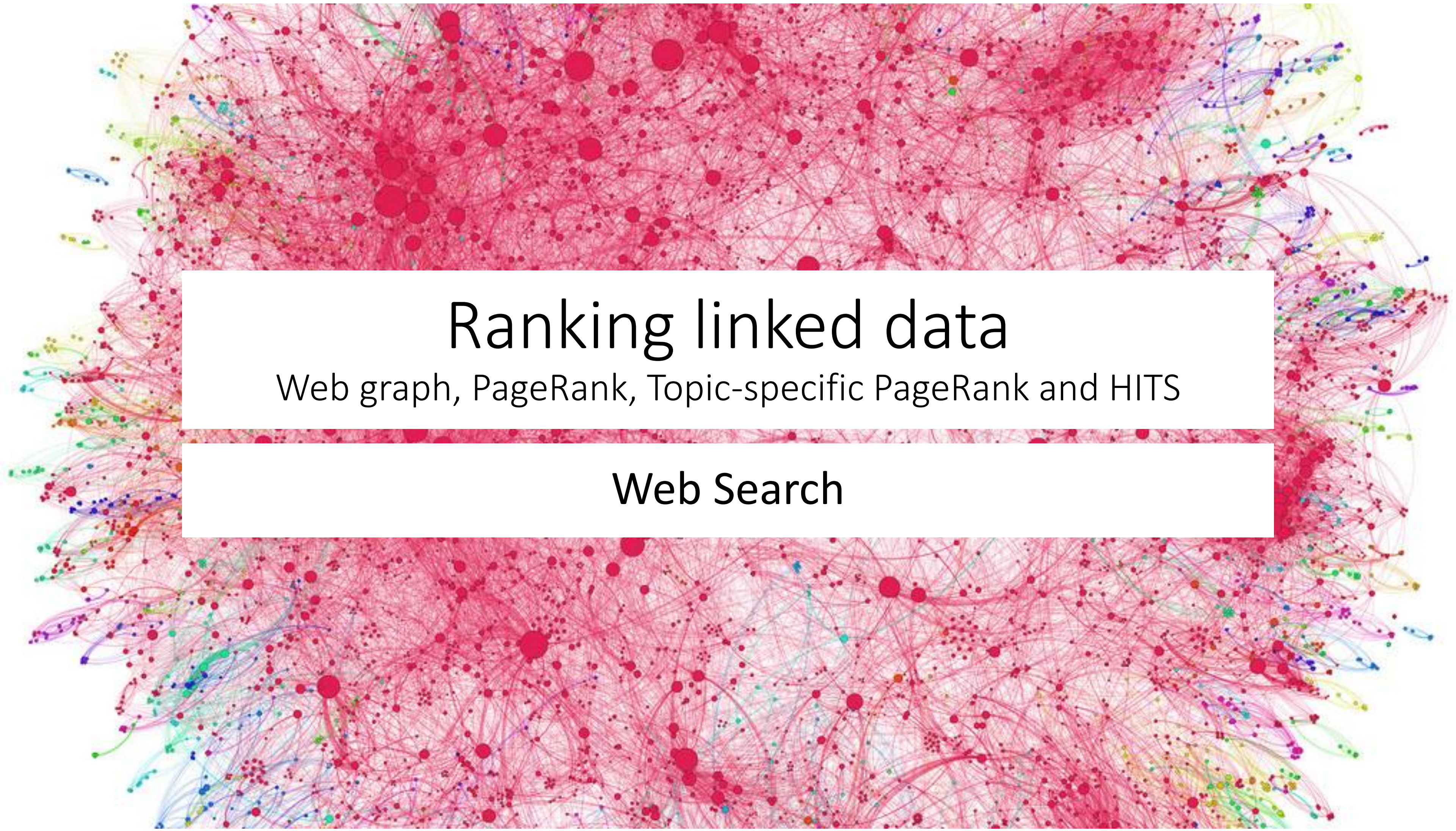


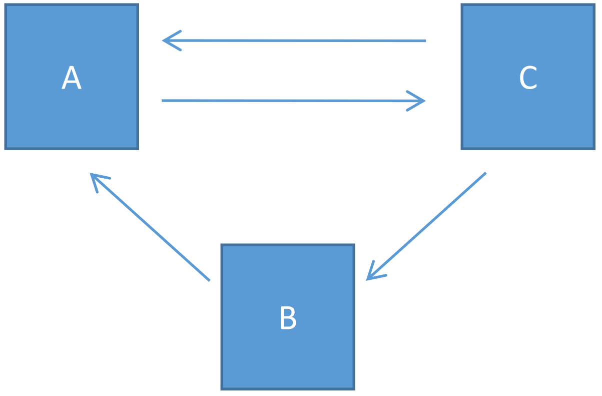
Ranking linked data

Web graph, PageRank, Topic-specific PageRank and HITS

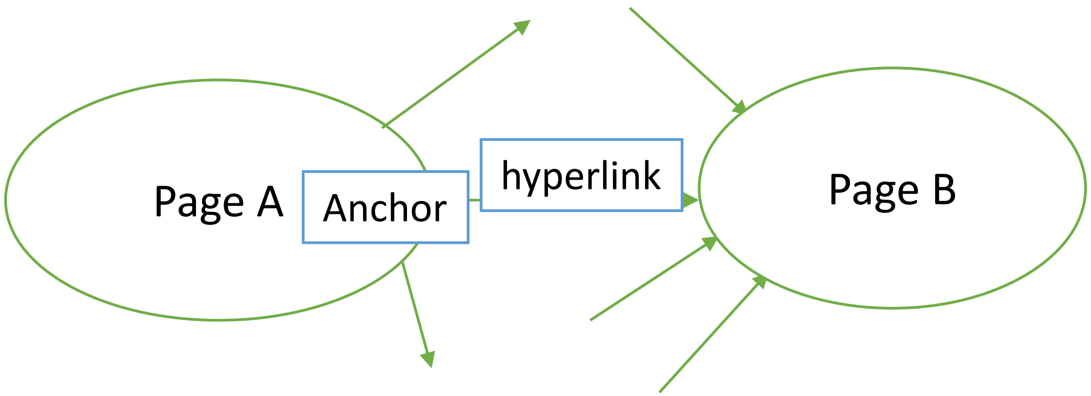
Web Search

Ranking linked data

- Links are inserted by humans.
- **They are one of the most valuable judgments of a page's importance.**
- A link is inserted to denote an association. The anchor text describes the type of association.



