Link Analysis: TrustRank and WebSpam

CS246: Mining Massive Datasets

Jure Leskovec, Stanford University
Charilaos Kanatsoulis, Stanford University

http://cs246.stanford.edu

PageRank with Random Teleports

- **PageRank equation** [Brin-Page, 98]

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{N}$$

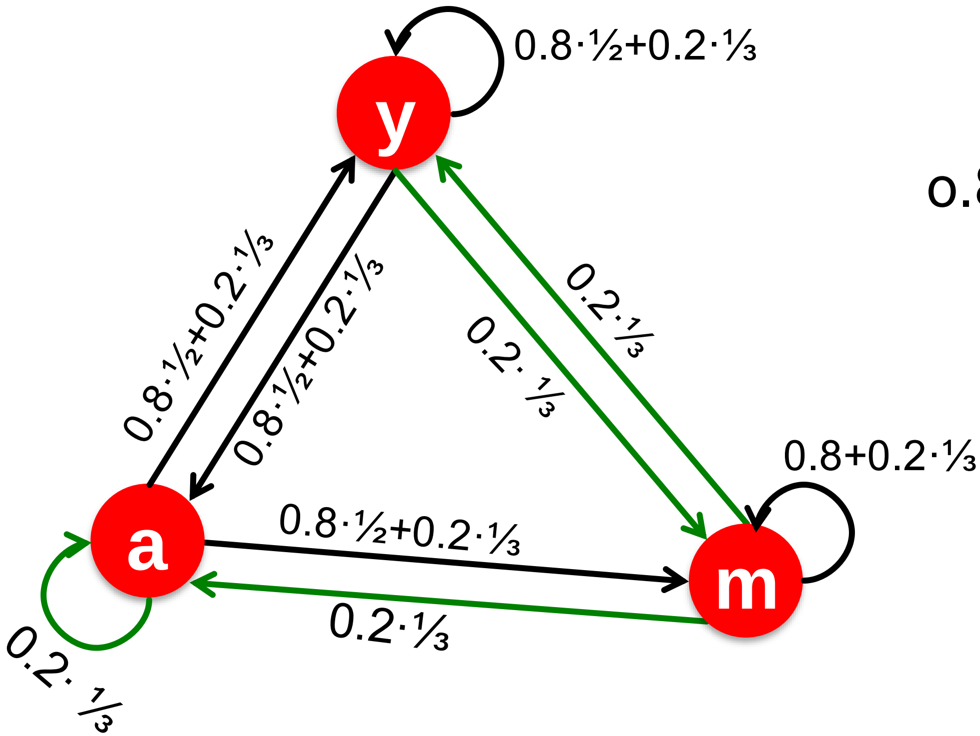
d_i … out-degree of node i

- **The Google Matrix *A*:**

$$A = \beta\, M + (1-\beta)\left[\frac{1}{N}\right]_{N \times N}$$

- At each step, random surfer has two options:
  - With probability β, follow a link at random
  - With probability $1\text{-}\beta$, jump to some random page

Random Teleports ($\beta = 0.8$)



$$\overset{\mathbf{M}}{0.8\begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix}} + 0.2 \overset{[\mathbf{1/N}]_{\mathbf{NxN}}}{\begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}}$$

$$\begin{array}{c} y \\ a \\ m \end{array} \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}$$

$$\begin{array}{c} y \\ a \\ m \end{array} = \begin{array}{c} 1/3 \\ 1/3 \\ 1/3 \end{array} \quad \begin{array}{c} 0.33 \\ 0.20 \\ 0.46 \end{array} \quad \begin{array}{c} 0.28 \\ 0.20 \\ 0.52 \end{array} \quad \begin{array}{c} 0.26 \\ 0.18 \\ 0.56 \end{array} \quad \dots \quad \begin{array}{c} 7/33 \\ 5/33 \\ 21/33 \end{array}$$

Web Search and PageRank

- **Model the web as a graph**
- **Compute the importance of webpages with PageRank**
- **Web-search query**
  - The user types the query “Trojan”
- **Identify relevant webpages**
  - Find webpages relevant to “Trojan”
- **Show them to the user**
  - Webpages with high generic PageRank will be presented first

Some Problems with PageRank

- **Measures generic importance of a page**
  - Will ignore/miss topic-specific authorities
  - **Solution:** Topic-Specific PageRank (**next**)
- **Uses a single measure of importance**
  - Other models of importance
  - **Solution:** Hubs-and-Authorities
- **Susceptible to Link spam**
  - Artificial link topographies created in order to boost page rank
  - **Solution:** TrustRank

Topic-Specific PageRank

Web Search and PageRank

- **Model the web as a graph**
- **Compute the importance of webpages with PageRank**
- **Web-search query**
  - The user types the query “Trojan”
- **Identify relevant webpages**
  - Find webpages relevant to “Trojan”
- **Show them to the user**
  - Webpages with high generic PageRank will be presented first

Topic-Specific PageRank

- **Model the web as a graph**
- **Web-search query**
  - The user types the query “Trojan”
- **Identify relevant webpages**
  - Find webpages relevant to “Trojan”
- **Compute the importance of a webpage according to their relevance to a topic**
- **Show them to the user**
  - Webpages with high generic PageRank will be presented first

Topic-Specific PageRank

- **Instead of generic importance, can we measure importance within a topic?**
- **Goal:** Evaluate Web pages not just according to their importance, but also by how close they are to a particular topic, e.g. "sports" or "history"
- **Allows search queries to be answered based on the interests of a user**
  - **Example:** Query "Trojan" wants different pages depending on whether you are interested in sports, history, or computer security

Topic-Specific Teleportation

- Random walker has a small probability of teleporting at any step
- **Teleport can go to:**
  - **Standard PageRank: Any page with equal probability**
    - To avoid dead-end and spider-trap problems
  - **Topic Specific PageRank: A topic-specific set of "relevant" pages (teleport set)**
- **Idea: Bias the random walk**
  - When the walker teleports, they pick a page from a set S
  - S contains only pages that are relevant to the topic
  - For each teleport set S, we get a different vector r_S

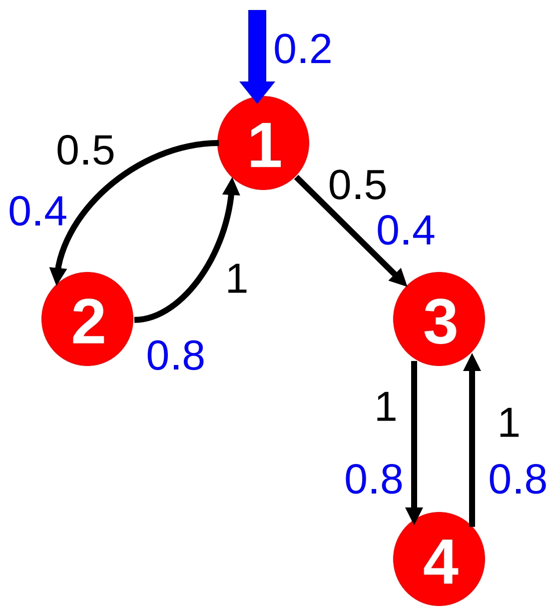
Matrix Formulation

- **To make this work all we need is to update the teleportation part of the PageRank formulation:**

$$A_{ij} = \begin{cases} \beta M_{ij} + (1-\beta)/|S| & \textbf{if } i \in S \\ \beta M_{ij} + 0 & \textbf{otherwise} \end{cases}$$

  - ***A*** is a stochastic matrix!
- We weighted all pages in the teleport set ***S*** equally
  - **Could also assign different weights to pages!**
- **Compute as for regular PageRank:**
  - Multiply by ***M***, then add a vector of $(1-\beta)/|S|$
  - Maintains sparseness

Example: Topic-Specific PageRank



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

Node	Iteration 0	1	2	…	stable
1	0.25	0.4	0.28		0.294
2	0.25	0.1	0.16		0.118
3	0.25	0.3	0.32		0.327
4	0.25	0.2	0.24		0.261

S={1}, β=0.9:
r=[0.17, 0.07, 0.40, 0.36]
S={1} , β=0.8:
r=[0.29, 0.11, 0.32, 0.26]
S={1}, β=0.7:
r=[0.39, 0.14, 0.27, 0.19]

S={1,2,3,4}, β=0.8:
r=[0.13, 0.10, 0.39, 0.36]
S={1,2,3} , β=0.8:
r=[0.17, 0.13, 0.38, 0.30]
S={1,2} , β=0.8:
r=[0.26, 0.20, 0.29, 0.23]
S={1} , β=0.8:
r=[0.29, 0.11, 0.32, 0.26]

