalability early

- e.g., picking a scalable algorithm early on

When building interactive applications, use **iterative** design approach (as in Apolo)

- Why? It's hard to get it right the first time
- Create prototype, evaluate, modify prototype, evaluate, ...
- Quick evaluation helps you identify **important fixes early** (can **save you a lot of time**)

Practitioners' guide to building (interactive) applications

How to do **iterative** design?

What kinds of **prototypes**?

- Paper prototype, lo-fi prototype, high-fi prototype

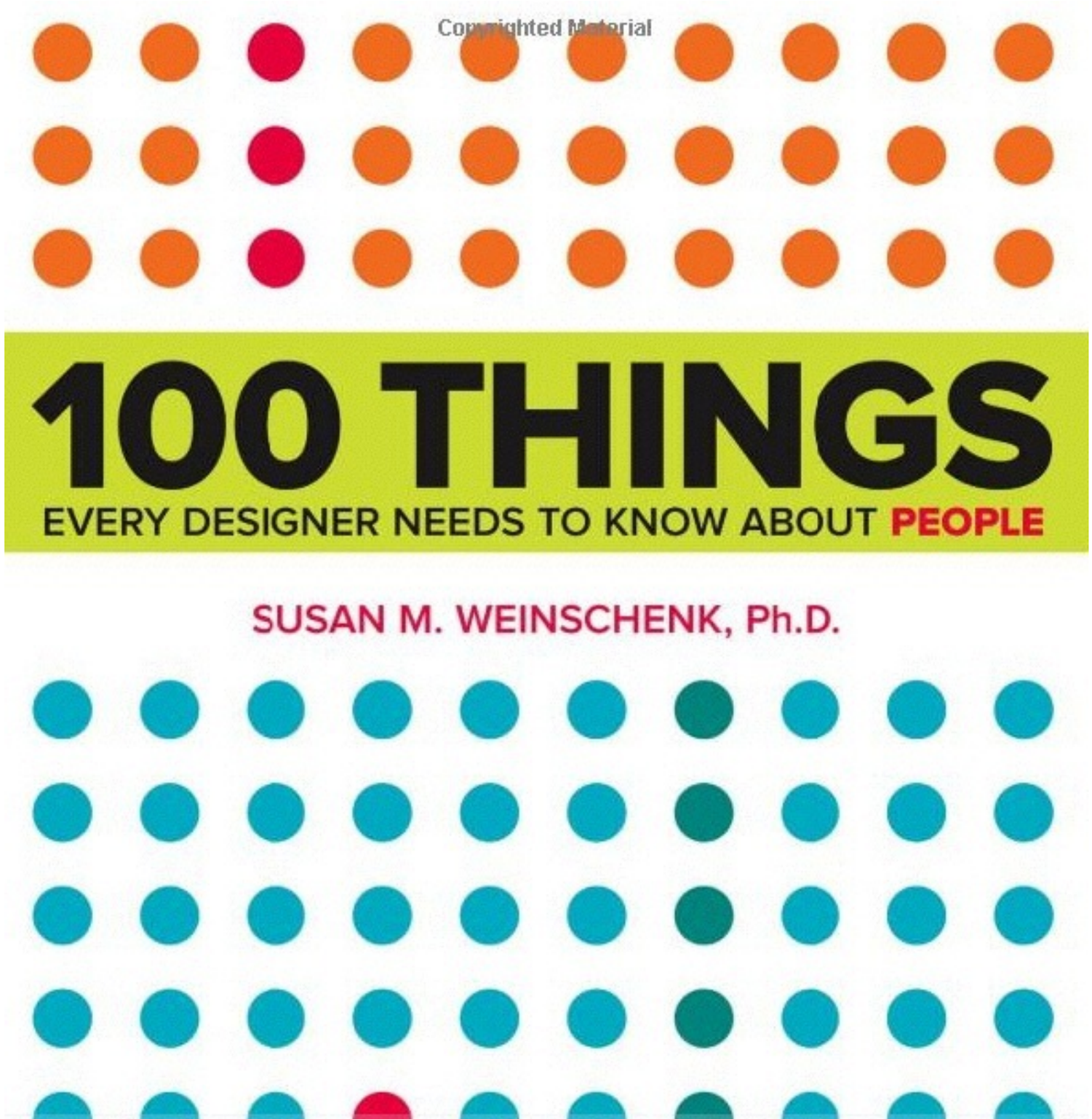
What kinds of **evaluation**? Important to involve **REAL users** as early as possible

- Recruit your friends to try your tools
- Lab study (controlled, as in Apolo)
- Longitudinal study (usage over months)
- Deploy it and see the world's reaction!

- To learn more:
  - CS 6750 Human-Computer Interaction
  - CS 6455 User Interface Design and Evaluation

If you want to know more about people…

http://amzn.com/0321767535



Polonium:
Web-Scale Malware Detection

SDM 2011

Polonium: Tera-Scale Graph Mining and Inference for Malware Detection

Typical Malware Detection Method

Signature-based detection

1. Collect malware
2. Generate **signatures**
3. Distribute to users
4. Scan computers for matches

What about "zero-day" malware?

No samples → No signatures → No detection

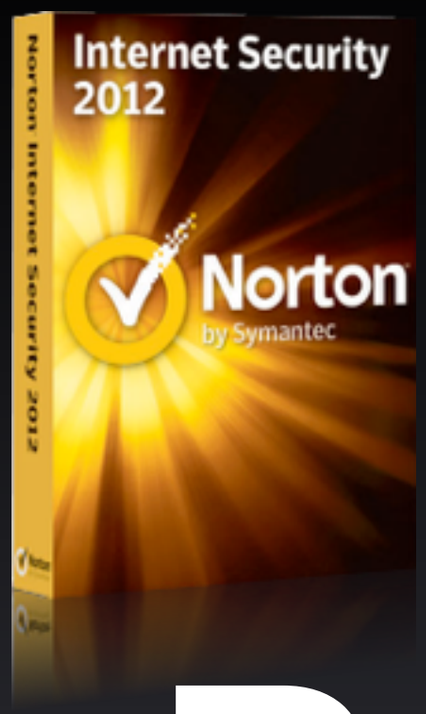
How to detect them early?