The Web as a directed graph

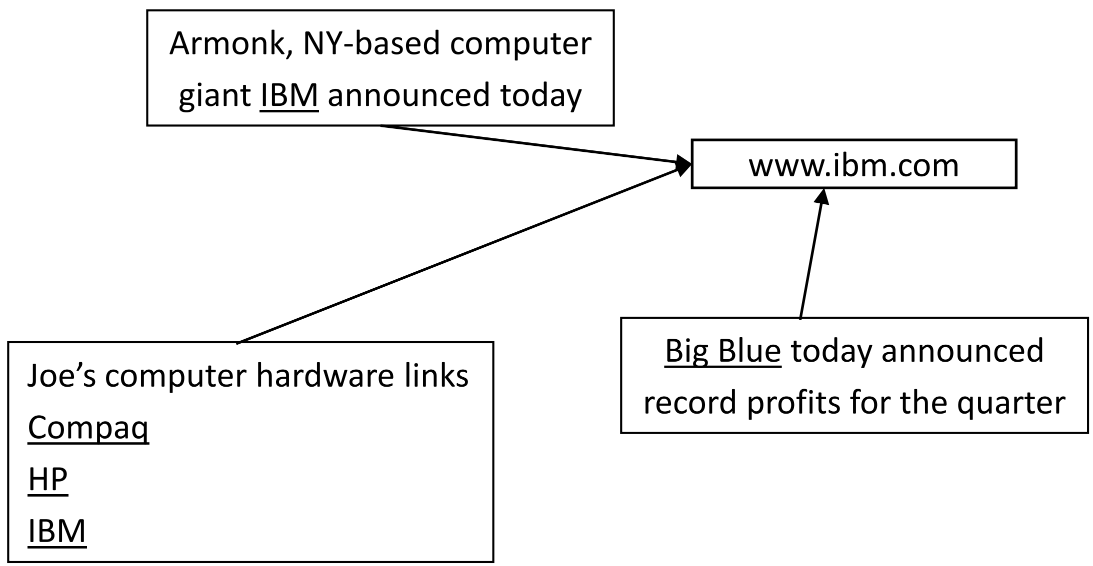


Assumption 1: A hyperlink between pages denotes author perceived relevance (quality signal)

Assumption 2: The anchor of the hyperlink describes the target page (textual context)

Anchor text

- When indexing a document *D*, include anchor text from links pointing to *D*.



