COMP 345: Data Mining
More on PageRank

Slides Adapted From: www.mmds.org (Mining Massive Datasets)

Rhodes College

Reminder

- Assignment 6
  - due Wed. Nov. 14th/Thurs. Nov. 15th

2

PageRank: The Complete Algorithm

- **Input: Graph G and parameter β**
  - Directed graph G (can have **spider traps** and **dead ends**)
  - Parameter β
- **Output: PageRank vector r^{new}**
  - **Set:** $r_j^{old} = \frac{1}{N}$
  - **repeat until convergence:** $\sum_j |r_j^{new} - r_j^{old}| > \varepsilon$
    - $\forall j$: $r'^{new}_j = \sum_{i \to j} \beta \frac{r_i^{old}}{d_i}$
      $r'^{new}_j = 0$ if in-degree of j is 0
    - **Now re-insert the leaked PageRank:**
      $\forall j$: $r_j^{new} = r'^{new}_j + \frac{1-S}{N}$ where: $S = \sum_j r'^{new}_j$
    - $r^{old} = r^{new}$

If the graph has no dead-ends then the amount of leaked PageRank is **1-β**. But since we have dead-ends the amount of leaked PageRank may be larger. We have to explicitly account for it by computing **S**.

Sparse Matrix Encoding

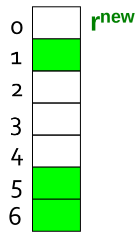
- **Encode sparse matrix using only nonzero entries**
  - Space proportional roughly to number of links
  - Say 10N, or 4*10*1 billion = 40GB
  - **Still won't fit in memory, but will fit on disk**

source node	degree	destination nodes
0	3	1, 5, 7
1	5	17, 64, 113, 117, 245
2	2	13, 23

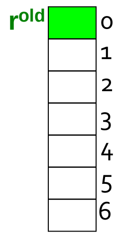
Basic Algorithm: Update Step

- **Assume enough RAM to fit r^{new} into memory**
  - Store r^{old} and matrix **M** on disk
- **1 step of power-iteration is:**

Initialize all entries of r^{new} = **(1-β) / N**
For each page ***i*** (of out-degree d_i):
  Read into memory: i, d_i, $dest_1$, …, $dest_{d_i}$, $r^{old}(i)$
  For **j = 1…d_i**
    $r^{new}(dest_j)$ += β $r^{old}(i)$ / d_i



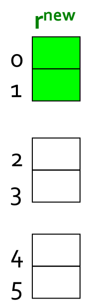
source	degree	destination
0	3	1, 5, 6
1	4	17, 64, 113, 117
2	2	13, 23



Analysis

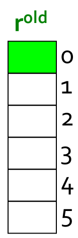
- **Assume enough RAM to fit r^{new} into memory**
  - Store r^{old} and matrix ***M*** on disk
- **In each iteration, we have to:**
  - Read r^{old} and ***M***
  - Write r^{new} back to disk
  - **Cost per iteration of Power method:**
    $= 2|r| + |M|$
- **Question:**
  - What if we could not even fit r^{new} in memory?

Block-based Update Algorithm



src	degree	destination
0	4	0, 1, 3, 5
1	2	0, 5
2	2	3, 4

M

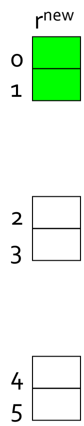


- Break r^{new} into k blocks that fit in memory
- Scan M and r^{old} once for each block

Analysis of Block Update

- **Similar to nested-loop join in databases**
  - Break r^{new} into k blocks that fit in memory
  - Scan M and r^{old} once for each block
- **Total cost:**
  - k scans of M and r^{old}
  - **Cost per iteration of Power method:**
    $k(|M| + |r|) + |r| = k|M| + (k+1)|r|$
- **Can we do better?**
  - **Hint:** M is much bigger than r (approx 10-20x), so we must avoid reading it k times per iteration

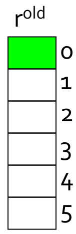
Block-Stripe Update Algorithm



src	degree	destination
0	4	0, 1
1	3	0
2	2	1

0	4	3
2	2	3

0	4	5
1	3	5
2	2	4



Break *M* into stripes! Each stripe contains only destination nodes in the corresponding block of r^{new}

Block-Stripe Analysis

- Break ***M*** into stripes
  - Each stripe contains only destination nodes in the corresponding block of r^{new}
- Some additional overhead per stripe
  - But it is usually worth it
- **Cost per iteration of Power method:**
  $=|M|(1+\varepsilon) + (k+1)|r|$

Some Problems with Page Rank

- **Measures generic popularity of a page**
  - Biased against topic-specific authorities
  - **Solution:** Topic-Specific PageRank
- **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 popularity, can we measure popularity within a topic?**
- **Goal:** Evaluate Web pages not just according to their popularity, but by how close they are to a particular topic, e.g. "sports" or "history"
- **Allows search queries to be answered based on interests of the user**
  - **Example:** Query "Trojan" wants different pages depending on whether you are interested in sports, history and computer security

Topic-Specific PageRank

- 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 walker teleports, she pick 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

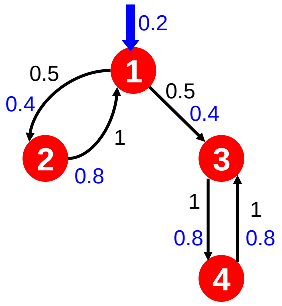
Matrix Formulation

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

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

  - A is stochastic!
- 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
  - 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=0.90:
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=0.70:
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 Topic Vector 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, …

PageRank: Summary

- **"Normal" PageRank:**
  - Teleports uniformly at random to any node
  - All nodes have the same probability of surfer landing there: **S =** [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
- **Topic-Specific PageRank** also known as **Personalized PageRank:**
  - Teleports to a topic specific set of pages
  - Nodes can have different probabilities of surfer landing there: **S =** [0.1, 0, 0, 0.2, 0, 0, 0.5, 0, 0, 0.2]
- **Random Walk with Restarts:**
  - Topic-Specific PageRank where teleport is always to the same node. S=[0, 0, 0, 0, **1**, 0, 0, 0, 0, 0, 0]

TrustRank: Combating the Web Spam

What is Web Spam?