Discovering the Teleport Set *S*

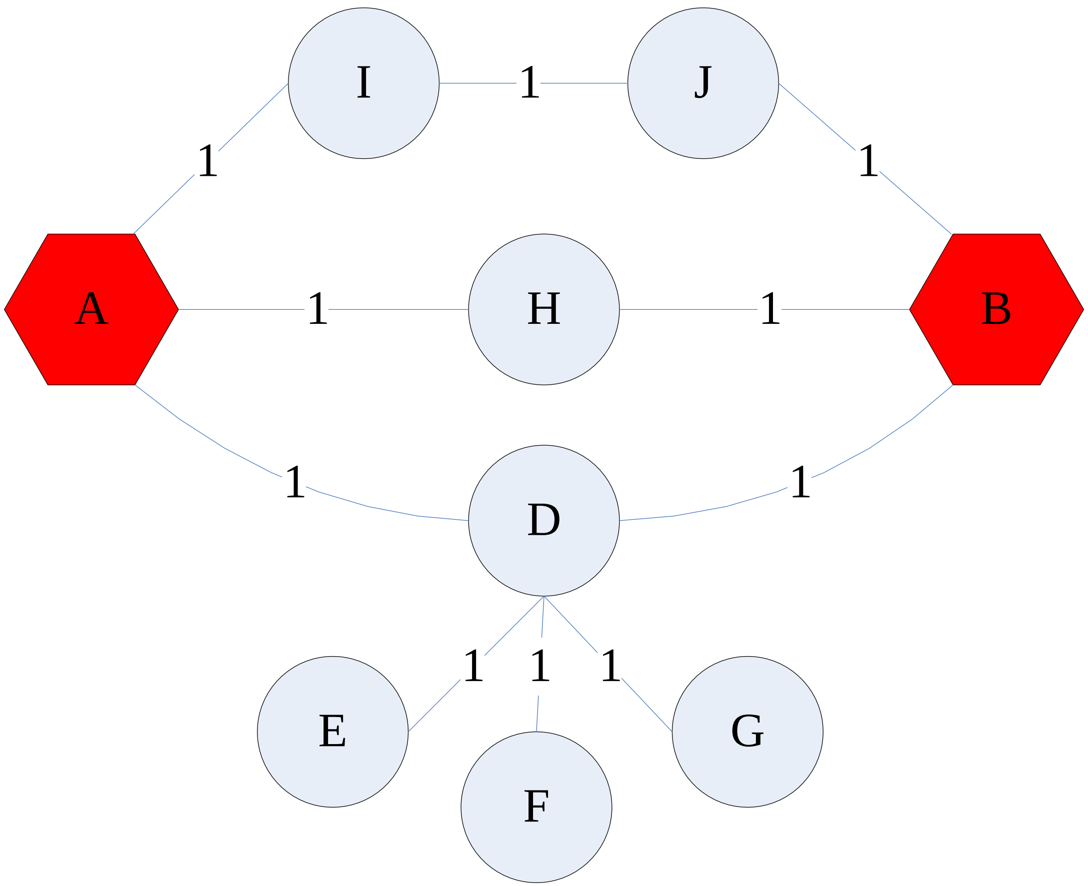
- **Create different PageRanks for different topics**
  - The 16 DMOZ top-level categories:
    - Arts, Business, Sports,…
- **Which topic ranking to use?**
  - User can pick from a menu
  - Classify query into a topic
  - Can use the **context** of the query
    - E.g., query is launched from a web page talking about a known topic
    - History of queries e.g., “basketball” followed by “Jordan”
  - User context, e.g., user’s bookmarks, …

Application to Measuring Proximity in Graphs

Random Walk with Restarts: Set *S* is a single node

[Tong-Faloutsos, ‘06]

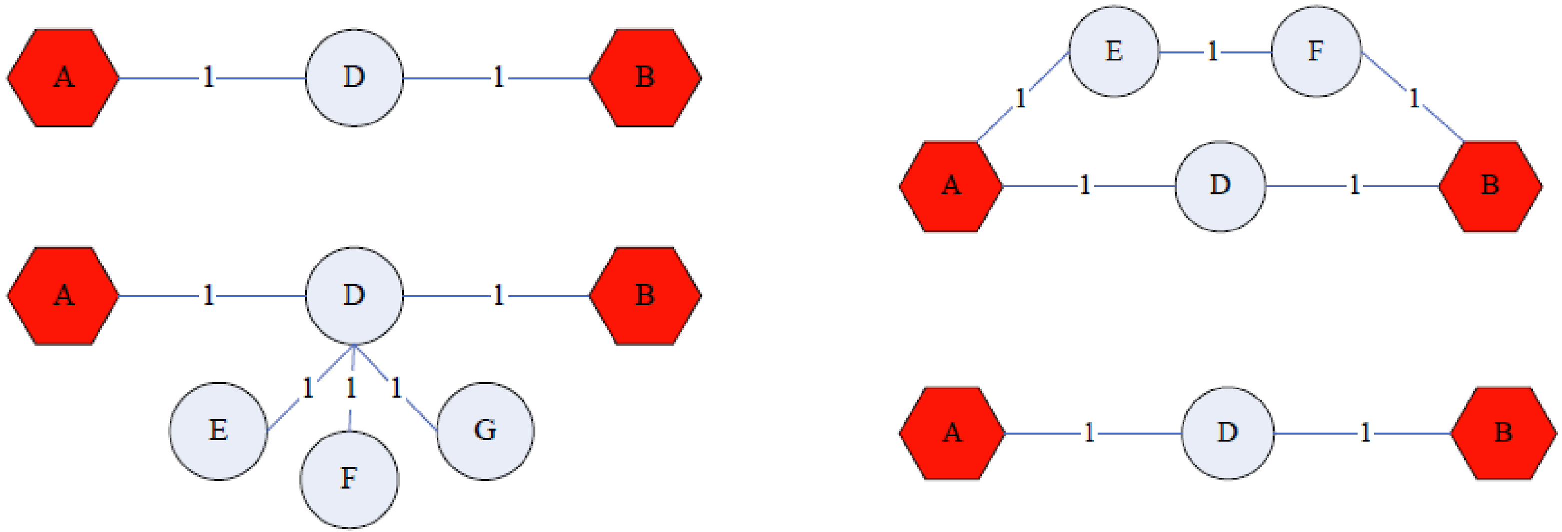
Proximity on Graphs



a.k.a.: Relevance, Closeness, ‘Similarity’…

Good proximity measure?

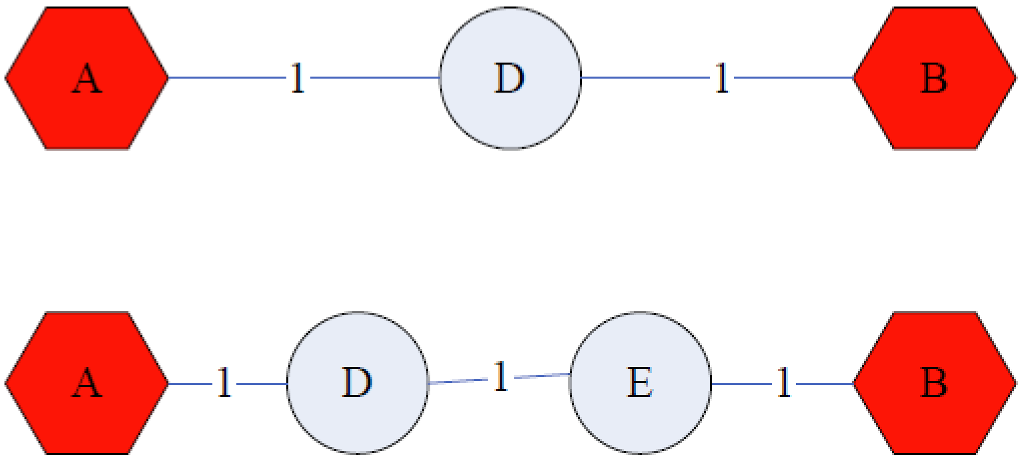
- **Shortest path is not good:**



- **No effect of degree-1 nodes (E, F, G)!**
- Multi-faceted relationships

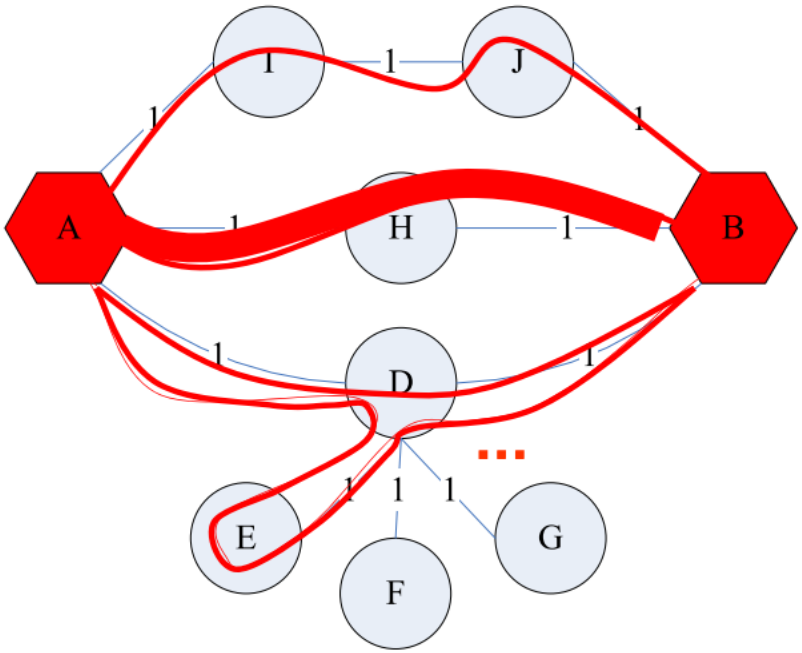
Good proximity measure?

- **Network flow is not good:**



- **Does not punish long paths**

What is a good notion of proximity?