Indexing anchor text

Sec. 21.1.1

- Can sometimes have unexpected side effects - *e.g., evil empire.*

- Can boost anchor text with weight depending on the authority of the anchor page's website

  - E.g., if we were to assume that content from cnn.com or yahoo.com is authoritative, then trust the anchor text from them

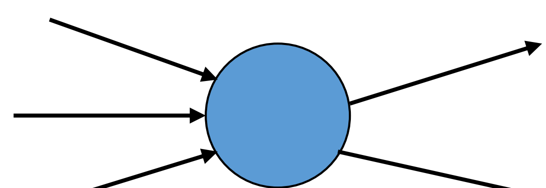
Citation analysis

- Citation frequency
- Co-citation coupling frequency
  - Co-citations with a given author measures “impact”
  - Co-citation analysis [Mcca90]
- Bibliographic coupling frequency
  - Articles that co-cite the same articles are related
- Citation indexing
  - Who is author cited by? [Garf72]
- PageRank preview: Pinsker and Narin ’60s

Incoming and outgoing links

- The popularity of a page is related to the number of incoming links
  - Positively popular
  - Negatively popular

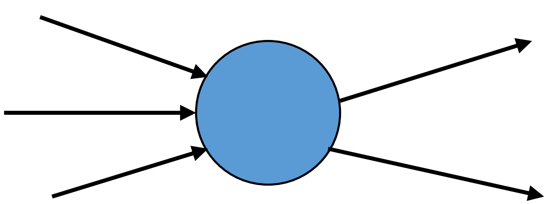
- The popularity of a page is related to the popularity of pages pointing to them



Query-independent ordering

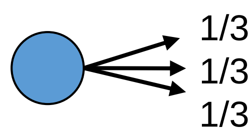
- First generation: using link counts as simple measures of popularity.

- Two basic suggestions:
  - Undirected popularity:
    - Each page gets a score = the number of in-links plus the number of out-links (3+2=5).
  - Directed popularity:
    - Score of a page = number of its in-links (3).



PageRank scoring

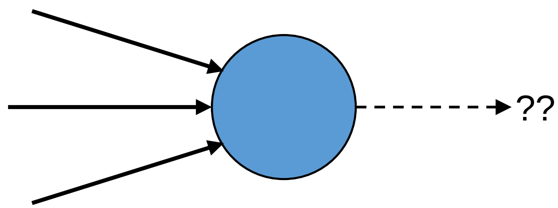
- Imagine a browser doing a random walk on web pages:
  - Start at a random page
  - At each step, go out of the current page along one of the links on that page, equiprobably



- "In the steady state" each page has a long-term visit rate - use this as the page's score.

Not quite enough

- The web is full of dead-ends.
  - Random walk can get stuck in dead-ends.
  - Makes no sense to talk about long-term visit rates.



Teleporting

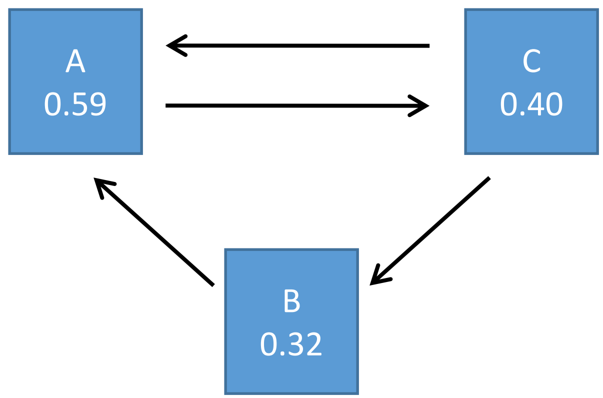
- At a dead end, jump to a random web page.

- At any non-dead end, with probability 10%, jump to a random web page.
  - With remaining probability (90%), go out on a random link.
  - 10% - a parameter.

- Result of teleporting:
  - Now cannot get stuck locally.
  - There is a long-term rate at which any page is visited.
  - How do we compute this visit rate?

The random surfer

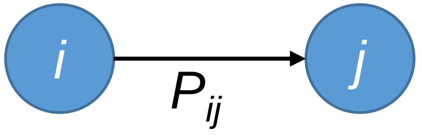
- The PageRank of a page is the probability that a given random “Web surfer” is currently visiting that page.



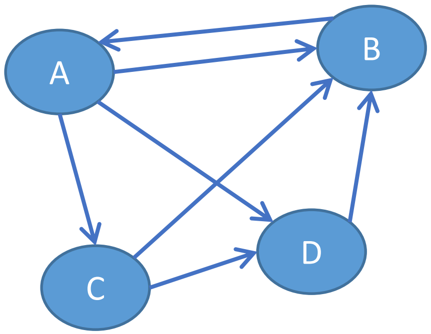
- This probability is related to the incoming links and to a certain degree of browsing randomness (e.g. reaching a page through a search engine).

Markov chains

- A Markov chain consists of n states, plus an $n \times n$ transition probability matrix **P**.
- At each step, we are in exactly one of the states.
- For $1 \leq i,j \leq n$, the matrix entry P_{ij} tells us the probability of j being the next state, given we are currently in state i.



