ΑΝΑΚΤΗΣΗ ΠΛΗΡΟΦΟΡΙΩΝ ΚΑΙ ΑΝΑΖΗΤΗΣΗ ΣΤΟΝ ΠΑΓΚΟΣΜΙΟ ΙΣΤΟ

Παροράματα από το Πανεπιστήμιο της Στουγκάρδης

Introduction to

Information Retrieval

Hinrich Schütze and Christina Lioma

Lecture 19: Web Search

Overview

1. Recap
2. Big picture
3. Ads
4. Duplicate detection

Outline

1. Recap
2. Big picture
3. Ads
4. Duplicate detection

Indexing anchor text

- Anchor text is often a better description of a page’s content than the page itself.
- Anchor text can be weighted more highly than the text on the page.
- A Google bomb is a search with “bad” results due to maliciously manipulated anchor text.
  - [dangerous cult] on Google, Bing, Yahoo

PageRank

- Model: a web surfer doing a random walk on the web
- Formalization: Markov chain
- PageRank is the long-term visit rate of the random surfer or the steady-state distribution.
- Need teleportation to ensure well-defined PageRank
- Power method to compute PageRank
  - PageRank is the principal left eigenvector of the transition probability matrix.

Computing PageRank: Power method

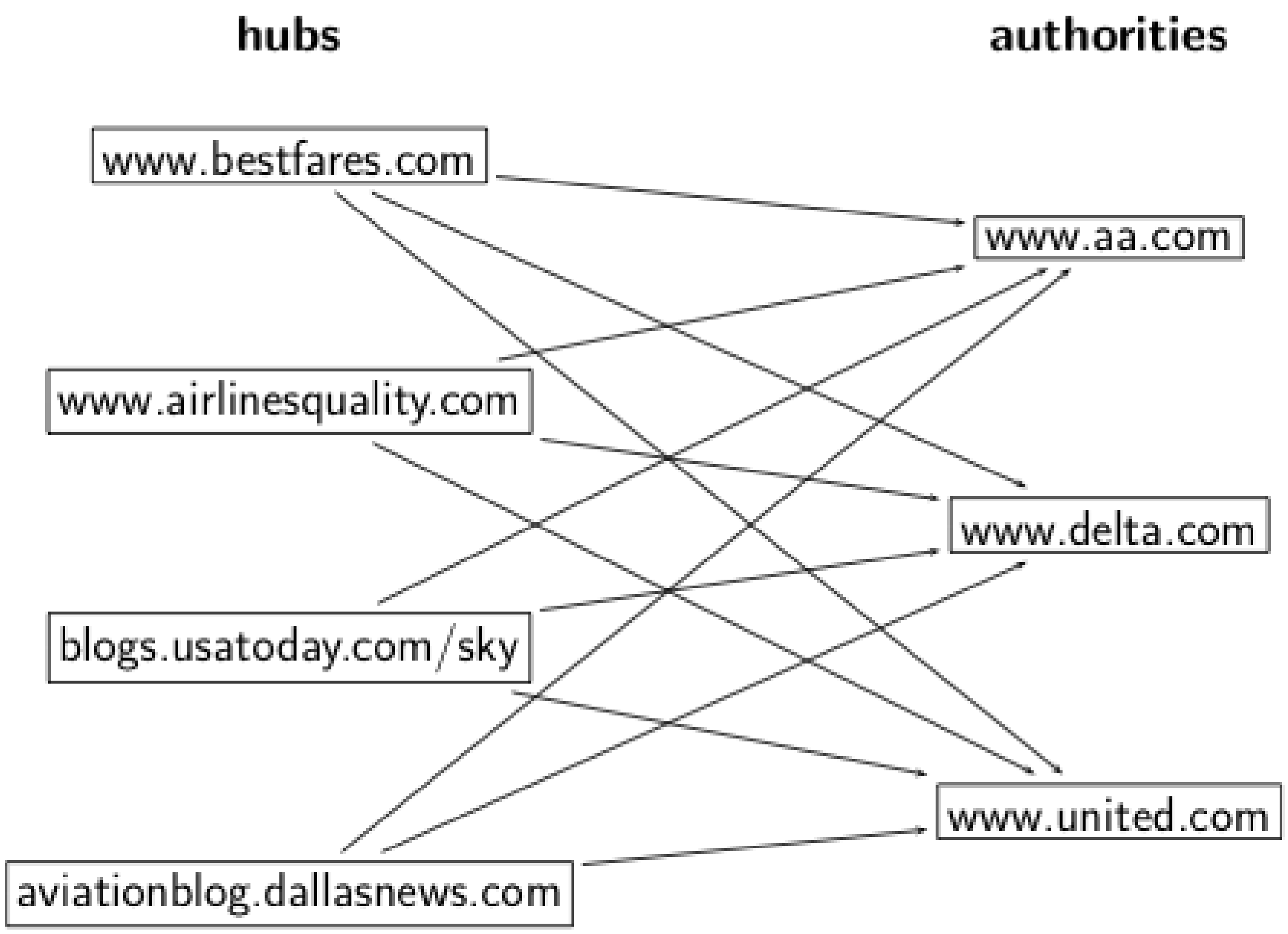
	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2	0.24	0.76	0.252	0.748	$= \vec{x}P^3$
t_3	0.252	0.748	0.2496	0.7504	$= \vec{x}P^4$
				...	
t_∞	0.25	0.75	0.25	0.75	$= \vec{x}P^\infty$