- **Spamming:**
  - Any deliberate action to boost a web page's position in search engine results, incommensurate with 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 first in the listings
  - Copy it into your page, make it "invisible"
- **These and similar techniques are term spam**

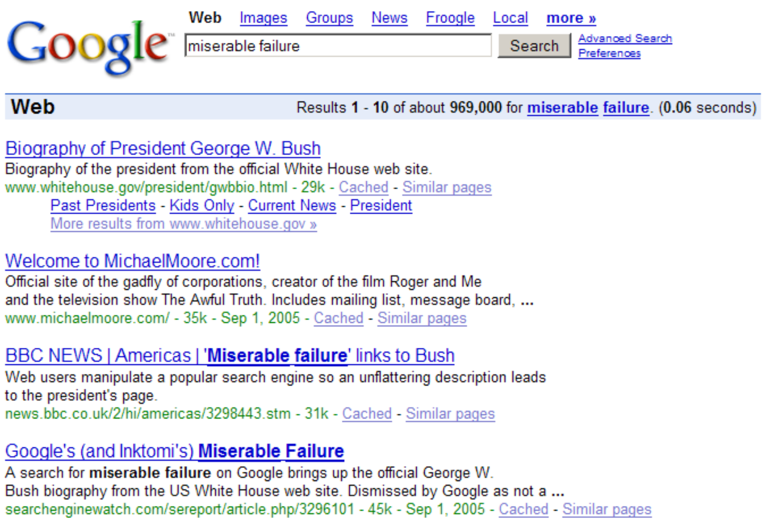
Google's Solution to Term Spam

- **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 It Works?

- **Our hypothetical shirt-seller looses**
  - 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 it does not work?



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:**
  - Creating link structures that boost PageRank of a particular page

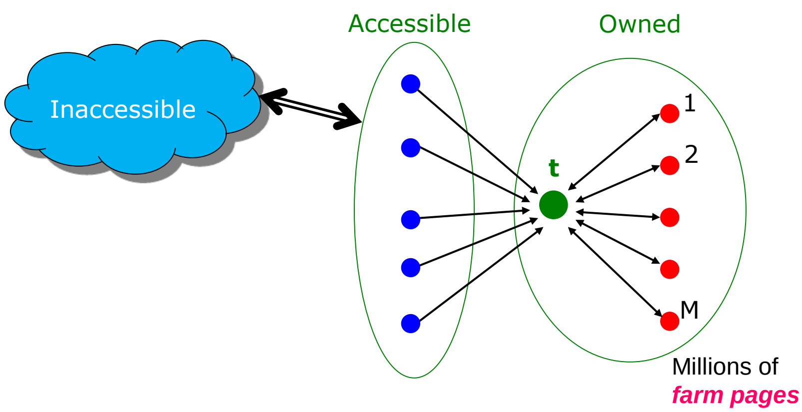
Link Spamming

- **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

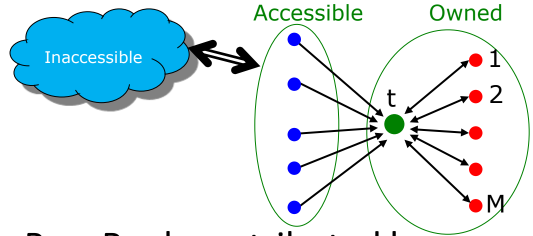
- **Spammer's goal:**
  - Maximize the PageRank of target page ***t***
- **Technique:**
  - Get as many links from accessible pages as possible to target page ***t***
  - Construct "link farm" to get PageRank multiplier effect

Link Farms



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

Analysis



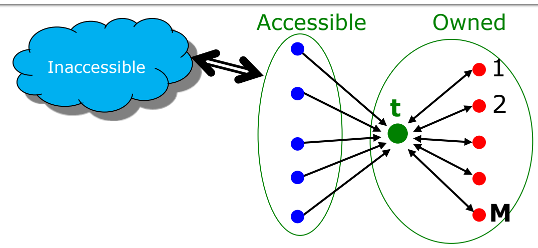
N…# pages on the web
M…# of pages spammer owns

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

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

  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

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

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

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**

Simple Model: Trust Propagation

- 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

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

Picking the Seed Set

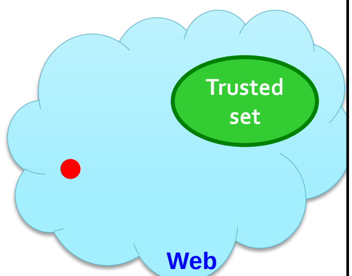
- **Two conflicting considerations:**
  - Human has to inspect each seed page, so seed set must be as small as possible
  - Must ensure every **good page** gets adequate trust rank, so need make all good pages reachable from seed set by short paths

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 you can't get a bad page's rank really high
- **(2) Use trusted domains** whose membership is controlled, like .edu, .mil, .gov

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 ***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.