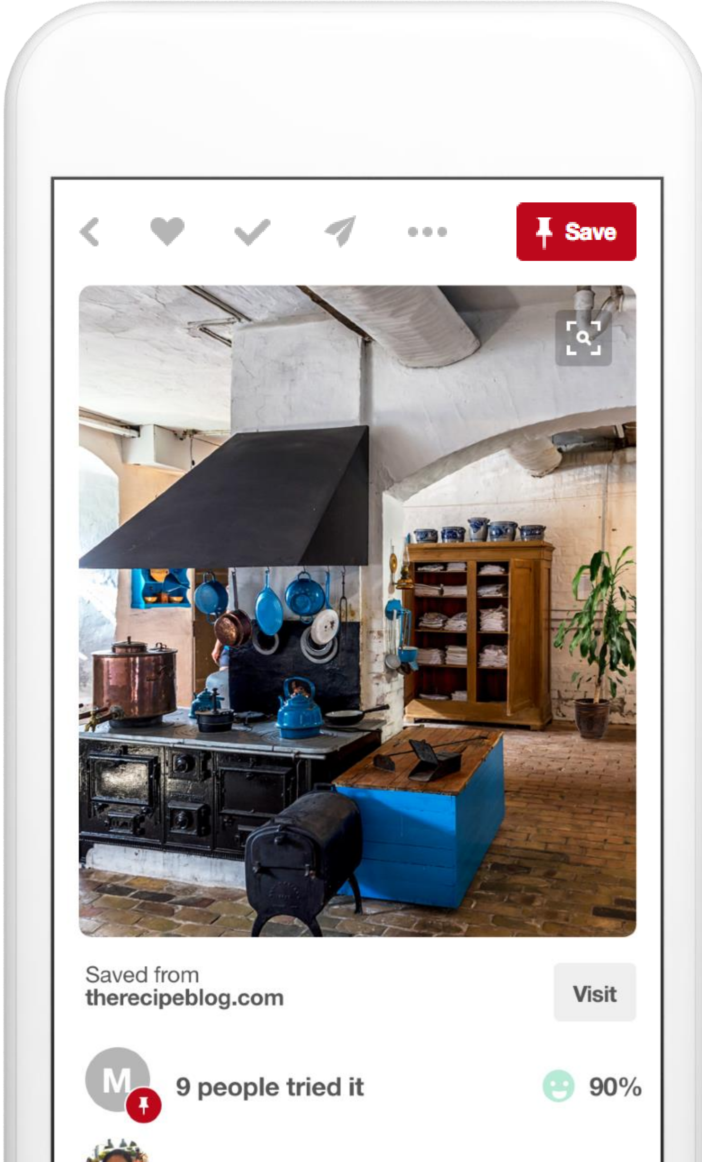


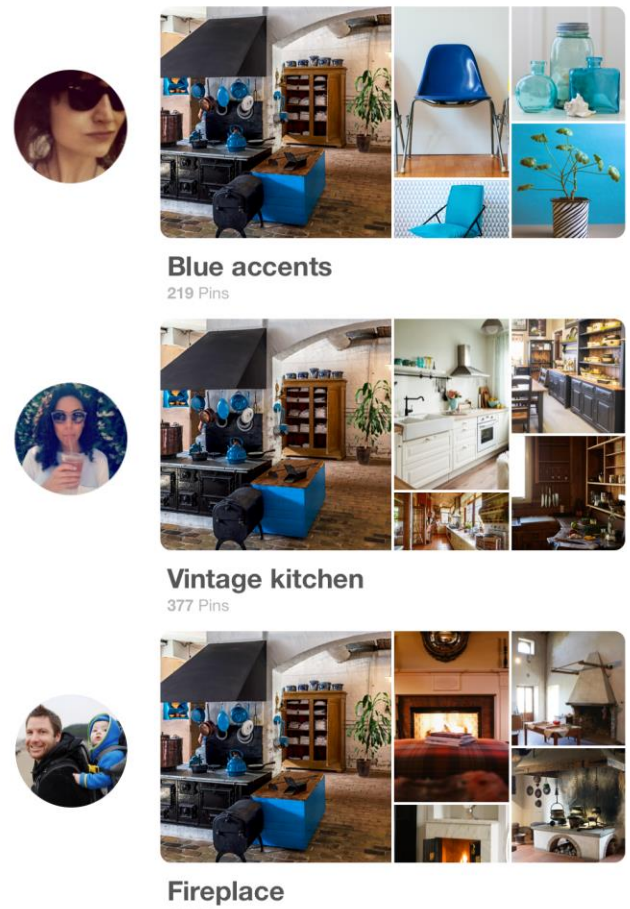
- **Need a method that considers:**
  - Multiple connections
  - Multiple paths
  - Degree of the node

Pixie: Random Walk-based Real-Time Recommender System at Pinterest

https://labs.pinterest.com/user/themes/pin_labs/assets/paper/paper-pixie.pdf

Pinterest


Save
Saved from
therecipeblog.com
Visit
M
9 people tried it
90%


Blue accents
219 Pins
Vintage kitchen
377 Pins
Fireplace
138 Pins

Goal: Radical Personalization

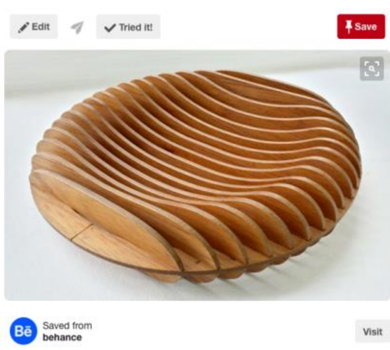
- **Recommendations can be radically personalized.**

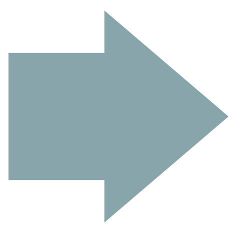
- Adapting in real-time
- **Highly scalable**

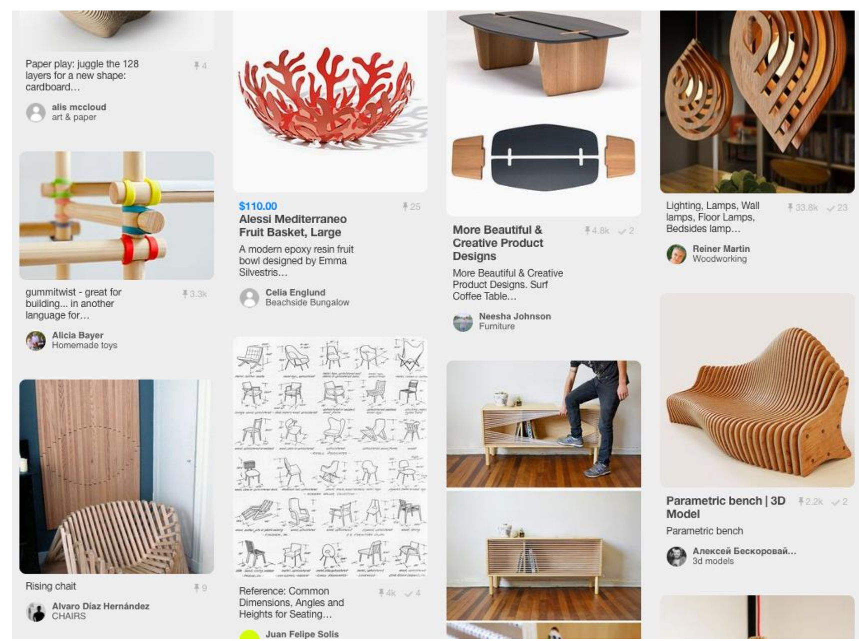
Recommendation problem

How to provide relevant and responsive recommendations

- From 100B Pins to 1K Pins in **real-time (50ms, 200,000x/s)**







From Pins to Pins

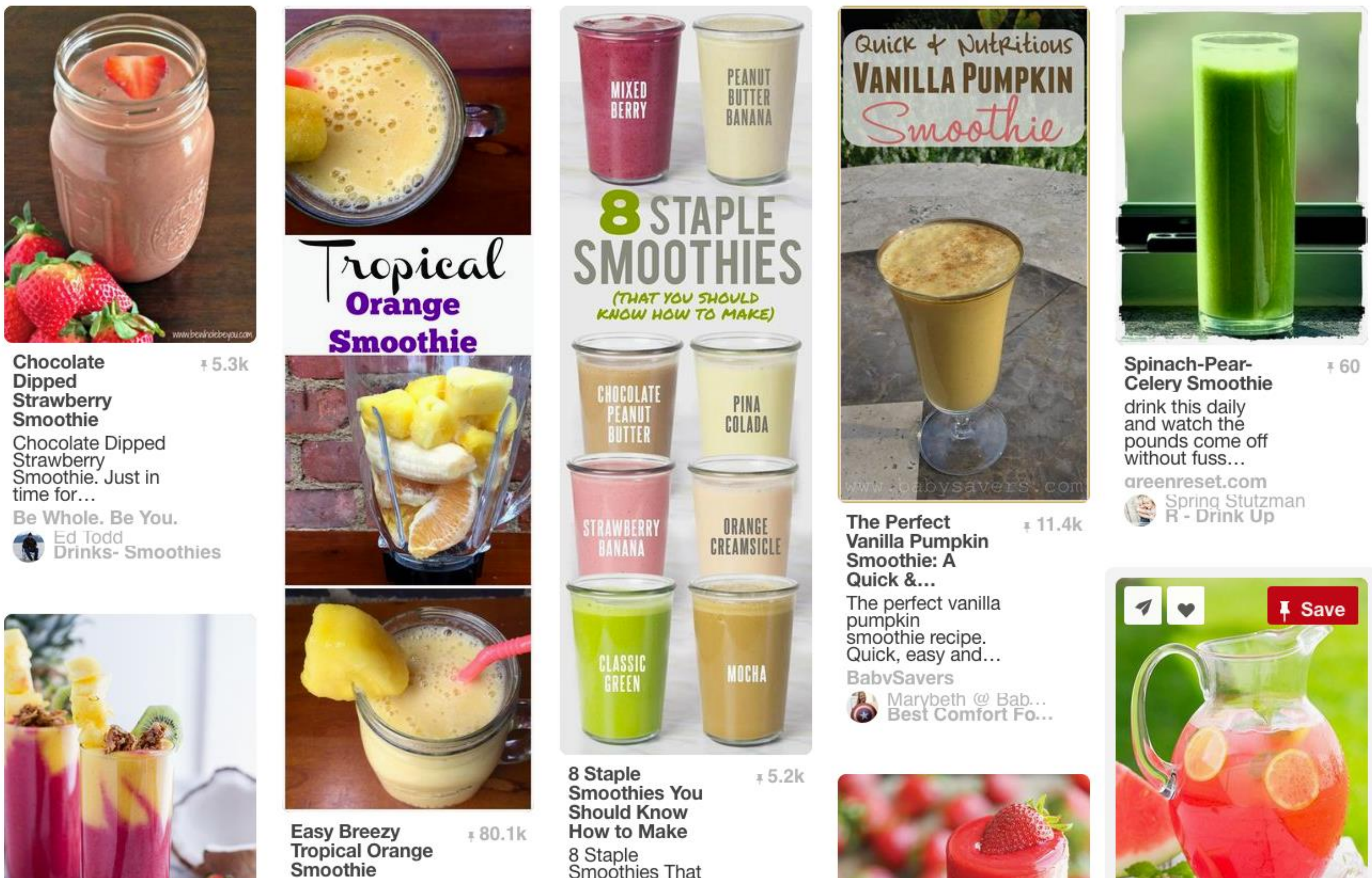
Input:

From Pins to Pins

- Pins to Pins

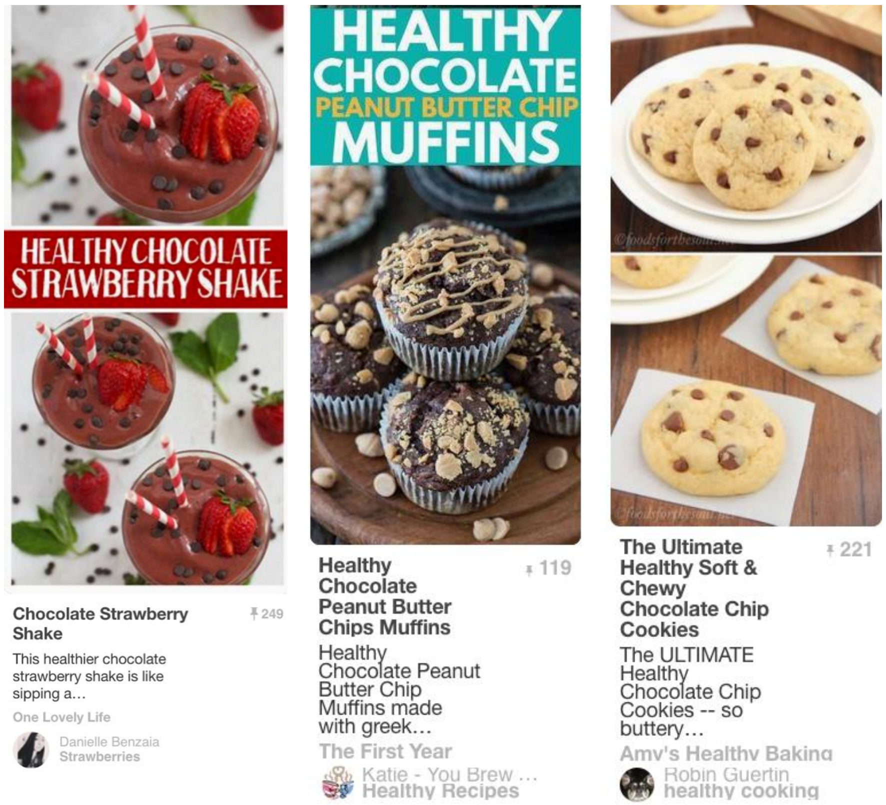
Input:

Output:



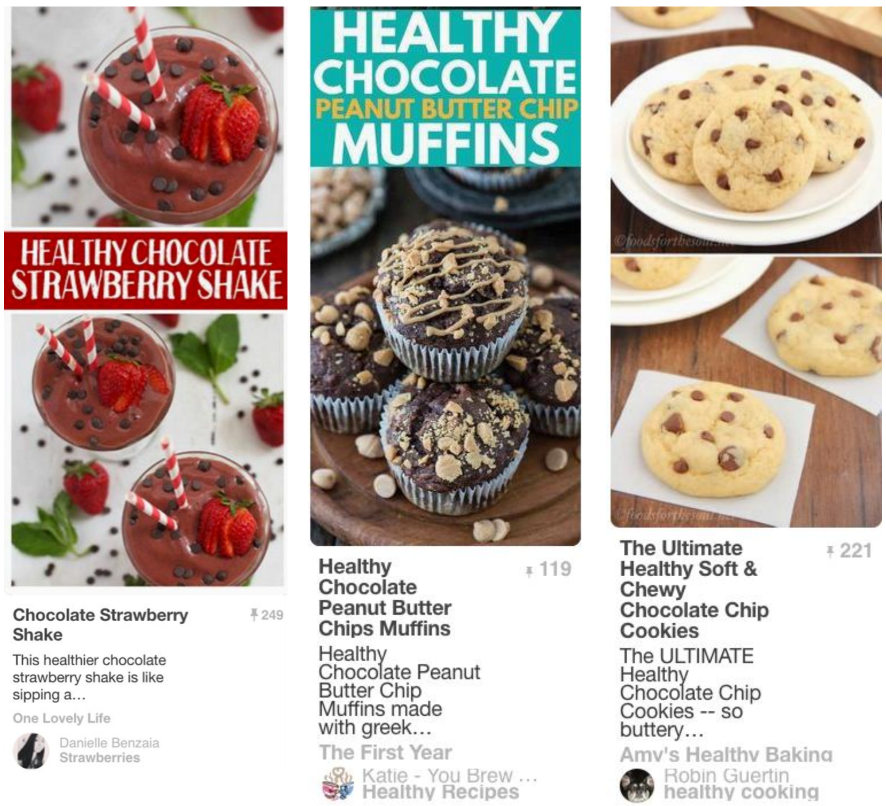
From Pins to Pins

Input:

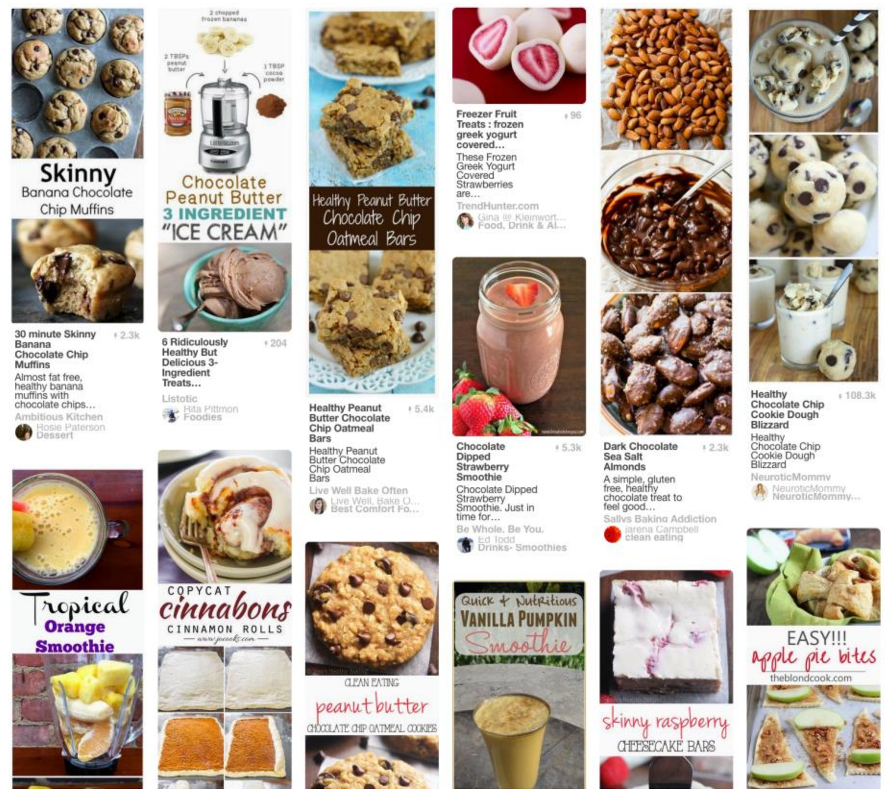


From Pins to Pins

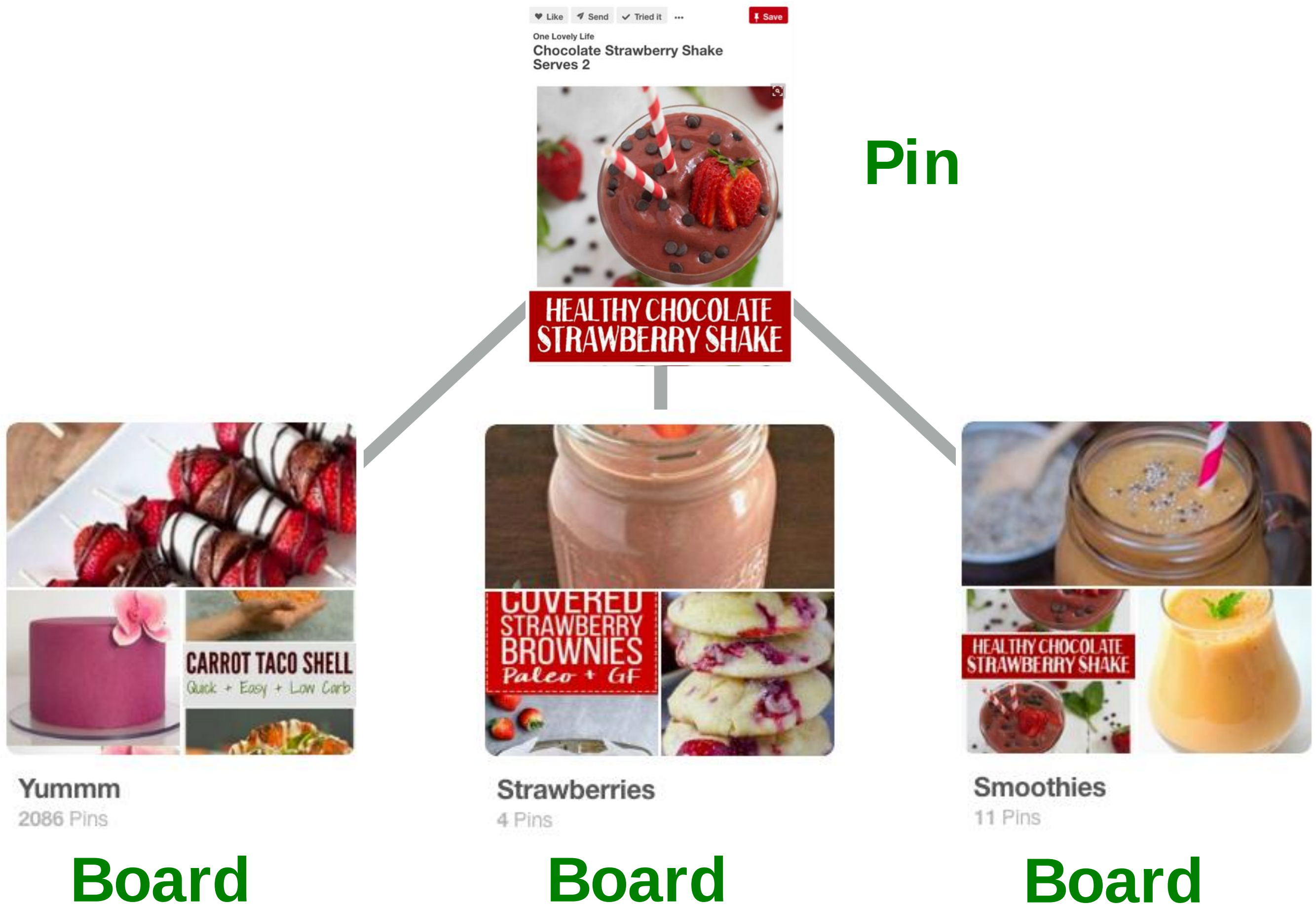
Input:



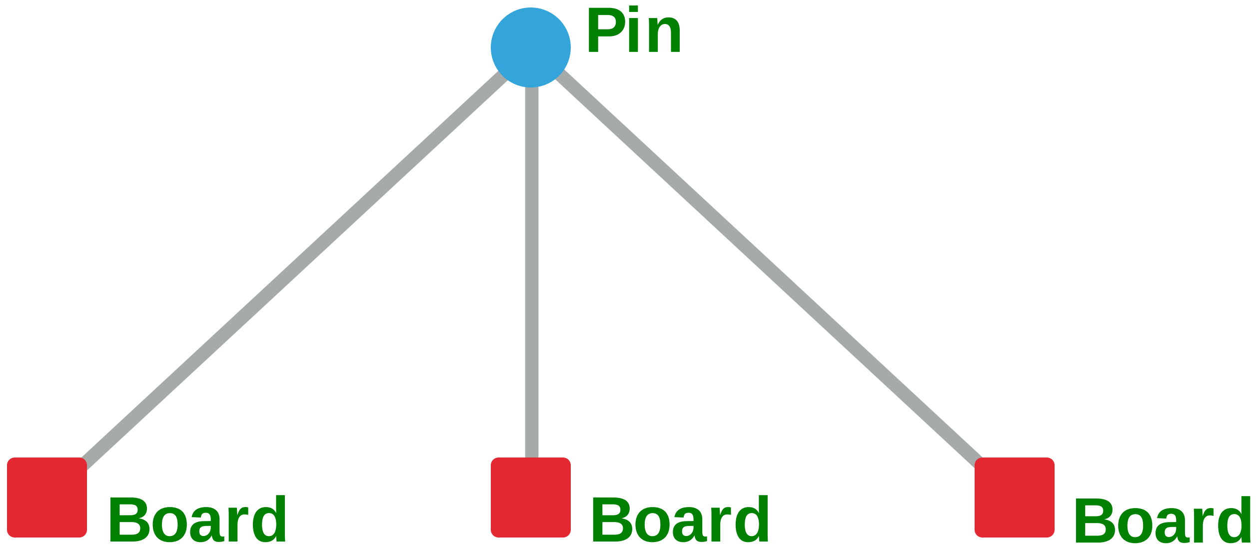
Output:



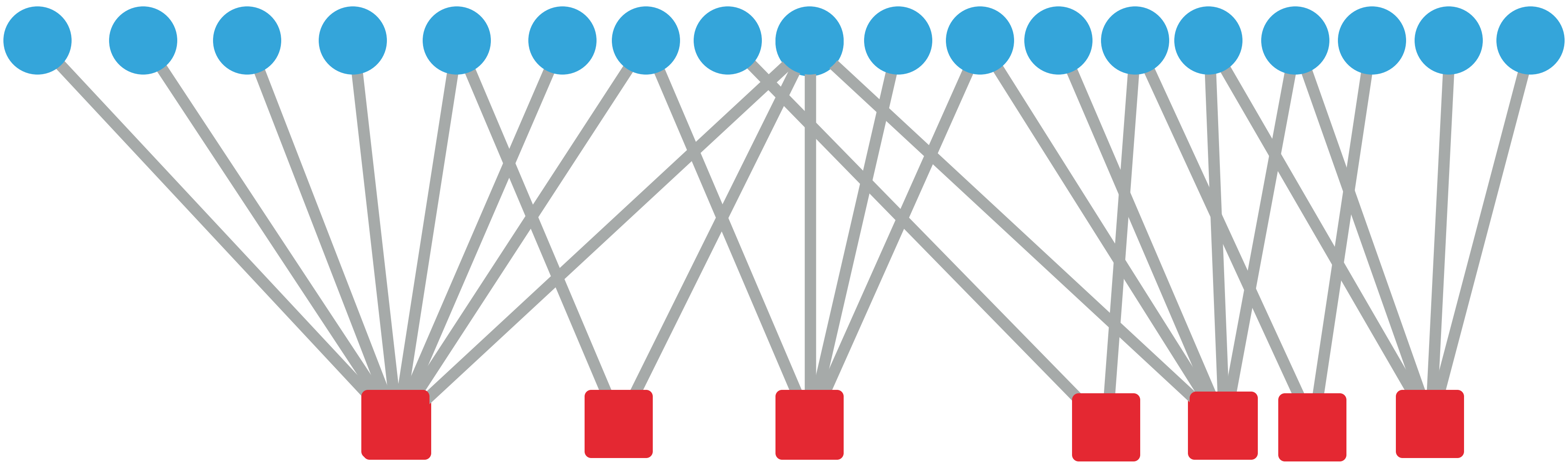
Pinterest is a Giant Bipartite Graph



Bipartite Pin And Board Graph

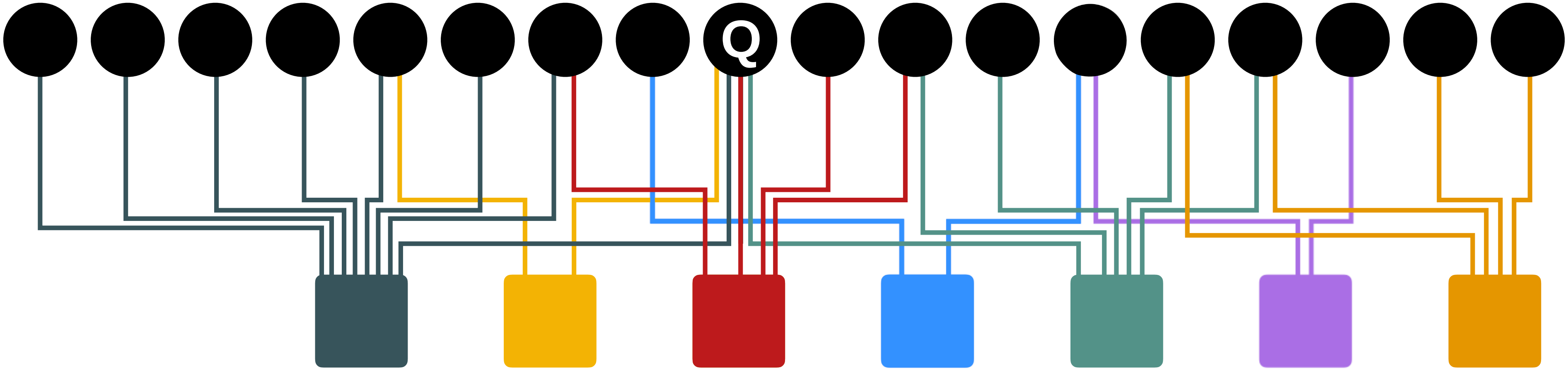


Bipartite Pin And Board Graph



Pixie Random Walks

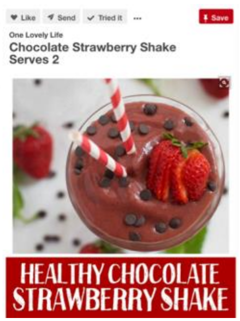
- **Idea:**
  - Every node has some importance
  - Importance gets evenly split among all edges and pushed to the neighbors
- Given a set of QUERY NODES Q, **simulate a random walk**:



Pixie Random Walk Algorithm

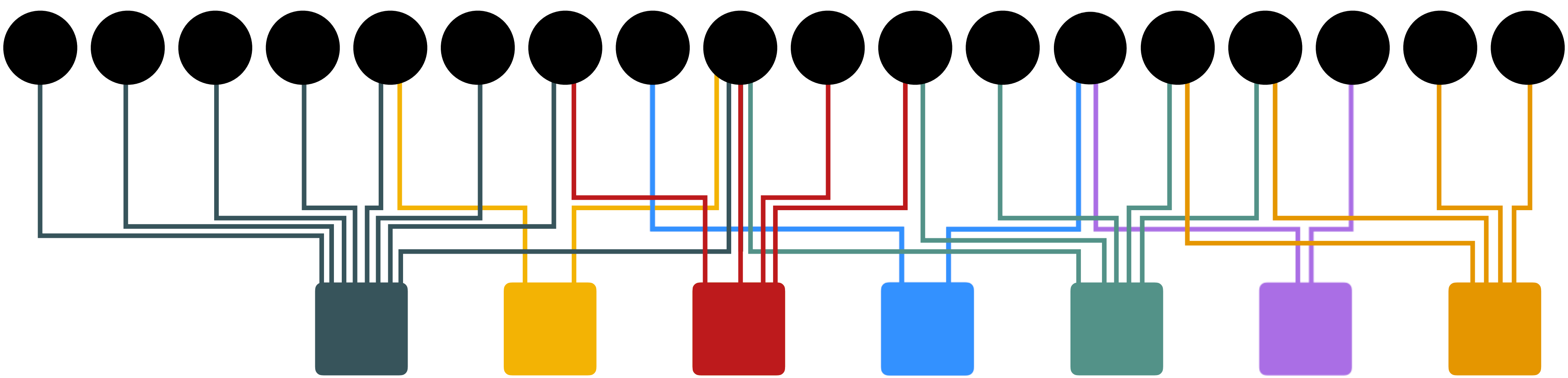
- Proximity to query node(s) ***Q***:

```
ALPHA = 0.5
QUERY_NODES = {
```




```
}
```

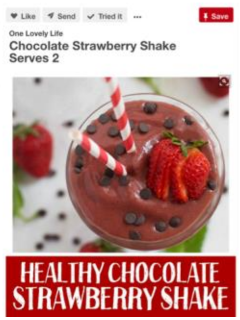
```
pin_node = QUERY_NODES.sample_by_weight()
for i in range(N_STEPS):
    board_node = pin_node.get_random_neighbor()
    pin_node = board_node.get_random_neighbor()
    pin_node.visit_count += 1
    if random() < ALPHA:
        pin_node = QUERY_NODES.sample_by_weight()
```