PageRank vector = $\vec{\pi} = (\pi_1, \pi_2) = (0.25, 0.75)$

$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$

$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$

HITS: Hubs and authorities



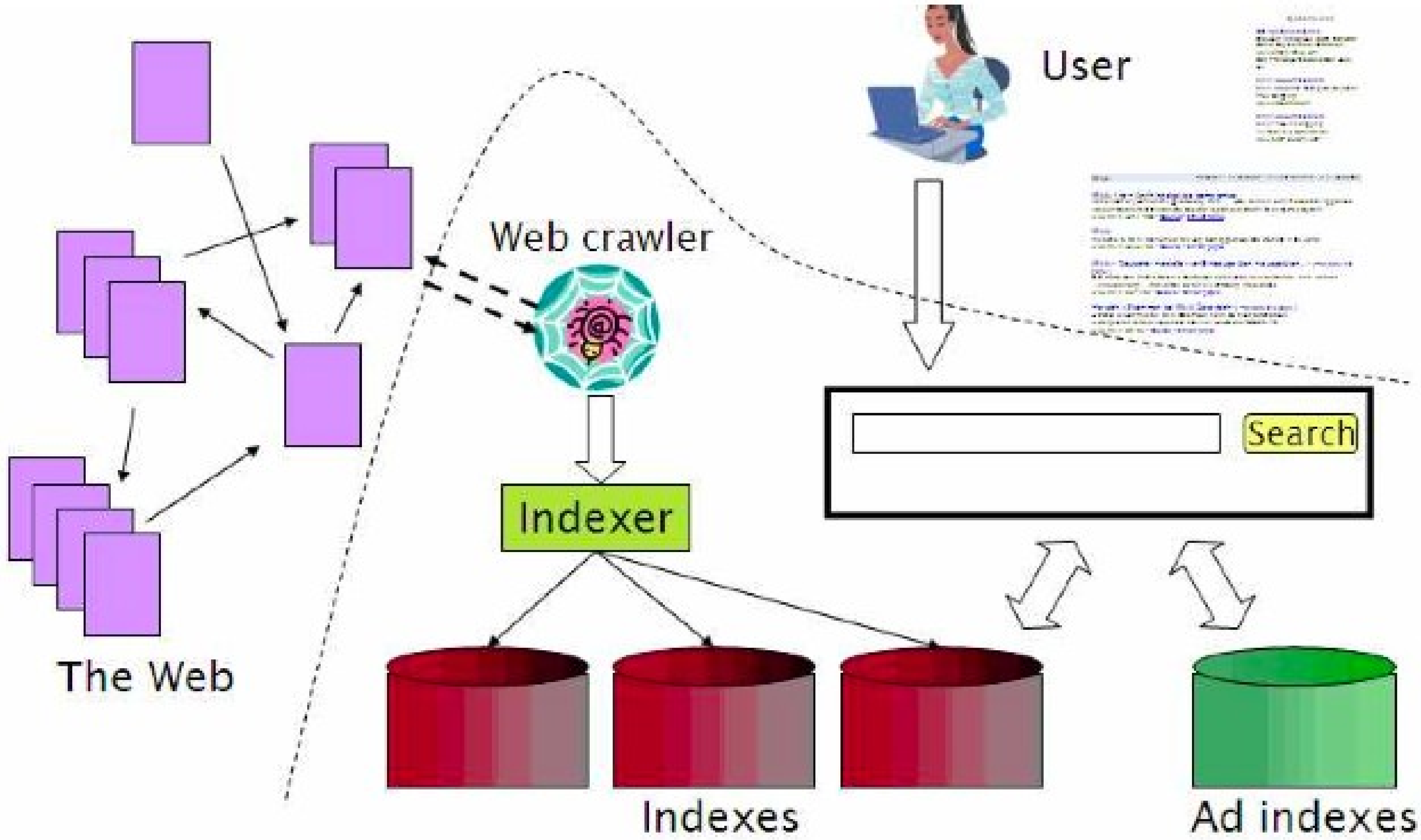
HITS update rules

- A: link matrix
- $\vec{h}$: vector of hub scores
- $\vec{a}$: vector of authority scores
- HITS algorithm:
  - Compute $\vec{h} = A\vec{a}$
  - Compute $\vec{a} = A^T\vec{h}$
  - Iterate until convergence
  - Output (i) list of hubs ranked according to hub score and (ii) list of authorities ranked according to authority score