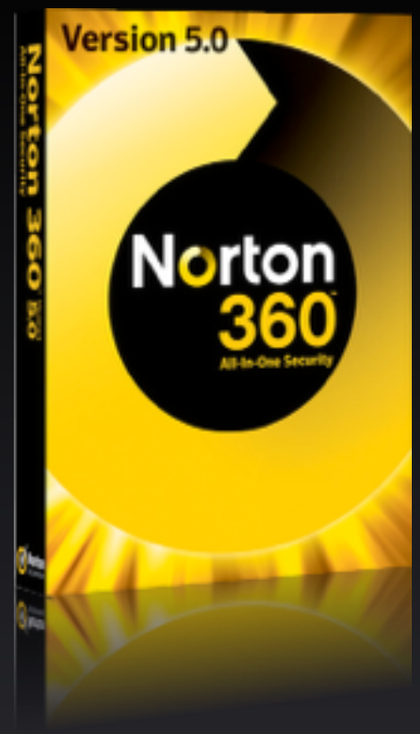
Reputation-Based Detection

Computes **reputation score** for each application

e.g., MSWord.exe

Poor reputation = Malware





Polonium

Propagation of leverage of network influence unearths malware

Patented

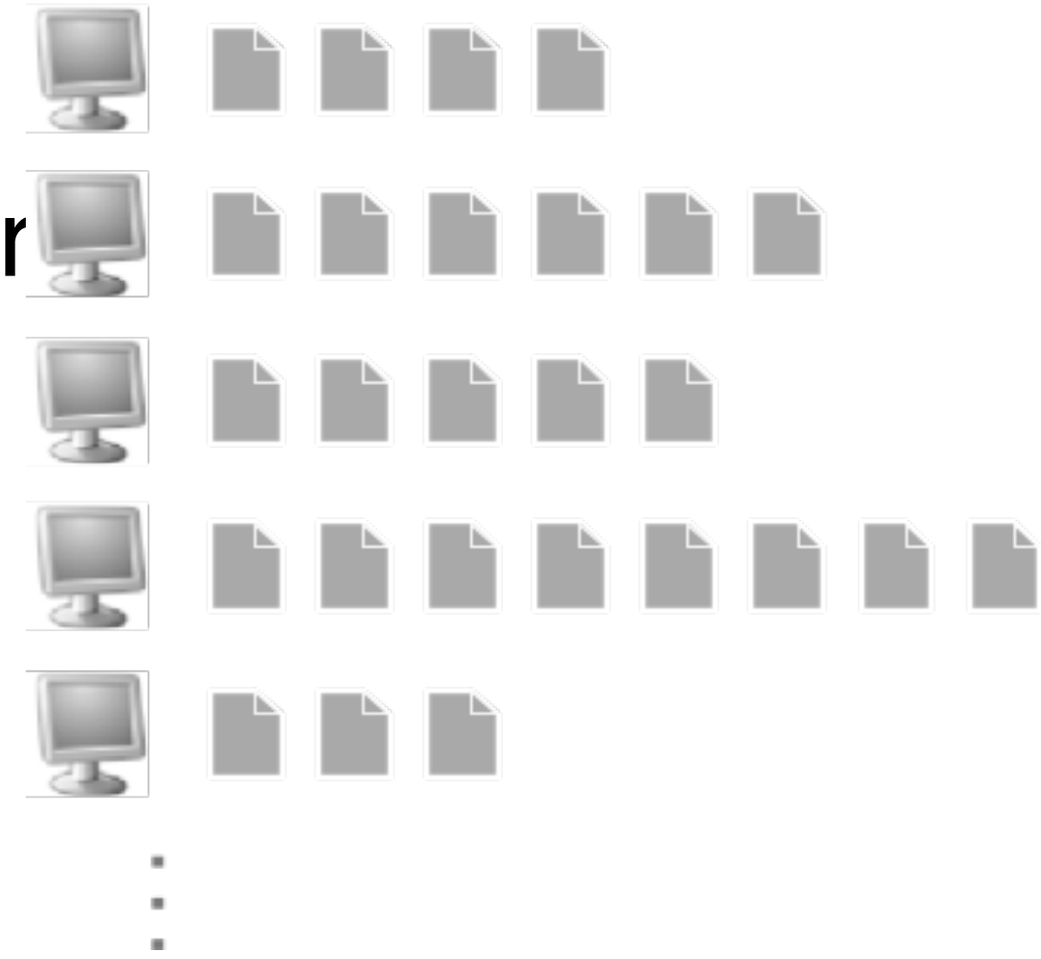
I led initial **design** and **development**