Transitions probability matrix

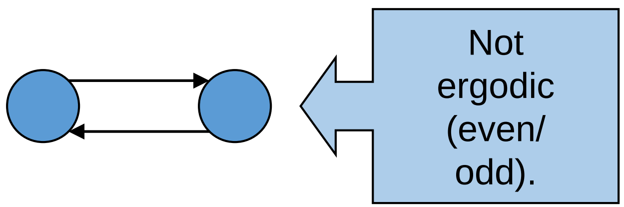


	A	B	C	D
A	0	1	1	1
B	1	0	0	0
C	0	1	0	1
D	0	1	0	0

	A	B	C	D
A	0	P_{ab}	P_{ac}	P_{ad}
B	P_{ba}	0	0	0
C	0	P_{cb}	0	P_{cd}
D	0	P_{db}	0	0

Ergodic Markov chains

- A Markov chain is ergodic if
  - you have a path from any state to any other
  - For any start state, after a finite transient time T_0, the probability of being in any state at a fixed time $T>T_0$ is nonzero.



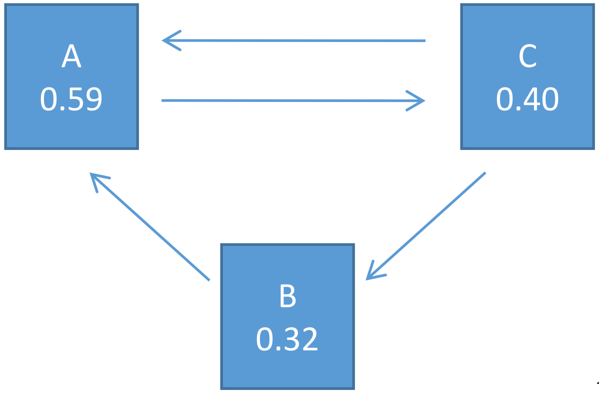
Ergodic Markov chains

- For any ergodic Markov chain, there is a unique long-term visit rate for each state.
  - Steady-state probability distribution.
- Over a long time-period, we visit each state in proportion to this rate.
- It doesn't matter where we start.

The PageRank of Web page ***i*** corresponds to the probability of being at page ***i*** after an infinite random walk across all pages (i.e., the stationary distribution).

PageRank

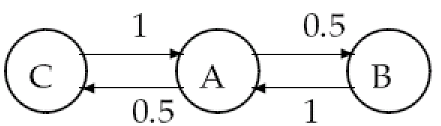
- The rank of a page is related to the number of incoming links of that page and the rank of the pages linking to it.



$$PR(A) = \frac{(1-d)}{N-1} + d \cdot \left[\frac{PR(B)}{OL(B)} + \frac{PR(C)}{OL(C)}\right]$$

PageRank: formalization

- The RandomSurfer model assumes that the pages with more inlinks are visited more often

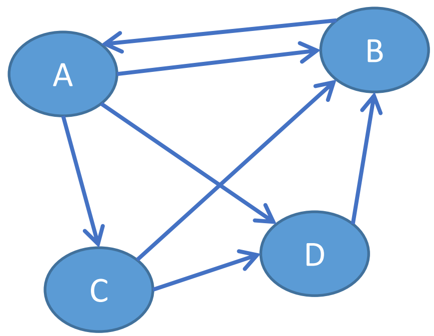


- The rank of a page *j* is computed as:

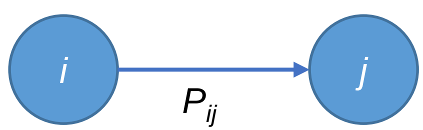
$$\Pr(p_j) = \frac{1-d}{N-1} + d \sum_{i \in Parents(j)} \frac{\Pr(p_i)}{OL_i}$$

where OL_i is the number of outgoing links of page i and p_i is the PageRank of that page

Transitions probability matrix



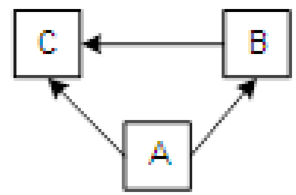
	A	B	C	D
A	0	1	1	1
B	1	0	0	0
C	0	0	1	1
D	0	1	0	0



	A	B	C	D
A	0	P_{ab}	P_{ac}	P_{ad}
B	P_{ba}	0	0	0
C	0	0	P_{cc}	P_{cd}
D	0	P_{db}	0	0

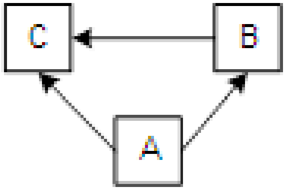
Example

- Consider three Web pages:



- Compute the PageRank of each page on the following Web graph. Consider d = 0.9.

$$\Pr(p_j) = \frac{1-d}{N-1} + d \sum_{i \in Parents(j)} \frac{\Pr(p_i)}{OL_i}$$



$$\Pr(p_j) = \frac{1-d}{N-1} + d \sum_{i \in Parents(j)} \frac{\Pr(p_i)}{OL_i}$$

Page	Iteration 0	Iteration 1	Iteration 2
A			
B			
C			

PageRank: issues and variants

- How realistic is the random surfer model?
  - What if we modeled the back button? [Fagi00]
  - Surfer behavior sharply skewed towards short paths [Hube98]
  - Search engines, bookmarks & directories make jumps non-random.