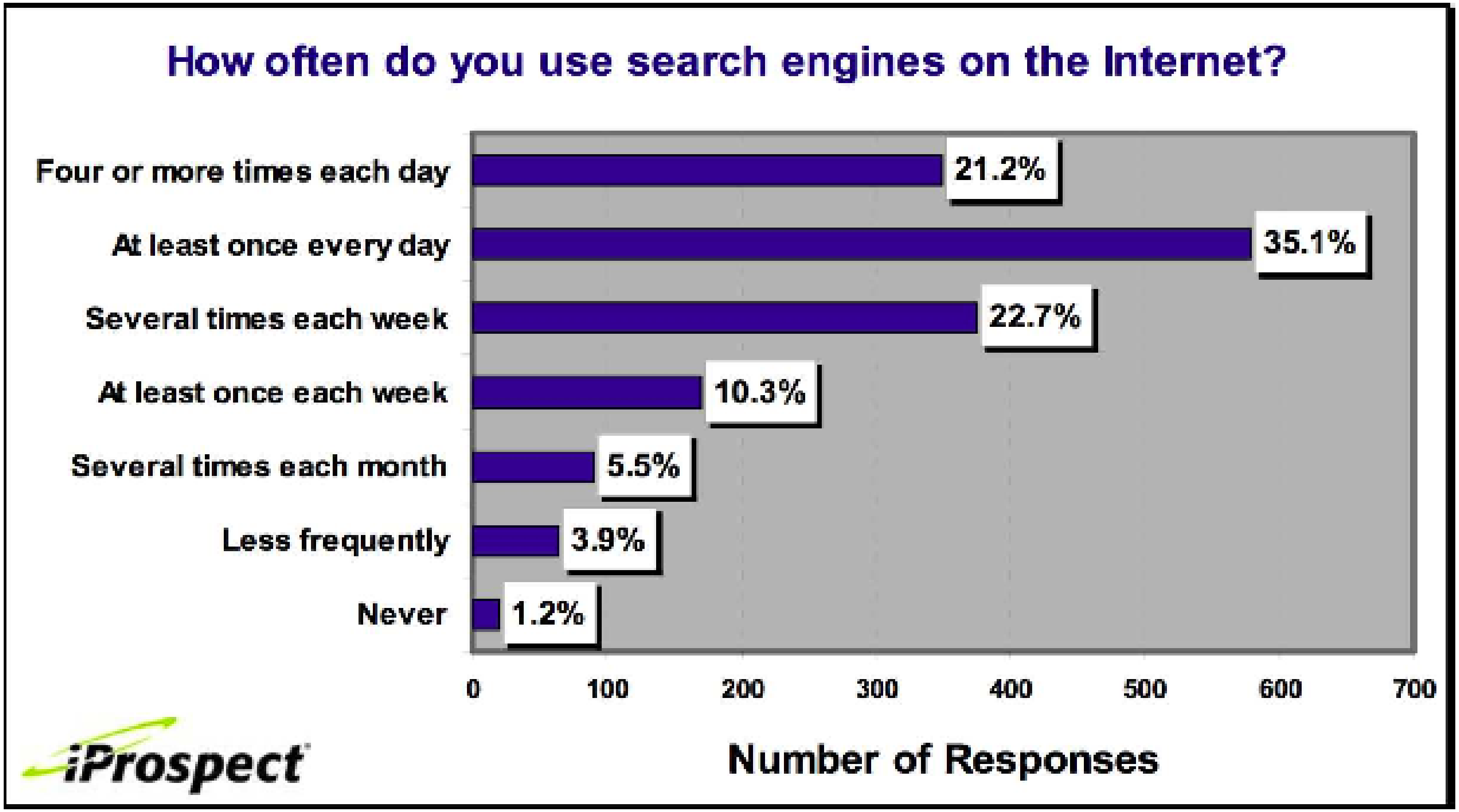
Outline

1. Recap
2. Big picture
3. Ads
4. Duplicate detection

Web search overview



Search is the top activity on the web



Without search engines, the web wouldn’t work

- Without search, content is hard to find.
- → Without search, there is no incentive to create content.
  - Why publish something if nobody will read it?
  - Why publish something if I don’t get ad revenue from it?
- Somebody needs to pay for the web.
  - Servers, web infrastructure, content creation
  - A large part today is paid by search ads.
  - Search pays for the web.

Interest aggregation

- Unique feature of the web: A small number of geographically dispersed people with similar interests can find each other.
  - Elementary school kids with hemophilia
  - People interested in translating R5R5 Scheme into relatively portable C (open source project)
  - Search engines are a key enabler for interest aggregation.