Serving **120 million** users

Answered **trillions** of queries

Polonium works with 60 Terabyte Data

50 million machines
anonymously reported their
executable **files**



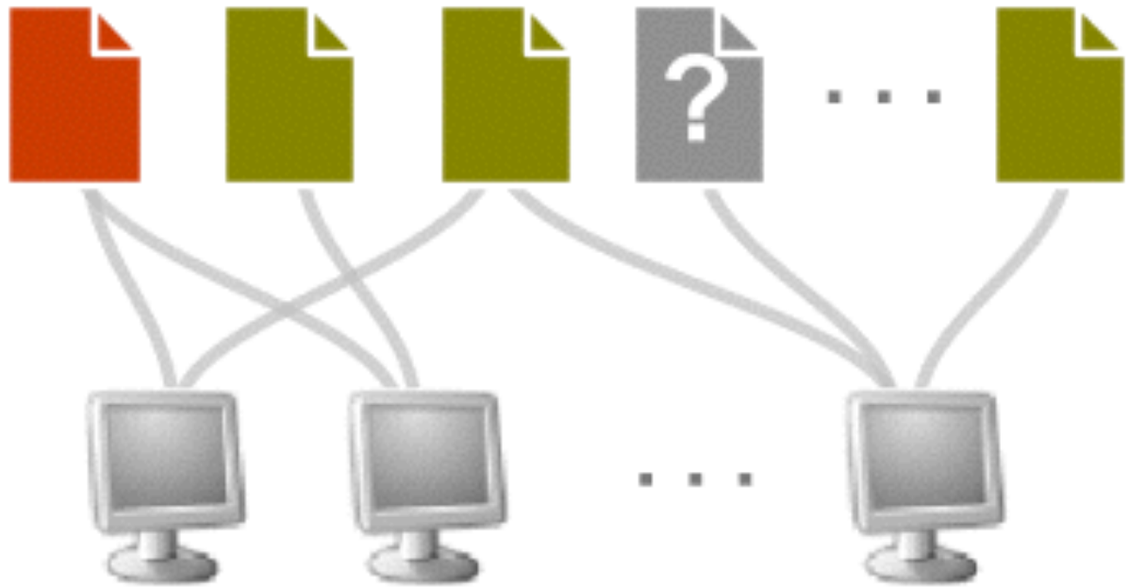
900 million unique files
(Identified by their
cryptographic hash values)

Goal: label **malware** and **good** files

Why A Hard Problem?

Existing Research	Polonium
Small dataset	**Huge** dataset (60 terabytes)
Detects **specific** malware (e.g., worm, trojans)	Detects **all** types (needs a general method)
Many false alarms (>10%)	**Strict** (<1%)

Polonium: Problem Definition



Given

Undirected machine-file bipartite graph

37 billion edges , **1 billion** nodes (machines, files)

Some file labels from Symantec (**good** or **bad**)

Find

Labels for all **unknown** files

Where to Get Good and Bad Labels?

Symantec has a **ground truth database** of **known-good** and **known-bad** files

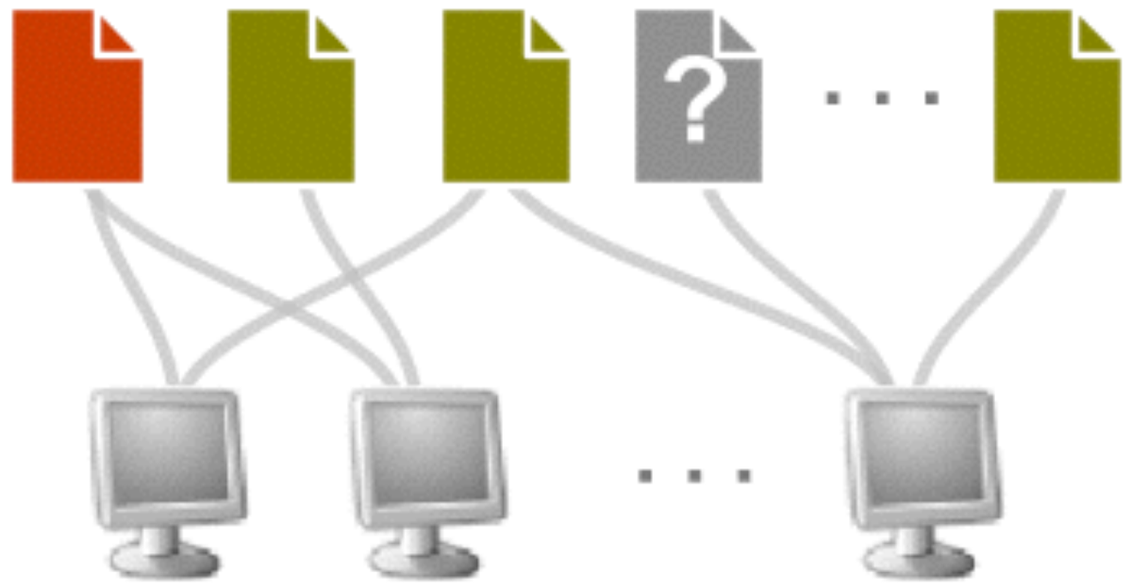
e.g., set **known-good** file's prior to **0.9**

How to Gauge Machine Reputation?

Computed using Symantec's proprietary formula;
a value between 0 and 1

Derived from anonymous aspects of machine's usage and behavior

How to propagate known information to the unknown?