- Biased Surfer Models
  - Weight edge traversal probabilities based on match with topic/query (non-uniform edge selection)
  - Bias jumps to pages on topic (e.g., based on personal bookmarks & categories of interest)

Topic Specific PageRank [Have02]

- The idea is that links between pages of the same category are more likely to be followed.

- Conceptually, we use a random surfer who teleports, with ~10% probability, using the following rule:
  - Selects a category (say, one of the 16 top level categories) based on a query & user -specific distribution over the categories
  - Teleport to a page uniformly at random within the chosen category.
  - Do not teleport to a pages outside the chosen category.

Topic Specific PageRank - Implementation

- **offline**: Compute pagerank distributions wrt individual categories
  - Query independent model as before
  - Each page has multiple pagerank scores – one for each category, with teleportation only to that category

- **online**: Distribution of weights over categories computed by query context classification
  - Generate a dynamic pagerank score for each page - weighted sum of category-specific pageranks

Offline: Web page topic classifier

- Web pages have specific topics that can be detected by some classifier.

- Links are more likely between topics of the same topic.

- Links between pages of the same topic are more likely to be followed.

fastText https://fasttext.cc/docs/en/english-vectors.html

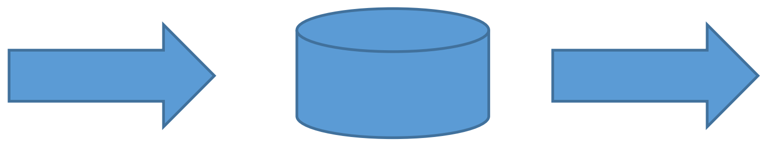
Offline: Topic Specific PageRank

- Compute the PageRank of each page for each topic.
- Key difference: The teleporting is canceled between pages of different topics.