IR on the web vs. IR in general

- On the web, search is not just a nice feature.
  - Search is a key enabler of the web: . . .
  - . . . financing, content creation, interest aggregation etc.

→ look at search ads

- The web is a chaotic und uncoordinated collection. → lots of duplicates – need to detect duplicates
- No control / restrictions on who can author content → lots of spam – need to detect spam
- The web is very large. → need to know how big it is

Take-away today

- Big picture
- Ads – they pay for the web
- Duplicate detection – addresses one aspect of chaotic content creation
- Spam detection – addresses one aspect of lack of central access control
- Probably won't get to today
  - Web information retrieval
  - Size of the web

Outline

1. Recap
2. Big picture
3. Ads
4. Duplicate detection

First generation of search ads: Goto (1996)

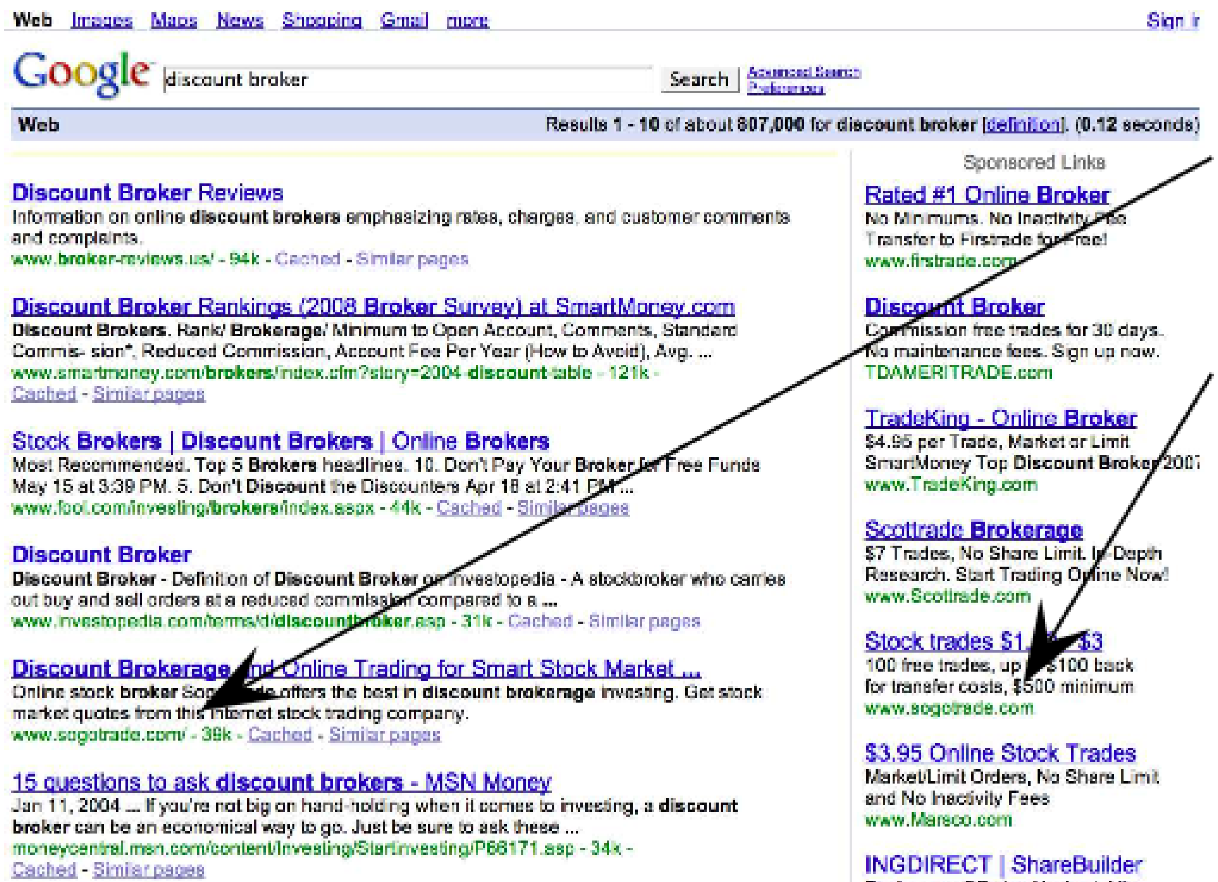
First generation of search ads: Goto (1996)