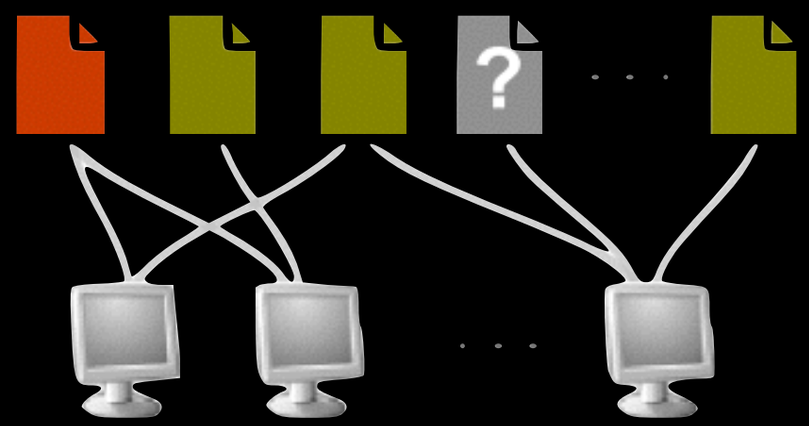
Key Idea: Guilt-by-Association

GOOD files likely appear on **GOOD machines**

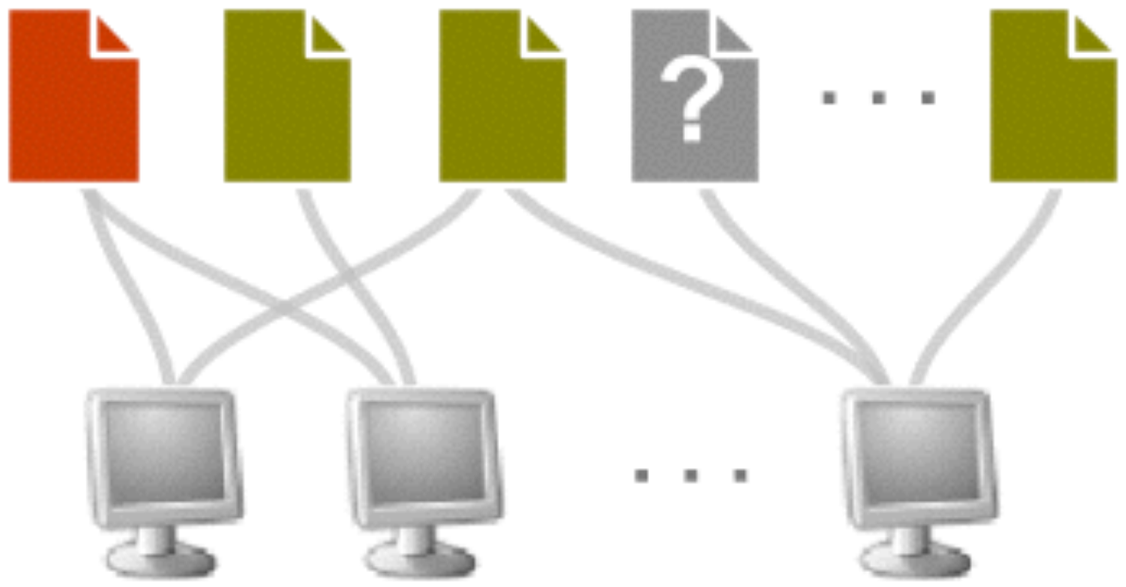
BAD files likely appear on **BAD machines**

Also known as **Homophily**

File \ Machine	Good	Bad
Good	0.9	0.1
Bad	0.1	0.9



How to propagate known information to the unknown?



Adapts **Belief Propagation (BP)**

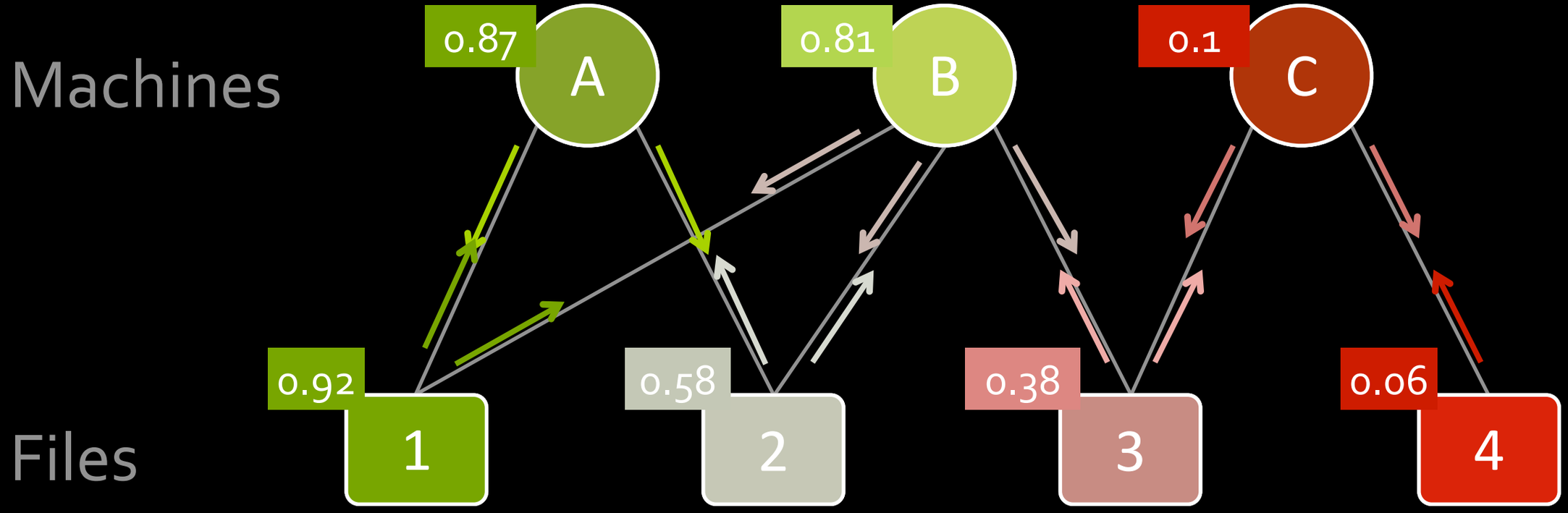
A powerful **inference** algorithm

Used in image processing, computer vision, error-correcting codes, etc.