Page	PR_Health	PR_Sports	PR_Travel	PR_Business
1				
2 (sports)				
3				
…				

Online: Query topic classification

Query

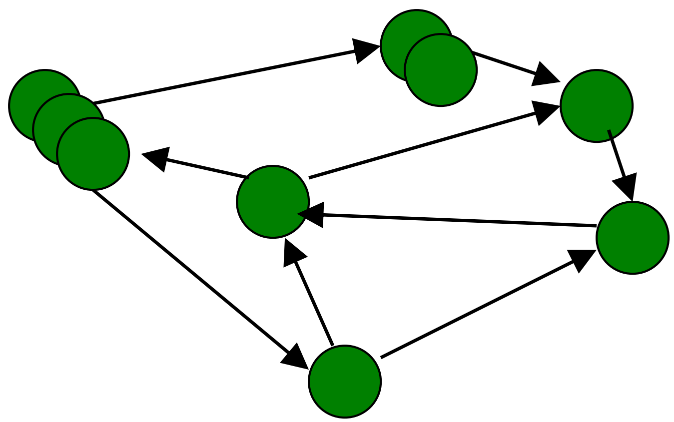


Doc 1	Sports
Doc 2	Health
Doc 3	Sports
Doc 4	Sports
Doc 5	Sports

Query category = 80% sports + 20% health

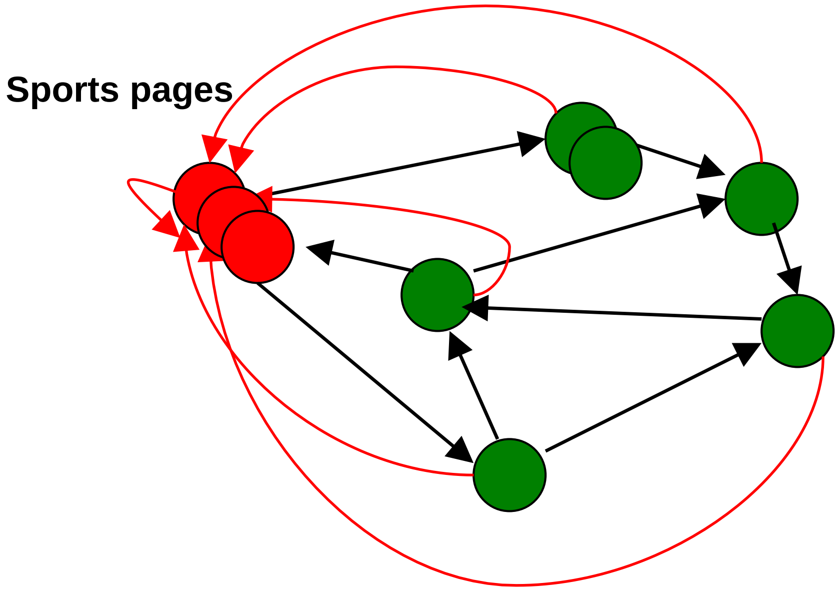
Example

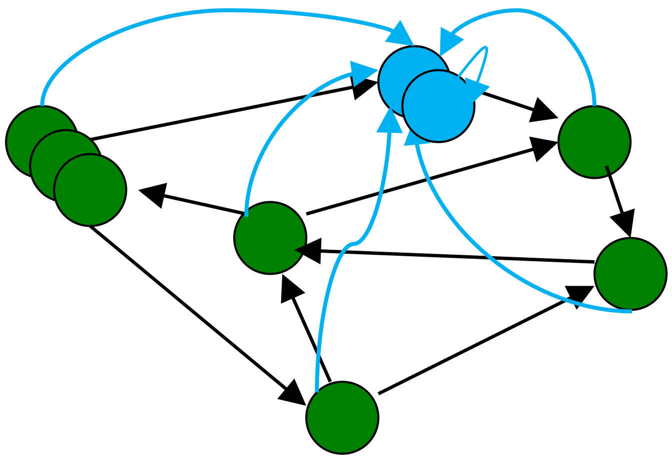
- Consider a query on a given set of Web pages with the following graph:



- The query has **80%** probability of being about **Sports**.
- The query has **20%** probability of being about **Health**.

Non-uniform Teleportation





$$\mathrm{PR}_{\mathrm{Sports}}(p_3) = \frac{1-d}{N-1} + d \sum_{i \in Parents(p_3)} \frac{\Pr(p_i)}{OL_i}$$

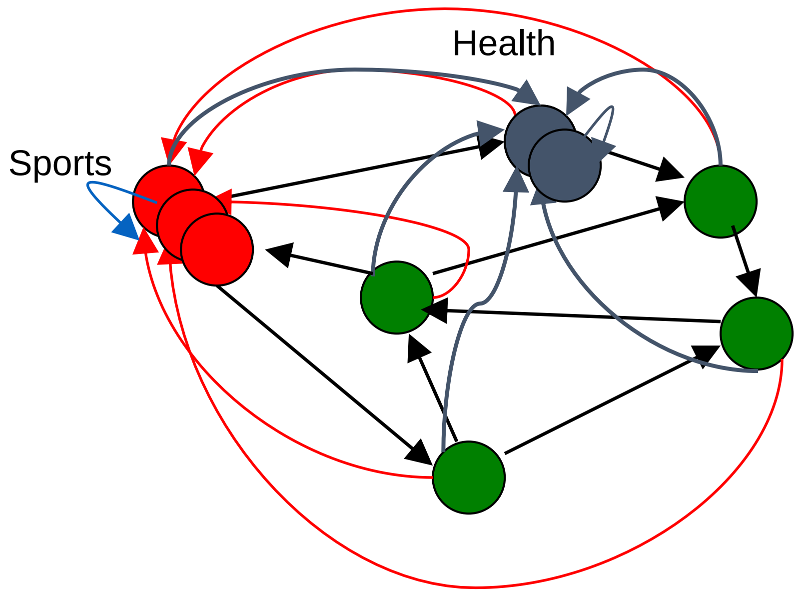
$$\mathrm{PR}_{\mathrm{Sports}}(p_5) = \sum_{i \in Parents(p_5)} \frac{\Pr(p_i)}{OL_i}$$

$$\mathrm{PR}_{\mathrm{Health}}(p_3) = \sum_{i \in Parents(p_3)} \frac{\Pr(p_i)}{OL_i}$$

$$\mathrm{PR}_{\mathrm{Health}}(p_5) = \frac{1-d}{N-1} + d \sum_{i \in Parents(p_5)} \frac{\Pr(p_i)}{OL_i}$$