Pixie Random Walk Algorithm

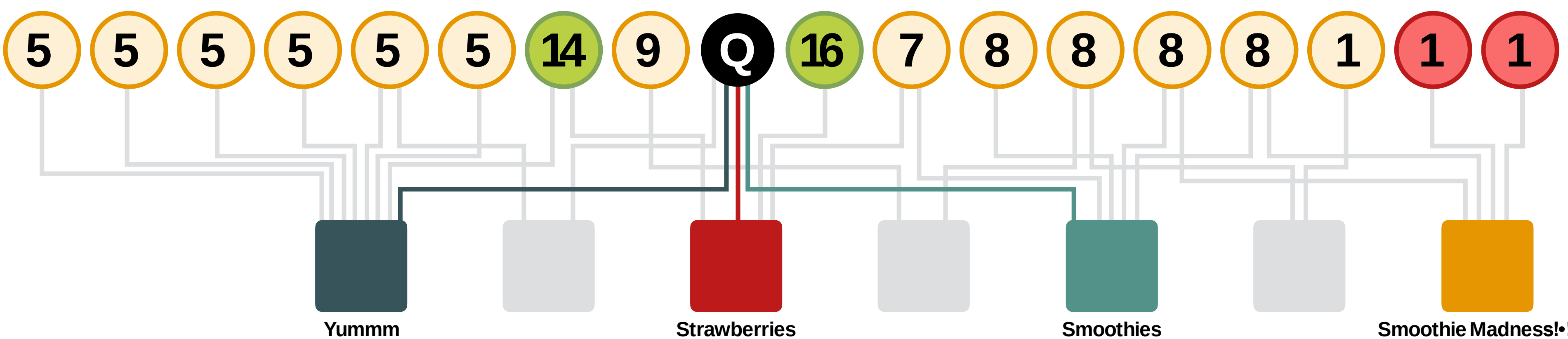
- Proximity to query node(s) ***Q***:

```
ALPHA = 0.5
QUERY_NODES = {
```




```
}
```

```
pin_node = QUERY_NODES.sample_by_weight()
for i in range(N_STEPS):
    board_node = pin_node.get_random_neighbor()
    pin_node = board_node.get_random_neighbor()
    pin_node.visit_count += 1
    if random() < ALPHA:
        pin_node = QUERY_NODES.sample_by_weight()
```



Pixie Recommendations

- **Pixie:**
  - **Outputs top 1k pins with highest visit count**

Extensions:

- **Weighted edges:** The walk prefers to traverse certain edges:
  - Edges to pins in your local language
  - Personalized edge weights:
  - Pixie for different users and query pins can choose to bias edge selection dynamically based on user and edge features.
    - `Weight = PersonalizedNeighbor(E,U)`, where E is edge and U is the user.

Pixie Recommendations

Extensions:

- **Multiple query pins:**
  - Each query pin q gets a different importance w_q
  - Run PixieRandomWalk for each q in parallel.
  - Combine visit counts.
  - **Important insight:** The number of steps required to obtain meaningful visit counts depends on the query pin's degree
    - Scale the number of steps allocated to each query pin to be proportional to its degree

Pixie Recommendations

Extensions:

- **Multi-hit Booster:**
  - For multi-pin queries we prefer recommendations related to multiple query pins q.
    - Candidates with high visit counts from multiple query pins are more relevant to the query than candidates having equally high total visit count but all coming from a single query pin.
  - **Solution:** When combining visit counts use:

$$V[p] = \left(\sum_{q \in Q} \sqrt{V_q[p]} \right)^2$$

Note that when a candidate pin p is visited by walks from only a single query pin q then the count is unchanged. However, if the candidate pin is visited from multiple query pins, then the count is boosted.

Pixie Recommendations

Extensions:

- **Early stopping:**
  - Insight: We only care about top-1k most visited pins.
  - So, we don't need to walk a fixed big number of steps
  - We just walk until 1k-th most visited pin has at least 20 visits.

https://labs.pinterest.com/user/themes/pin_labs/assets/paper/paper-pixie.pdf

Graph Cleaning/Pruning

- **Pinterest graph has 200B edges**
- We don't need all of them!
  - Super popular pins are pinned to millions of boards
    - **Not useful:** When the random walk hits the pin, the signal just disperses. Such pins appear randomly in our recommendations.
- **What we did:** Keep only good boards for pins
  - Compute the similarity between pin's topic vector and each of its boards. Only take boards with high similarity.

Data Type	Number	Size	Memory
Pin Nodes	3 Billion	8 Bytes	24 GiB
Board Nodes	2 Billion	8 Bytes	16 GiB
Undirected Edges	20 Billion	8 Bytes	160 GiB
			208 GiB

Benefits of Pixie

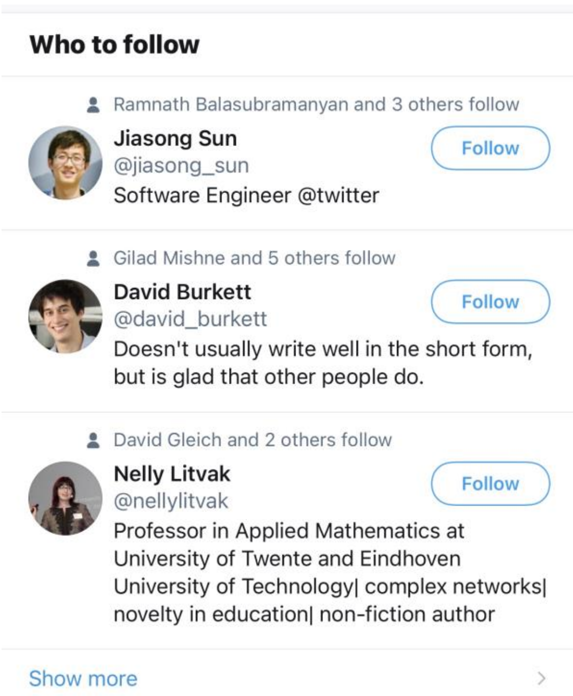
- **Benefits:**
  - Blazingly fast: Given Q, we can output top 1k in 50ms (after doing ~100k steps of the random walk)
  - Single machine can run 1,500 walks in parallel (1500 recommendation requests per second).
  - Fit entire graph in RAM of a single machine (17B edges, 3B nodes)
  - Can scale it by just adding more machines

To learn more read: https://cs.stanford.edu/people/jure/pubs/pixie-www18.pdf

Recommendations@Twitter

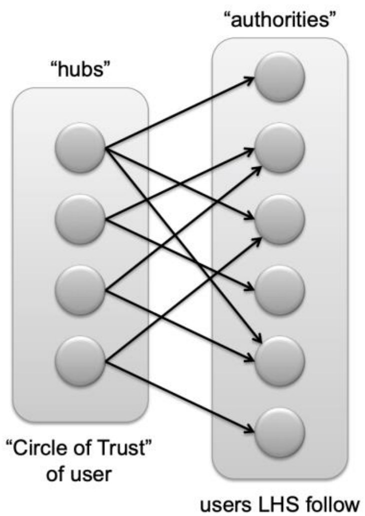
Joint work with many Twitter folks over several years:
http://www2013.w3c.br/proceedings/p505.pdf
https://www.vldb.org/pvldb/vol9/p1281-sharma.pdf

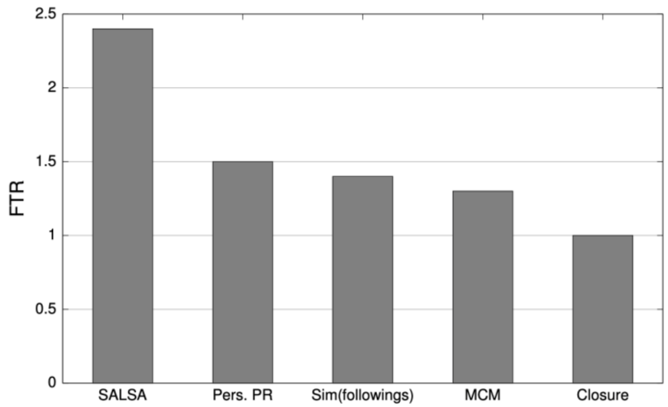
Recommendations@Twitter



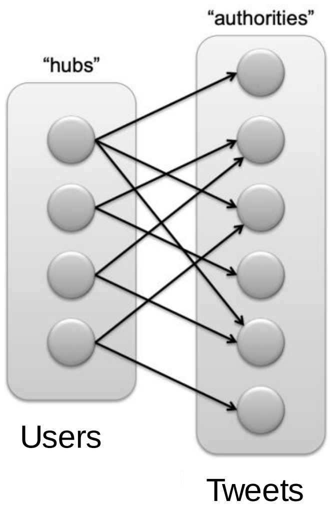
SALSA for Recommendations

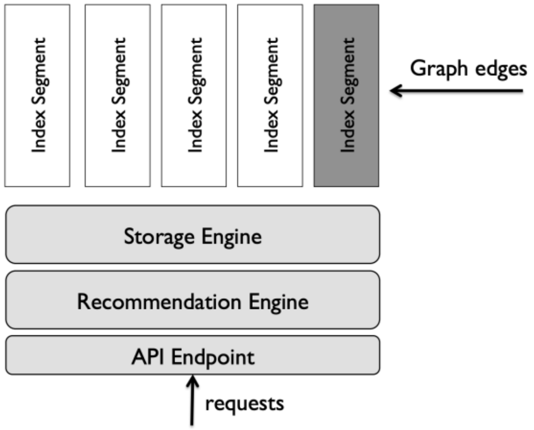
User Recs





Content Recs





TrustRank: Combating Spam on the Web

What is Web Spam?