Example

Propagating Reputation

File \ Machine	Good	Bad
Good	0.9	0.1
Bad	0.1	0.9



Two Equations in Belief Propagation

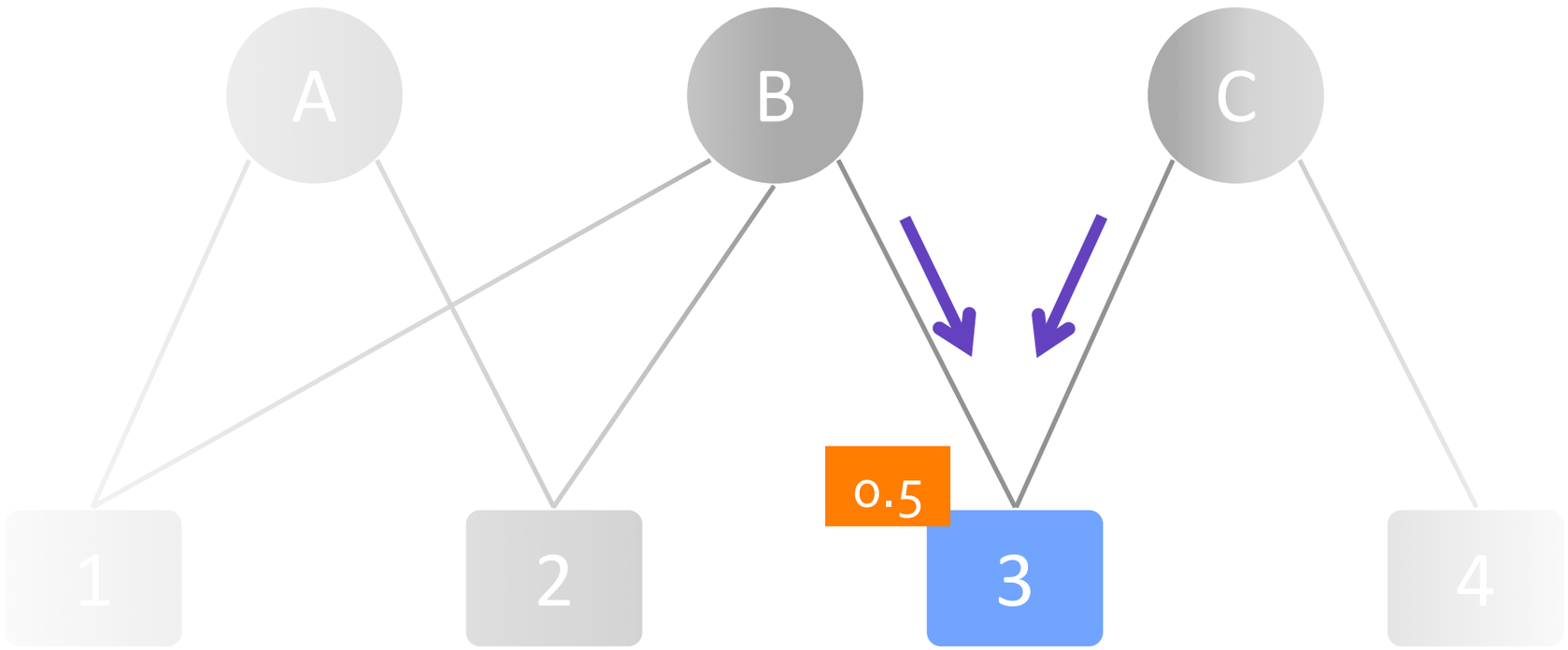
Details

$$b_i(x_i) = k\phi(x_i) \prod_{x_j \in N(i)} m_{ji}(x_i)$$

$$m_{ij}(x_j) \leftarrow \sum_{x_i \in X} \psi_{ij}(x_i, x_j)\ \phi(x_i) \prod_{k \in N(i) \setminus j} m_{ki}(x_i)$$

Computing Node Belief (Reputation)

