Note to other teachers and users of these slides: We would be delighted if you found our material useful for giving your own lectures. Feel free to use these slides verbatim, or to modify them to fit your own needs. If you make use of a significant portion of these slides in your own lecture, please include this message, or a link to our web site: http://www.mmds.org

Link Analysis: TrustRank and WebSpam

CS246: Mining Massive Datasets
Jure Leskovec, Stanford University
Mina Ghashami, Amazon
http://cs246.stanford.edu
Example: PageRank Scores

A 3.3

B 38.4

C 34.3

D 3.9

E 8.1

F 3.9

1.6 1.6 1.6

1.6 1.6 1.6
Random Teleports ($\beta = 0.8$)

\[
\begin{align*}
\mathbf{M} &= \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} + 0.2 \\
\mathbf{[1/N]}_{N \times N} &= \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix} \\
\mathbf{A} &= \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
y &= 1/3 & 0.33 & 0.28 & 0.26 & 7/33 \\
a &= 1/3 & 0.20 & 0.20 & 0.18 & \ldots & 5/33 \\
m &= 1/3 & 0.46 & 0.52 & 0.56 & 21/33 \\
r &= \mathbf{A} r
\end{align*}
\]
Input: Graph \( G \) and parameter \( \beta \)
- Directed graph \( G \) (can have spider traps and dead ends)
- Parameter \( \beta \)

Output: PageRank vector \( r \)

Set: \( r_j^{(0)} = \frac{1}{N}, t = 1 \)

Do: \( \forall j: r'_j = \sum_{i \rightarrow j} \beta \frac{r_i^{(t-1)}}{d_i} \)
- \( r'_j = 0 \) if in-degree of \( j \) is 0
- Now re-insert the leaked PageRank:
  \( \forall j: r_j^{(t)} = r'_j + \frac{1-S}{N} \)
- \( t = t + 1 \)

while \( \sum_j \left| r_j^{(t)} - r_j^{(t-1)} \right| < \varepsilon \)

If the graph has no dead-ends then the amount of leaked PageRank is \( 1-\beta \). But since we have dead-ends the amount of leaked PageRank may be larger. We have to explicitly account for it by computing \( S \).
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
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
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, she picks a page from a set \( S \)
- \( S \) contains only pages that are relevant to the topic
  - E.g., Open Directory (DMOZ) pages for a given topic/query
- For each teleport set \( S \), we get a different vector \( r_s \)
To make this work all we need is to update the teleportation part of the PageRank formulation:

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

- \( A \) 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
Suppose $S = \{1\}$, $\beta = 0.8$

<table>
<thead>
<tr>
<th>Node</th>
<th>Iteration</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>...</th>
<th>stable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.25</td>
<td>0.4</td>
<td>0.28</td>
<td>...</td>
<td>0.294</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0.25</td>
<td>0.1</td>
<td>0.16</td>
<td>...</td>
<td>0.118</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0.25</td>
<td>0.3</td>
<td>0.32</td>
<td>...</td>
<td>0.327</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0.25</td>
<td>0.2</td>
<td>0.24</td>
<td>...</td>
<td>0.261</td>
</tr>
</tbody>
</table>

$S=\{1\}$, $\beta=0.9$: 
$r=[0.17, 0.07, 0.40, 0.36]$ 

$S=\{1,2\}$, $\beta=0.8$: 
$r=[0.26, 0.20, 0.29, 0.23]$ 

$S=\{1\}$, $\beta=0.7$: 
$r=[0.39, 0.14, 0.27, 0.19]$ 

$S=\{1,2,3,4\}$, $\beta=0.8$: 
$r=[0.13, 0.10, 0.39, 0.36]$ 

$S=\{1,2,3\}$, $\beta=0.8$: 
$r=[0.17, 0.13, 0.38, 0.30]$ 

$S=\{1,2\}$, $\beta=0.8$: 
$r=[0.26, 0.20, 0.29, 0.23]$ 

$S=\{1\}$, $\beta=0.8$: 
$r=[0.29, 0.11, 0.32, 0.26]$
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
a.k.a.: Relevance, Closeness, ‘Similarity’…
Good proximity measure?

- **Shortest path is not good:**

  ![Diagram](image1)

  ![Diagram](image2)

- **No effect of degree-1 nodes (E, F, G)!**
- **Multi-faceted relationships**
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
  - Direct and indirect connections
  - Degree of the node
Pixie: Random Walk-based Real-Time Recommender System at Pinterest

Recommendsations can be radically personalized.

- Adapting in real-time
- Opportunity for human centered personalization.
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:

Chocolate Strawberry Shake
This healthier chocolate strawberry shake is like sipping a...
One Lovely Life
Danielle Benzaïs
Strawberries
From Pins to Pins

- Pins to Pins

**Input:**

- Healthy Chocolate Strawberry Shake
  - Chocolate Dipped Strawberry Smoothie
  - Chocolate Dipped Strawberry Smoothie. Just in time for...
  - Be Whole, Be You.

**Output:**

- Tropical Orange Smoothie
  - 8 Staple Smoothies (That You Should Know How to Make)
  - 8 Staple Smoothies You Should Know How to Make
  - Easy Breezy Tropical Orange Smoothie

- Quick & Nutritious Vanilla Pumpkin Smoothie
  - Spinach-Pear-Celery Smoothie
  - The Perfect Vanilla Pumpkin Smoothie: A Quick &...