- **Spamming:**
  - Any deliberate action to boost a web page's position in search engine results, incommensurate with the page's real value
- **Spam:**
  - Web pages that are the result of spamming
- This is a very broad definition
  - **SEO** industry might disagree!
  - SEO = search engine optimization
- Approximately **10-15%** of web pages are spam

Web Search

- **Early search engines:**
  - Crawl the Web
  - Index pages by the words they contained
  - Respond to search queries (lists of words) with the pages containing those words
- **Early page ranking:**
  - Attempt to order pages matching a search query by "importance"
  - **First search engines considered:**
    - **(1)** Number of times query words appeared
    - **(2)** Prominence of word position, e.g. title, header

First Spammers

- As people began to use search engines to find things on the Web, those with commercial interests tried to **exploit search engines** to bring people to their own site – whether they wanted to be there or not
- **Example:**
  - Shirt-seller might pretend to be about “movies”
- **Techniques for achieving high relevance/importance for a web page**

First Spammers: Term Spam

- **How do you make your page appear to be about movies?**
  - **(1)** Add the word movie 1,000 times to your page
    - Set text color to the background color, so only search engines would see it
  - **(2)** Or, run the query "movie" on your target search engine
    - See what page came on top of result ranking
    - Copy it into your page, make it "invisible"
- **These and similar techniques are term spam**

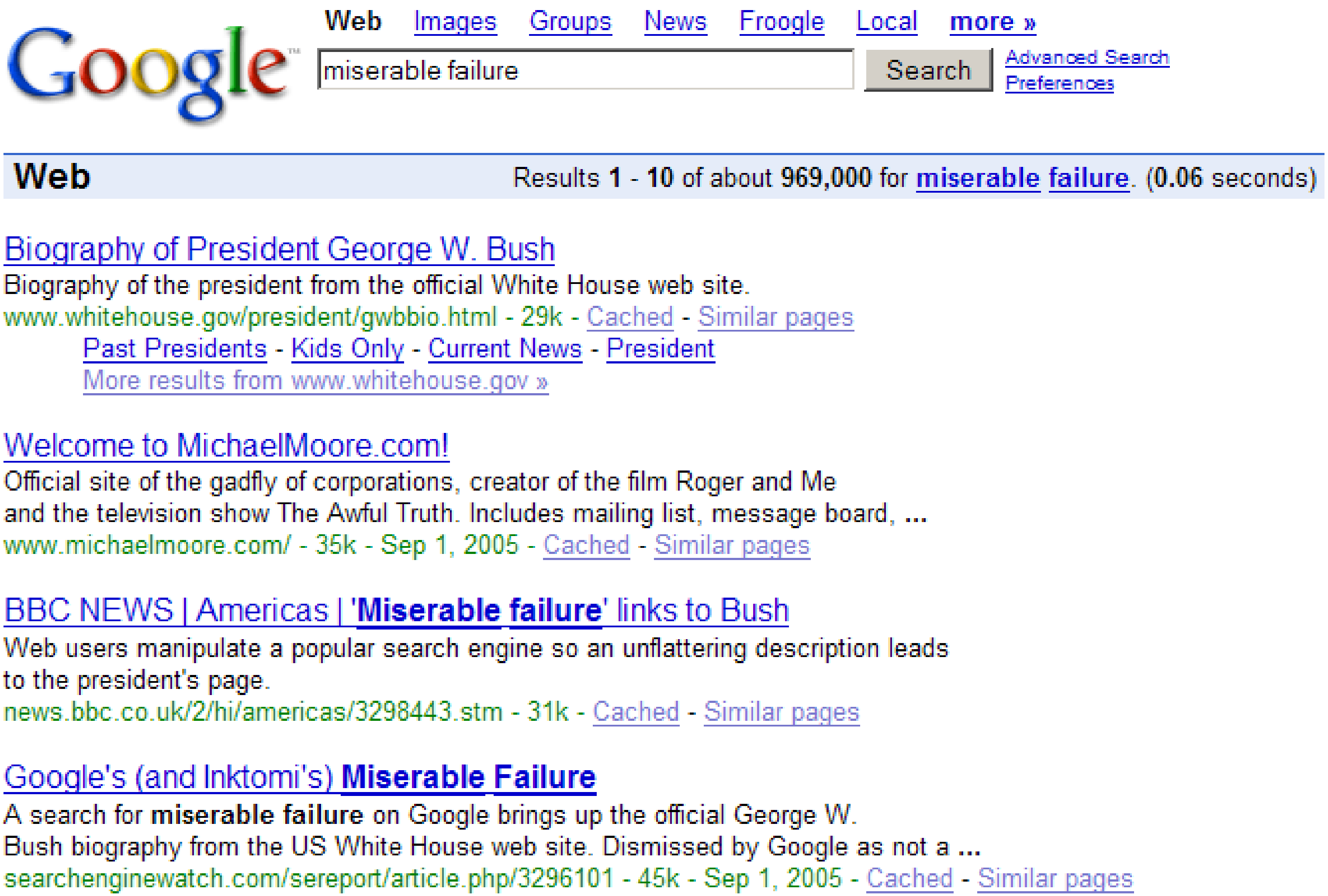
Google's Solution to Term Spam

- **Believe what people say about you, rather than what you say about yourself**
  - Use words in the anchor text (words that appear underlined to represent the link) and its surrounding text
- **PageRank as a tool to measure the "importance" of Web pages**

Why Does It Work?

- **Our hypothetical shirt-seller loses**
  - Saying they are about movies doesn't help, because others don't say they are about movies
  - Their page isn't very important, so it won't be ranked high for shirts or movies
- **Example:**
  - Shirt-seller creates 1,000 pages, each links to theirs with "movie" in the anchor text
  - These pages have no links in, so they get low PageRank
  - So the shirt-seller can't beat truly important movie pages, like IMDB

Why Does It NOT Work?




SPAM FARMING

Google vs. Spammers: Round 2!

- Once Google became the dominant search engine, spammers began to work out ways to fool Google
- **Spam farms** were developed to concentrate PageRank on a single page
- **Link spam:**
  - Create link structures that boost PageRank of a particular page

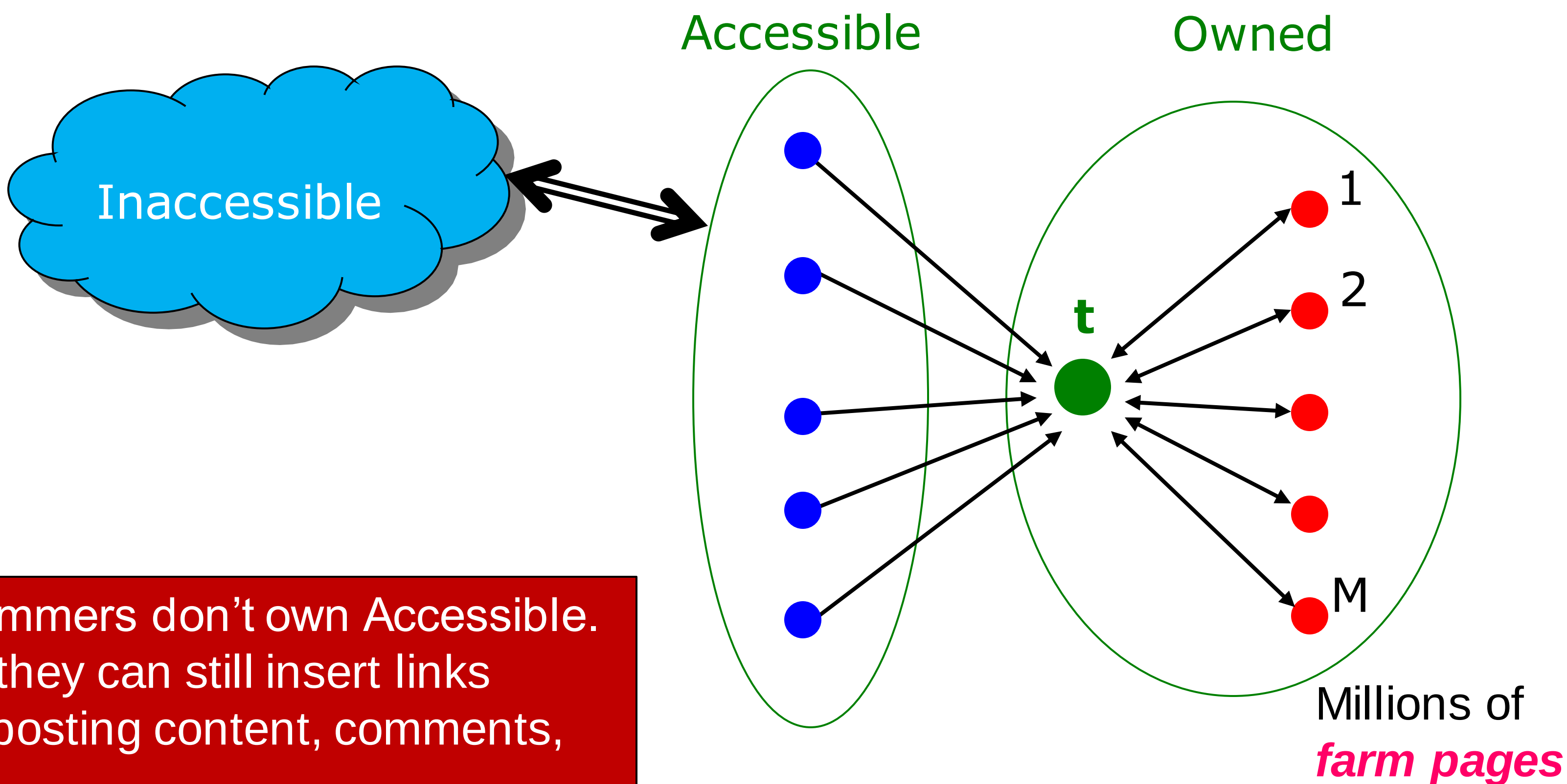
Link Spamming

- **Three kinds of web pages from a spammer's point of view**
  - **Inaccessible pages**
  - **Accessible pages**
    - e.g., blog comments pages
    - spammer can post links to his pages
  - **Owned pages**
    - Completely controlled by spammer
    - May span multiple domain names

Link Farms

- **Spammer's goal:**
  - Maximize the PageRank of target page ***t***
- **Technique:**
  - Get as many links from accessible pages as possible to target page ***t***
  - Construct "link farm" to get PageRank multiplier effect

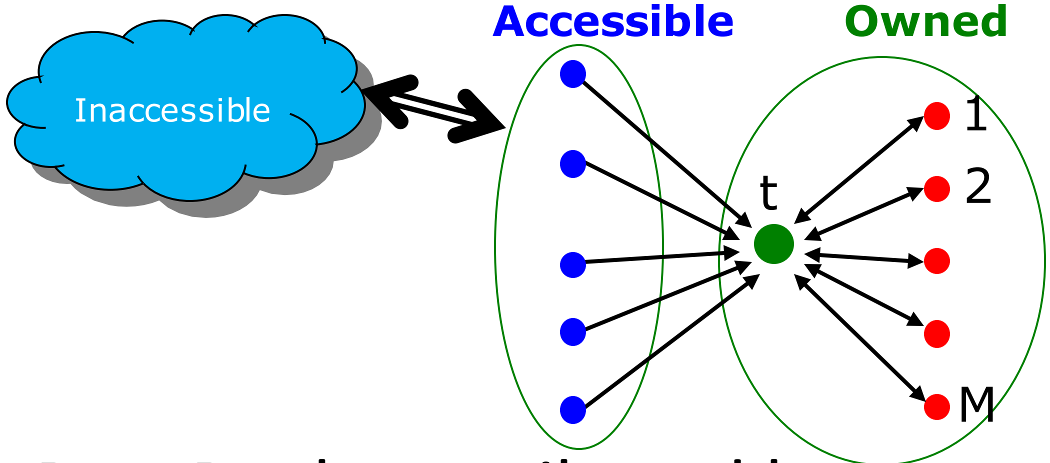
Link Farms



Spammers don't own Accessible. But they can still insert links (by posting content, comments, etc.)