- Buddy Blake bid the maximum ($0.38) for this search.
- He paid $0.38 to Goto every time somebody clicked on the link.
- Pages were simply ranked according to bid – revenue maximization for Goto.
- No separation of ads/docs. Only one result list!
- Upfront and honest. No relevance ranking, . . .
- . . . but Goto did not pretend there was any.

Second generation of search ads: Google (2000/2001)

- Strict separation of search results and search ads

Two ranked lists: web pages (left) and ads (right)



SogoTrade appears in search results.

SogoTrade appears in ads.

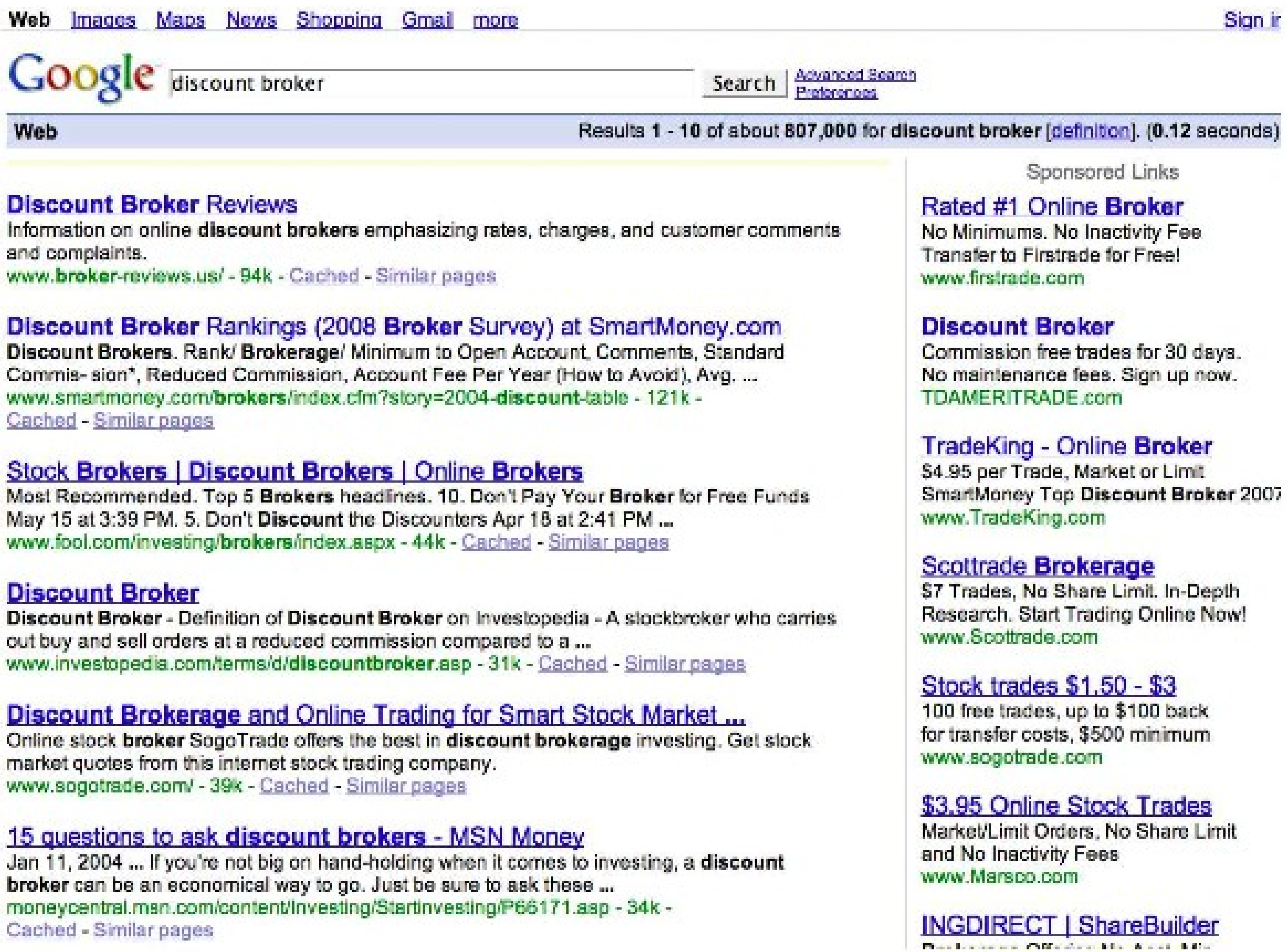
Do search engines rank advertisers higher than non-advertisers?