Details

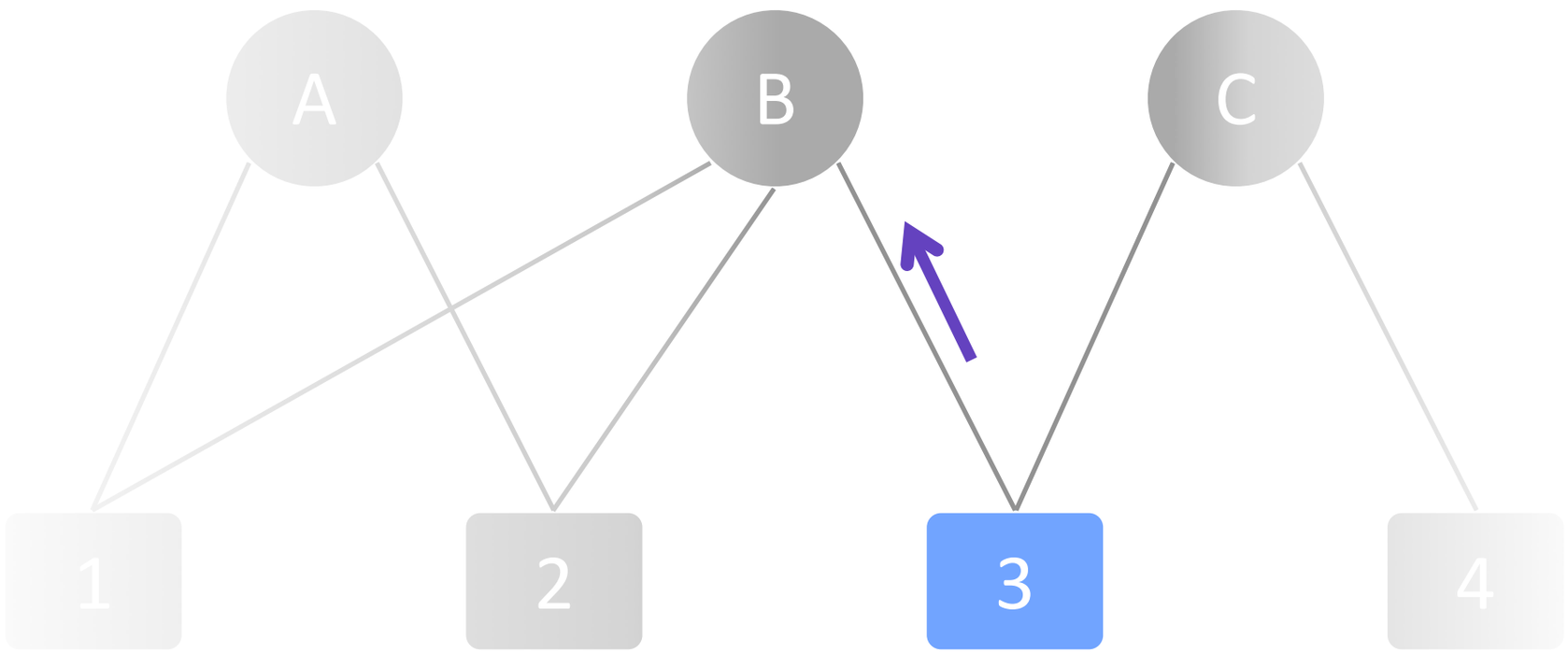


$$b_i(x_i) = k\phi(x_i) \prod_{x_j \in N(i)} m_{ji}(x_i)$$

Belief Prior belief Neighbors' opinions

Creating Message for Neighbor

Details



$$m_{ij}(x_j) \leftarrow \sum_{x_i \in X} \psi_{ij}(x_i, x_j) \, \phi(x_i) \prod_{k \in N(i) \setminus j} m_{ki}(x_i)$$

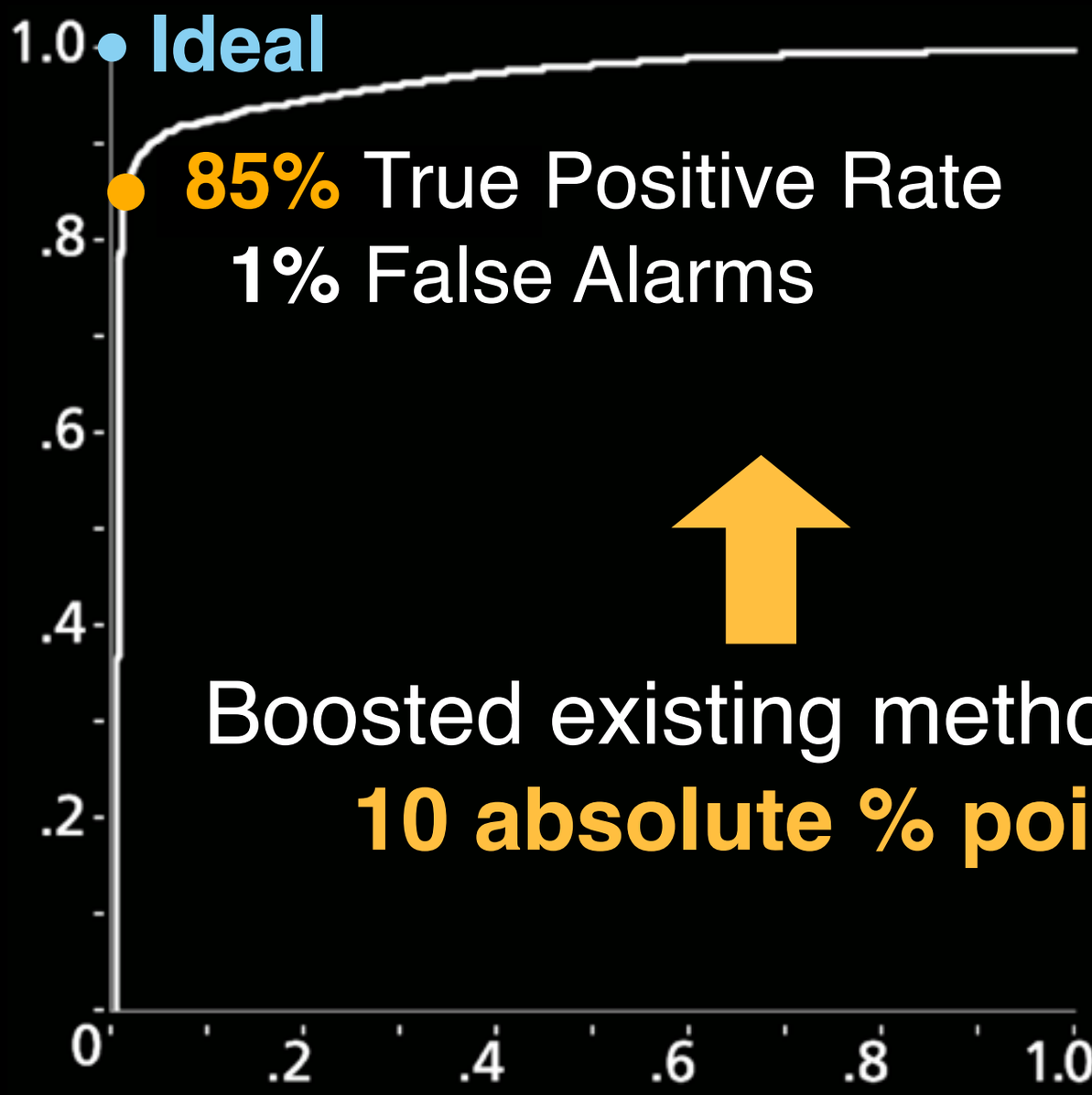
Opinion for neighbor — Edge potential — Belief