2/3/2022

Jure Leskovec & Mina Ghashami, Stanford C246: Mining Massive Datasets
From Pins to Pins

Input:

- Chocolate Strawberry Shake
  - This healthier chocolate strawberry shake is like sipping a...
  - One Lovely Life
  - Danielle Bevanis
  - Strawberries

- Healthy Chocolate Peanut Butter Chip Muffins
  - Healthy Chocolate Peanut Butter Chip Muffins made with greek...
  - The First Year
  - Katie - You Brew... Healthy Recipes

- The Ultimate Healthy Soft & Chewy Chocolate Chip Cookies
  - The ULTIMATE Healthy Chocolate Chip Cookies -- so buttery...
  - Amy's Healthy Baking
  - Robin Guerlin
  - healthy cooking
Pinterest is a Giant Bipartite Graph
Bipartite Pin And Board Graph
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:
Proximity to query node(s) $Q$:

$$\text{ALPHA} = 0.5$$

$$\text{QUERY\_NODES} = \{ \text{bipar}, \text{bodagraph} \}$$

```python
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()
```
Pixel Random Walk Algorithm

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

  $\text{ALPHA} = 0.5$

  $\text{QUERY\_NODES} =$

  ```python
  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 = \( \text{PersonalizedNeighbor}(E,U) \), where \( E \) is edge and \( U \) is the user.
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
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.
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.

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.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Number</th>
<th>Size</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin Nodes</td>
<td>3 Billion</td>
<td>8 Bytes</td>
<td>24 GiB</td>
</tr>
<tr>
<td>Board Nodes</td>
<td>2 Billion</td>
<td>8 Bytes</td>
<td>16 GiB</td>
</tr>
<tr>
<td>Undirected Edges</td>
<td>20 Billion</td>
<td>8 Bytes</td>
<td>160 GiB</td>
</tr>
</tbody>
</table>
**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

Recommendations@Twitter

Joint work with many Twitter folks over several years:
http://www2013.w3c.br/proceedings/p505.pdf
Recommendations@Twitter

Who to follow

- Ramnath Balasubramanyan and 3 others follow
  Jiasong Sun
  @jiasong_sun
  Software Engineer @twitter

- Gilad Mishne and 5 others follow
  David Burkett
  @david_burkett
  Doesn't usually write well in the short form, but is glad that other people do.

- David Gleich and 2 others follow
  Nelly Litvak
  @nellylitvak
  Professor in Applied Mathematics at University of Twente and Eindhoven University of Technology; complex networks; novelty in education; non-fiction author

Suggested

- Serena Williams
  @serenawilliams

  - Venus Williams
    @Venuseswilliams
    Tennis player, big sister, grown up girl. Double Tap! Be Well #CoachVenus @elevenbyvenus workouts @link in bio

  - Rafa Nadal
    @RafaelNadal
    Tennis player
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**
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
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 termed 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
Our hypothetical shirt-seller loses

- Saying he is about movies doesn’t help, because others don’t say he is about movies
- His 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 his with “movie” in the anchor text
- These pages have no links in, so they get little PageRank
- So the shirt-seller can’t beat truly important movie pages, like IMDB
Why Does It NOT Work?

Biography of President George W. Bush
Biography of the president from the official White House web site.
www.whitehouse.gov/president/gwbbio.html - 29k - Cached - Similar pages
Past Presidents - Kids Only - Current News - President
More results from www.whitehouse.gov »

Welcome to MichaelMoore.com!
Official site of the gadfly of corporations, creator of the film Roger and Me and the television show The Awful Truth. Includes mailing list, message board, ...
www.michaelmoore.com/ - 35k - Sep 1, 2005 - Cached - Similar pages

BBC NEWS | Americas | 'Miserable failure' links to Bush
Web users manipulate a popular search engine so an unflattering description leads to the president’s page.
news.bbc.co.uk/2/hi/americas/3298443.stm - 31k - Cached - Similar pages

Google’s (and Inktomi’s) Miserable Failure
A search for miserable failure on Google brings up the official George W. Bush biography from the US White House web site. Dismissed by Google as not a ...
searchenginewatch.com/sereport/article.php/3296101 - 45k - Sep 1, 2005 - Cached - Similar pages
SPAM FARMING
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
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
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
Inaccessible

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

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**

- **x**: PageRank contributed by accessible pages
- **y**: PageRank of target page \( t \)
- Rank of each “owned” page \( = \frac{\beta y}{M} + \frac{1 - \beta}{N} \)

\[
y = x + \beta M \left[ \frac{\beta y}{M} + \frac{1 - \beta}{N} \right] + \frac{1 - \beta}{N} \\
= x + \beta^2 y + \frac{\beta (1 - \beta) M}{N} + \frac{1 - \beta}{N}
\]

Very small; ignore

Now we solve for \( y \)

\[
y = \frac{x}{1 - \beta^2} + c \frac{M}{N}
\]

where \( c = \frac{\beta}{1 + \beta} \)
\[ y = \frac{x}{1-\beta^2} + c \frac{M}{N} \] 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

- **Combating term spam**
  - Analyze text using statistical methods
  - Similar to email spam filtering
  - Also useful: Detecting approximate duplicate pages

- **Combating 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, similar domains for non-US schools
TrustRank: Idea

- **Basic principle:** *Approximate isolation*
  - It is rare for a “good” page to point to a “bad” (spam) page

- Sample a set of *seed pages* from the web

- 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
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
Set trust of each trusted page to 1

Suppose trust of page $p$ is $t_p$

- Page $p$ has a set of out-links $o_p$

For each $q \in o_p$, $p$ confers the trust to $q$

- $\beta t_p / |o_p|$ for $0 < \beta < 1$

Trust is additive

- Trust of $p$ is the sum of the trust conferred on $p$ by all its in-linked pages

Note similarity to Topic-Specific PageRank

- Within a scaling factor, TrustRank = PageRank with trusted pages as teleport set
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
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
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
TrustRank
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.
Solution 2:

- \( r_p \) = PageRank of page \( p \)
- \( r_p^+ \) = PageRank of \( p \) 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