One of the most common and effective organizations for a link farm

Analysis



N…# pages on the web
M…# of pages spammer owns

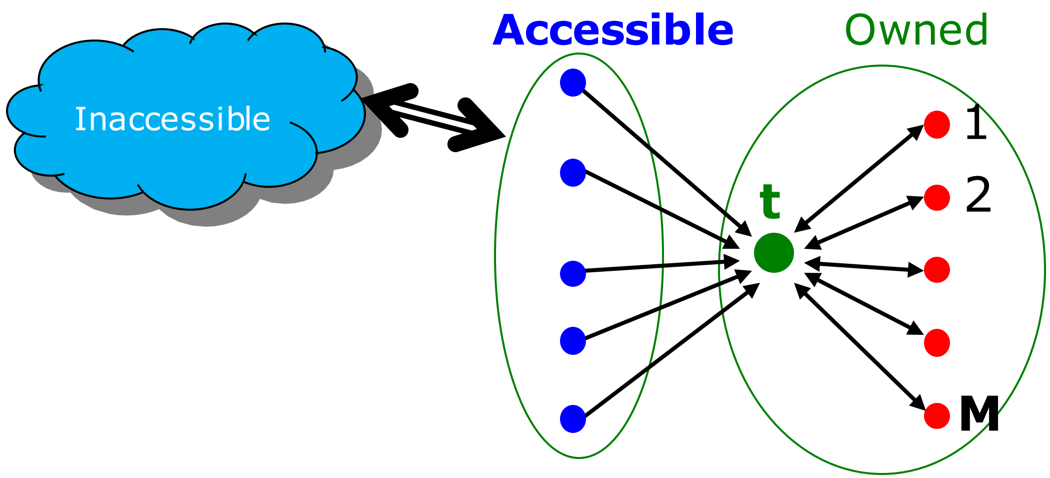
- **x**: PageRank contributed by accessible pages
- **y**: PageRank of target page ***t***
- Rank of each "owned" page $= \frac{\beta \boldsymbol{y}}{M} + \frac{1-\beta}{N}$
- $\boldsymbol{y} = x + \beta M \left[\frac{\beta \boldsymbol{y}}{M} + \frac{1-\beta}{N}\right] + \frac{1-\beta}{N}$

  $= x + \beta^2 y + \frac{\beta(1-\beta)M}{N} + \boxed{\frac{1-\beta}{N}}$

Very small; ignore
Now we solve for ***y***

- $\boldsymbol{y} = \frac{\boldsymbol{x}}{\boldsymbol{1-\beta^2}} + \boldsymbol{c}\frac{\boldsymbol{M}}{\boldsymbol{N}}$ where $c = \frac{\beta}{1+\beta}$

Analysis



N…# pages on the web
M…# of pages spammer owns

- $y = \frac{x}{1-\beta^2} + c\frac{M}{N}$ where $c = \frac{\beta}{1+\beta}$
- For $\beta = 0.85$, $1/(1-\beta^2) = 3.6$
- Multiplier effect for acquired PageRank
- By making ***M*** large, we can make ***y*** as **large as we want**

TrustRank: Combating Spam on the Web

Combating Spam

Two ways to Combat link spam:

- **Detection and blacklisting of structures that look like spam farms**
  - Leads to another war – hiding and detecting spam farms
- **TrustRank** = topic-specific PageRank with a teleport set of **trusted pages**
  - Example: .edu domains, .gov domains
  - similar domains for non-US websites

TrustRank: Idea

- **TrustRank is Topic-Specific PageRank**
  - **Topic** = the set of trustworthy pages
  - It is rare for a “good” page to point to a “bad” (spam) page

- **To develop a suitable teleport set:**
  1. Sample a set of **seed pages** from the web
  2. Have an **oracle (human)** to identify the good pages and the spam pages in the seed set
     - **Expensive task,** so we must make seed set as small as possible

Trust Propagation

- Call the subset of seed pages that are identified as **good** the **trusted pages**
- Perform a topic-sensitive PageRank with **teleport set = trusted pages**
  - **Propagate trust through links:**
    - Each page gets a trust value between **0** and **1**
- Solution 1: **Use a threshold value and mark all pages below the trust threshold as spam**

Approaches to Picking Seed Set

- Suppose we want to pick a seed set of ***k*** pages
- **How to do that?**
- **(1) PageRank:**
  - Pick the top ***k*** pages by PageRank
  - Theory is that bad pages can't get really high ranks
- **(2) Use trusted domains** whose membership is controlled, like .edu, .mil, .gov

Picking the Seed Set

- **Two conflicting considerations:**
  - Human has to inspect each seed page, so seed set must be as small as possible
  - Must ensure every **good page** gets adequate trust rank, so need to make all good pages reachable from seed set by short paths

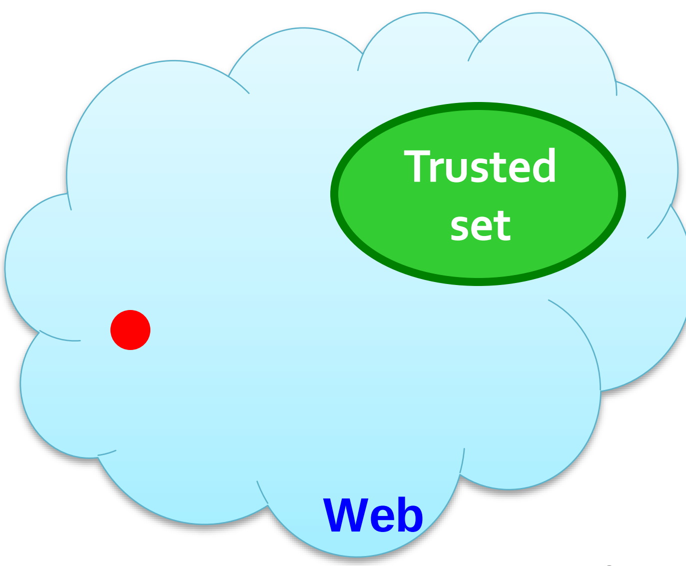
Why is it a good idea?

- **Trust attenuation:**
  - The degree of trust conferred by a trusted page decreases with the distance in the graph
- **Trust splitting:**
  - The larger the number of out-links from a page, the less scrutiny the page author gives each out-link
  - Trust is **split** across out-links

Spam Mass

Spam Mass

- In the **TrustRank** model, we start with good pages and propagate trust
- **Complementary view:**
  What fraction of a page's PageRank comes from **spam** pages?
- In practice, we don't know all the spam pages, so we need to estimate



Spam Mass Estimation

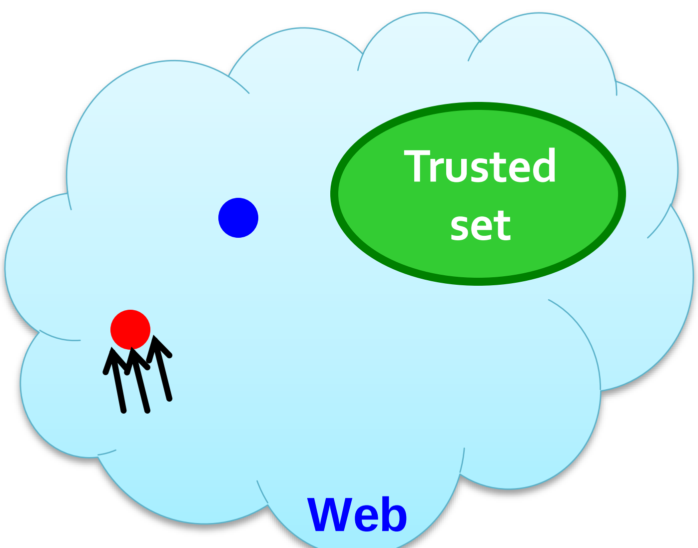
Solution 2:

- r_p = PageRank of page $\boldsymbol{p}$
- r_p^+ = PageRank of $\boldsymbol{p}$ with teleport into **trusted** pages only

- **Then:** What fraction of a page's PageRank comes from **spam** pages?

$$r_p^- = r_p - r_p^+$$

- **Spam mass of p = $\frac{r_p^-}{r_p}$**
  - Pages with high spam mass are spam



Summary of Today's lecture

- Topic specific PageRank
  - Custom teleportation vector
- Random Walk with Restarts
  - Recommendations
- Spam farming
- TrustRank and Spam Mass estimation

Extras

Trust Propagation: Simple Model

- **Set trust of each trusted page to 1**
- Suppose trust of page $\boldsymbol{p}$ is $\boldsymbol{t_p}$
  - Page $\boldsymbol{p}$ has a set of out-links $\boldsymbol{o_p}$
- For each $\boldsymbol{q} \in \boldsymbol{o_p}$, $\boldsymbol{p}$ **confers the trust** to $\boldsymbol{q}$
  - $\beta\, t_p / |o_p|$ for $0 < \beta < 1$
- **Trust is additive**
  - Trust of $\boldsymbol{p}$ is the sum of the trust conferred on $\boldsymbol{p}$ by all its in-linked pages
- **Note similarity to Topic-Specific PageRank**
  - Within a scaling factor, **TrustRank = PageRank** with trusted pages as teleport set