	Good	Bad
Good	0.9	0.1
Bad	0.1	0.9

Evaluation

Using **millions** of ground truth files,10-fold cross validation

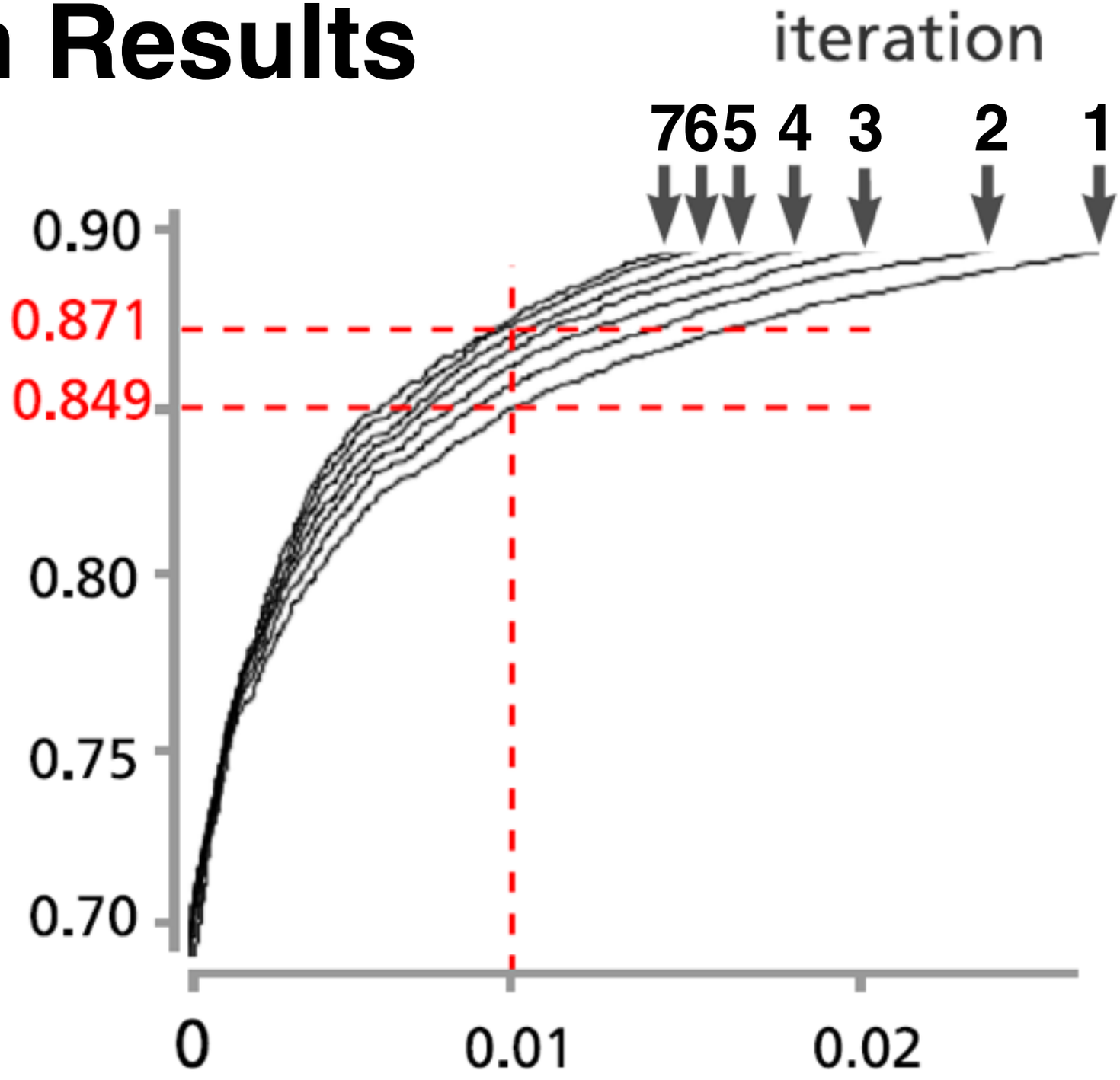
True Positive Rate
% of **bad** correctly labeled



Boosted existing methods by
10 absolute % point

False Positive Rate (False Alarms)
% of **good** labeled as **bad**

Multi-Iteration Results



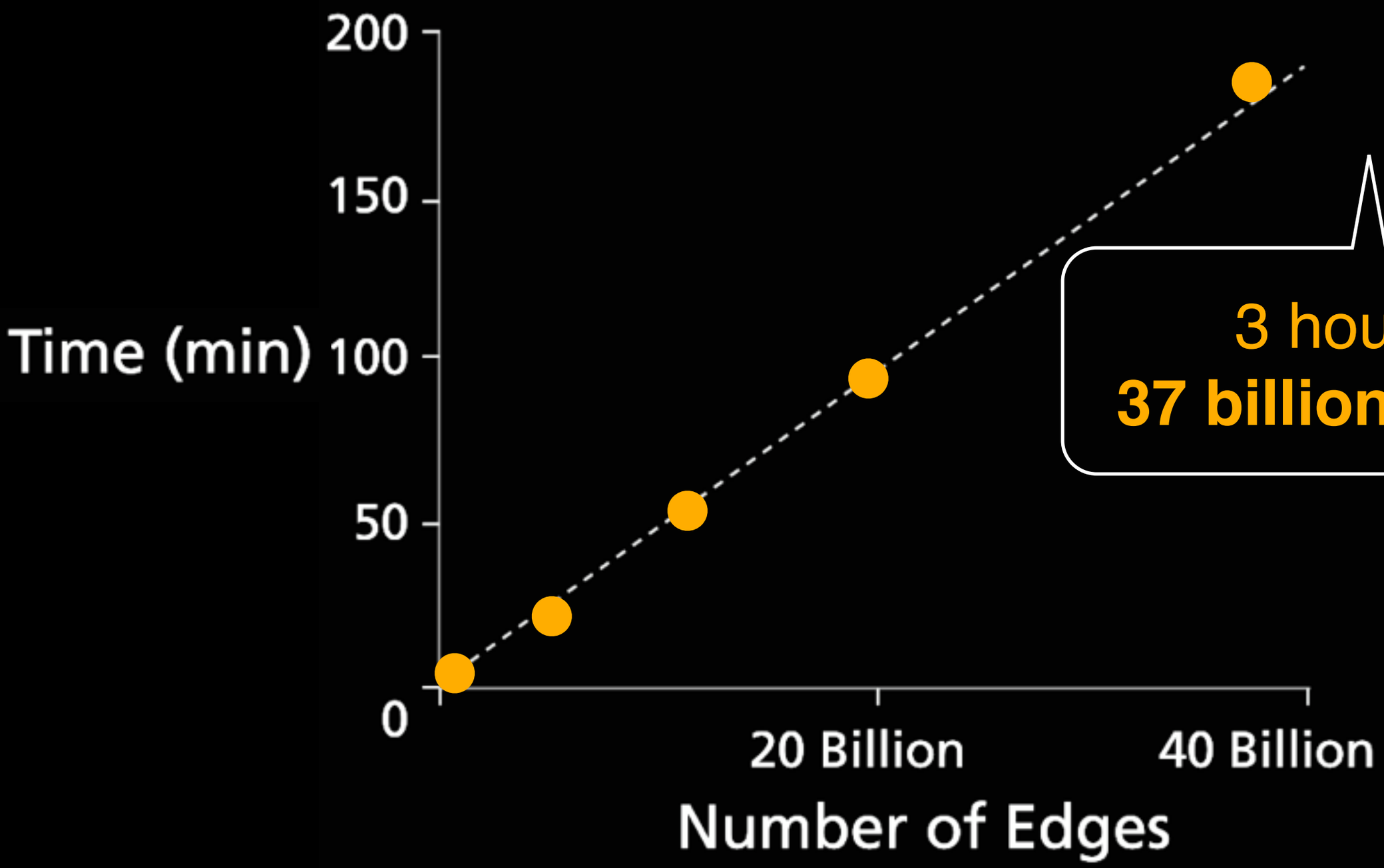
True Positive Rate
% of **bad** correctly labeled