Interpretation

Query has 80% sports probability and 20% health probability



$$pr = (0.8\ PR_{sports} + 0.2\ PR_{health})$$

80% Sports PageRank, 20% Health PageRank

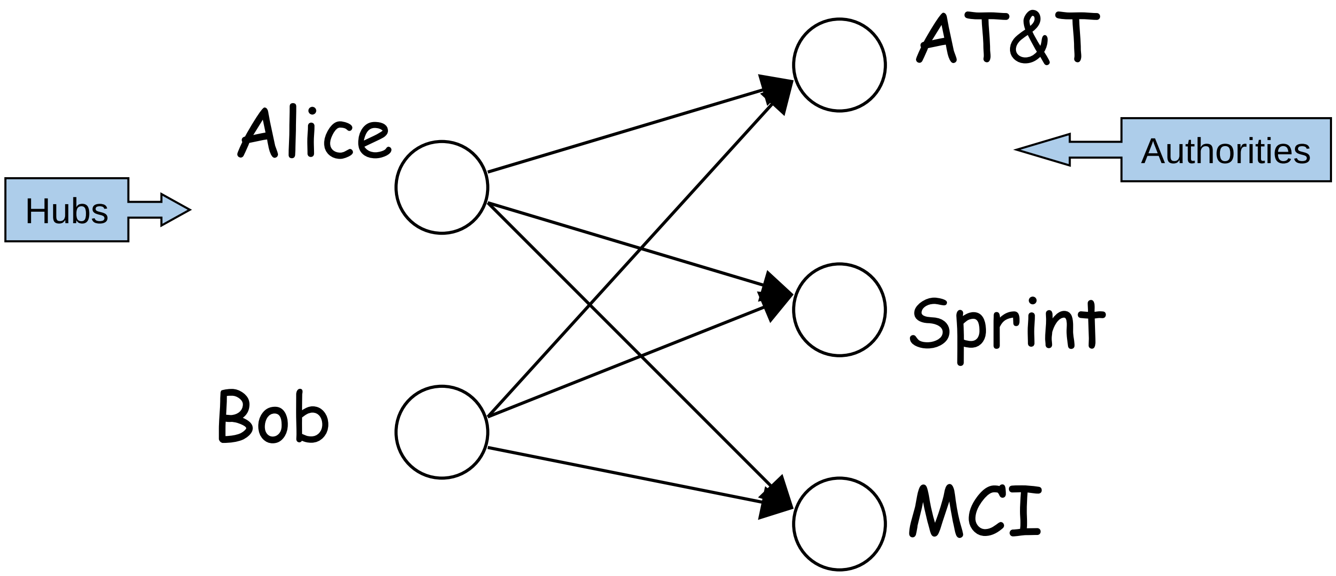
Hyperlink-Induced Topic Search (HITS) - Klei98

- In response to a query, instead of an ordered list of pages each meeting the query, find <u>two</u> sets of inter-related pages:
  - *Hub pages* are good lists of links on a subject.
    - e.g., “Bob’s list of cancer-related links.”
  - *Authority pages* occur recurrently on good hubs for the subject.

- Best suited for “broad topic” queries rather than for page-finding queries.

- Gets at a broader slice of common *opinion*.

The hope



Best Mobile+Net+TV bundles.

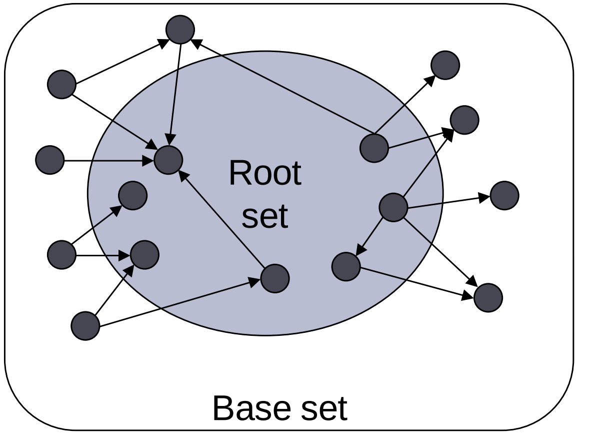
High-level scheme

- Extract from the web a base set of pages that could be good hubs or authorities.

- From these, identify a small set of top hub and authority pages;
  - iterative algorithm.

Base set and root set

- Given text query (say *browser*), use a text index to get all pages containing *browser*.
  - Call this the root set of pages.
- Add in any page that either
  - points to a page in the root set, or
  - is pointed to by a page in the root set.
- Call this the base set.



Distilling hubs and authorities

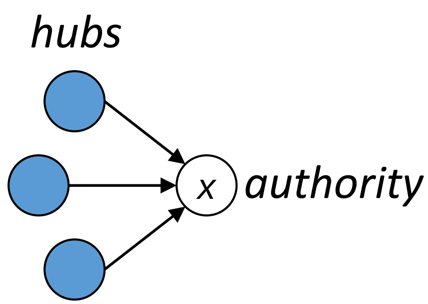
- Compute, for each page x in the base set, a hub score h(x) and an authority score a(x).
- Initialize: for all x, h(x)$\leftarrow$1; a(x) $\leftarrow$1;
- Iteratively update all h(x), a(x);

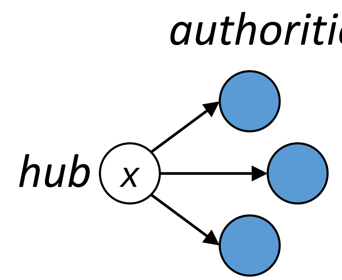
Key

- After iterations
  - output pages with highest h() scores as top hubs
  - highest a() scores as top authorities.

Iterative update

- Repeat the following updates, for all *x*:





$$a(x) \leftarrow \sum_{y \mapsto x} h(y)$$

$$h(x) \leftarrow \sum_{x \mapsto y} a(y)$$

How many iterations?