False Positive Rate (False Alarm)
% of **good** labeled as **bad**

Scalability

Running Time Per Iteration

Linux
16-core Opteron
256GB RAM



Details

Scalability

How Did I Scale Up BP?

1. Early termination (after 6 iterations) → Faster

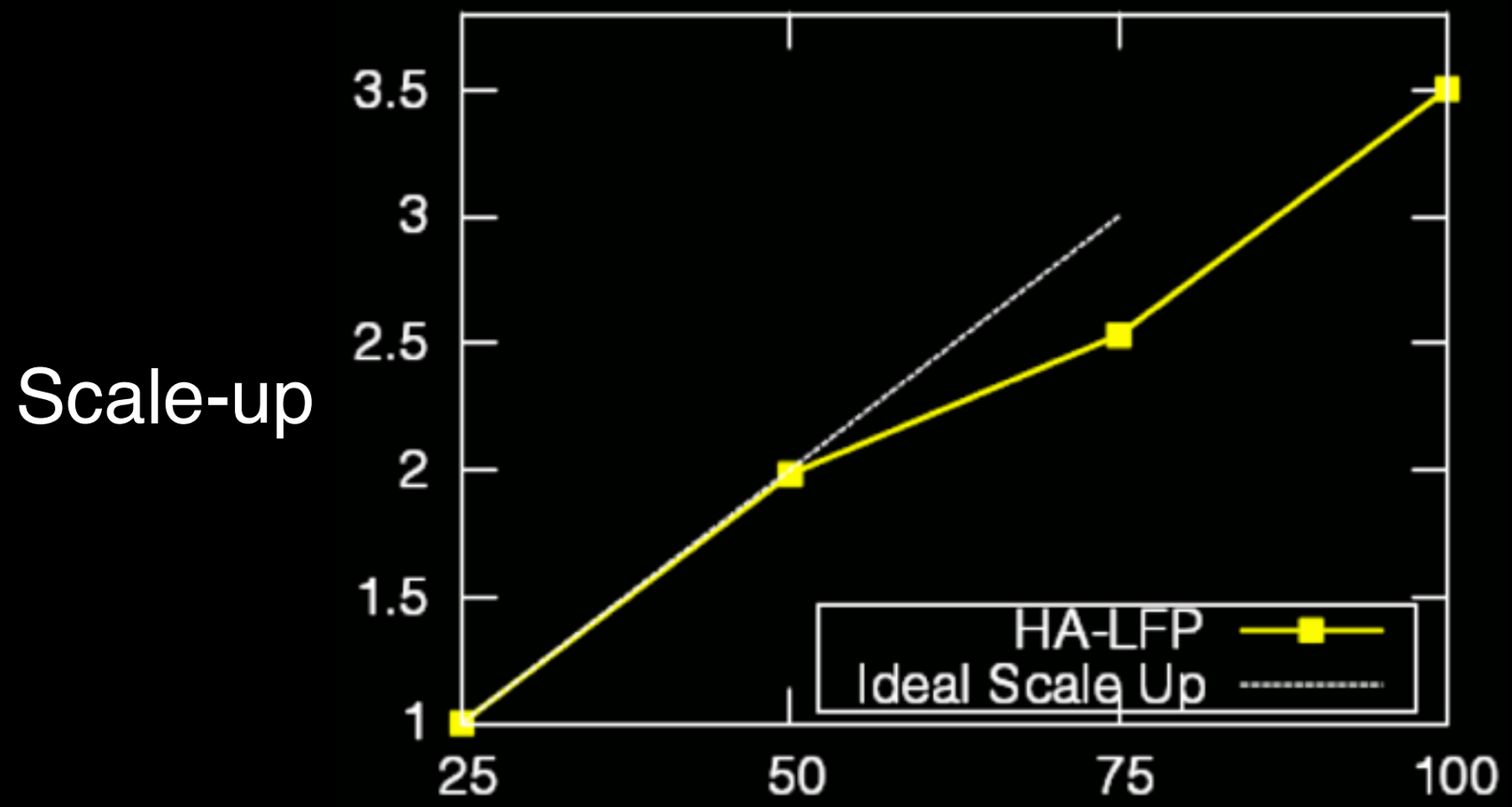
2. Keep edges on disk → Saves 200GB of RAM

3. Computes half of the messages → Twice as fast

Further Scale Up Belief Propagation

Use **Hadoop** if graph doesn't fit in memory [ICDE'11]

Speed scales up **linearly** with number of machines



Yahoo! M45 cluster

480 machines

1.5 PB storage

3.5TB machine