- Claim: relative values of scores will converge after a few iterations:
  - in fact, suitably scaled, h() and a() scores settle into a steady state!

- We only require the <u>relative orders</u> of the h() and a() scores - not their absolute values.

- In practice, ~5 iterations get you close to stability.

PageRank in Social Media

- Can be applied to rank people, entities, news sources, etc.

- The Who to Follow services is a:
  - personalized version of PageRank
  - combines PageRank with recommender systems

WTF: The Who to Follow Service at Twitter

Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, Reza Zadeh
Twitter, Inc.
@pankaj @ashishgoel @lintool @aneeshs @dongwang218 @reza_zadeh

ABSTRACT

WTF ("Who to Follow") is Twitter's user recommendation service, which is responsible for creating millions of connections daily between users based on shared interests, common connections, and other related factors. This paper provides an architectural overview and shares lessons we learned in building and running the service over the past few years. Particularly noteworthy was our design decision to process the entire Twitter graph in memory *on a single server*, which significantly reduced architectural complexity and allowed us to develop and deploy the service in only a few months. At the core of our architecture is Cassovary, an open-source in-memory graph processing engine we built from scratch for WTF. Besides powering Twitter's user recommendations, Cassovary is also used for search, discovery, promoted products, and other services as well. We describe and evaluate a few graph recommendation algorithms implemented in Cassovary, including a novel approach based on a combination of random walks and SALSA. Looking into the future, we revisit the design of our architecture and comment on its limitations, which are presently being addressed in a second-generation system under development.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database applications—*Data mining*

General Terms: Algorithms, Design

Keywords: graph processing, link prediction, Hadoop

1. INTRODUCTION

The lifeblood of a vibrant and successful social media service is an active and engaged user base. Therefore, maintaining and expanding the active user population is a top priority, and for Twitter, this is no exception. At the core, Twitter is an platform remarkably simple in concept: a user "follows" other users to subscribe to their 140-character tweets, which may be received on a variety of clients (e.g., the twitter.com website, mobile clients on iPhones and Android devices, etc.). The vibrancy of the service, whether in informing users of relevant breaking news or connecting them to communities of interest, derives from its users—all 200 million of them, collectively posting over 400 million tweets every day (as of early 2013).

One important way to sustain and grow Twitter is to help users, existing and new, discover connections. This is the goal of WTF ("Who to Follow"),[1] the Twitter user recommendation service. In the current interface, the WTF box is prominently featured in the left rail of the web client as well as in many other contexts across multiple platforms. WTF suggests Twitter accounts that a user may be interested in following, based on shared interests, common connections, and a number of other factors. Social networking sites such as Facebook and LinkedIn have comparable offerings as well. We identify two distinct but complementary facets to the problem, which we informally call "interested in" and "similar to". For example, a user interested in sports might follow @espn, but we probably wouldn't consider that user similar to @espn. On the other hand, two users might be considered similar based on their shared interest in, say, basketball, or if they follow many of the same users. Twitter also exposes profile similarity as a product feature, visible when visiting a user's profile page. Throughout this paper, our discussion of user recommendations covers both these aspects. Based on the homophily principle, similar users also make good suggestions. Besides powering user recommendations, WTF is also used for search relevance, discovery, promoted products, and other services as well.

This paper provides an overview of Twitter's WTF service, a project that began in spring 2010 and went into production the same summer.[2] Quite explicitly, our goal here is not to present novel research contributions, but to share the overall design of a system that is responsible for creating millions of connections daily and lessons that we have learned over the past few years.

We view this paper as having the following contributions:

- First, we explain and justify our decision to build the service around the assumption that the entire graph will fit in memory *on a single machine*. This might strike the reader as an odd assumption, especially in the era of "big data", but our approach has worked well and in retrospect we believe the decision was correct in the broader context of being able to launch the product quickly.
- Second, we describe the complete end-to-end architecture of the WTF service. At the core is Cassovary, our open-source in-memory graph processing engine.

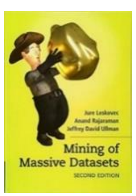
WWW 2013, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2035-1/13/05.

[1] The confusion with the more conventional expansion of the acronym is intentional and the butt of many internal jokes. Also, it has not escaped our attention that the name of the service is actually ungrammatical; the pronoun should properly be in the objective case, as in "whom to follow".

[2] http://blog.twitter.com/2010/07/discovering-who-to-follow.html

Summary

- Web graphs denote a relation of relevance between edges
- Introduced a new way of modeling the value of Web links.
- Key algorithms: PageRank, Topic Specific PageRank, HITS
- References:



Chapter 5 of Jure Leskovec, Anand Rajaraman, Jeff Ullman, “**Mining of Massive Datasets**”, Cambridge University Press, 2011.