All major search engines claim no.

Do ads influence editorial content?

- Similar problem at newspapers / TV channels
- A newspaper is reluctant to publish harsh criticism of its major advertisers.
- The line often gets blurred at newspapers / on TV.
- No known case of this happening with search engines yet?

How are the ads on the right ranked?



How are ads ranked?

- Advertisers bid for keywords – sale by auction.
- Open system: Anybody can participate and bid on keywords.
- Advertisers are only charged when somebody clicks on your ad.
- How does the auction determine an ad's rank and the price paid for the ad?
- Basis is a second price auction, but with twists
- For the bottom line, this is perhaps the most important research area for search engines – computational advertising.
  - Squeezing an additional fraction of a cent from each ad means billions of additional revenue for the search engine.

How are ads ranked?

- First cut: according to bid price `a la Goto
  - Bad idea: open to abuse
  - Example: query [does my husband cheat?] → ad for divorce lawyer
  - We don't want to show nonrelevant ads.
- Instead: rank based on bid price and relevance
- Key measure of ad relevance: clickthrough rate
  - clickthrough rate = CTR = clicks per impressions
- Result: A nonrelevant ad will be ranked low.
  - Even if this decreases search engine revenue short-term
  - Hope: Overall acceptance of the system and overall revenue is maximized if users get useful information.
- Other ranking factors: location, time of day, quality and loading speed of landing page
- The main ranking factor: the query

Google AdsWords demo

Google's second price auction

advertiser	bid	CTR	ad rank	rank	paid
A	\$4.00	0.01	0.04	4	(minimum)
B	\$3.00	0.03	0.09	2	\$2.68
C	\$2.00	0.06	0.12	1	\$1.51
D	\$1.00	0.08	0.08	3	\$0.51

- bid: maximum bid for a click by advertiser
- CTR: click-through rate: when an ad is displayed, what percentage of time do users click on it? CTR is a measure of relevance.
- ad rank: bid $\times$ CTR: this trades off (i) how much money the advertiser is willing to pay against (ii) how relevant the ad is
- rank: rank in auction
- paid: second price auction price paid by advertiser

Google's second price auction

advertiser	bid	CTR	ad rank	rank	paid
A	\$4.00	0.01	0.04	4	(minimum)
B	\$3.00	0.03	0.09	2	\$2.68
C	\$2.00	0.06	0.12	1	\$1.51
D	\$1.00	0.08	0.08	3	\$0.51

Second price auction: The advertiser pays the minimum amount necessary to maintain their position in the auction (plus 1 cent).

$\text{price}_1 \times \text{CTR}_1 = \text{bid}_2 \times \text{CTR}_2$ (this will result in $\text{rank}_1 = \text{rank}_2$)

$\text{price}_1 = \text{bid}_2 \times \text{CTR}_2 / \text{CTR}_1$

$p_1 = \text{bid}_2 \times \text{CTR}_2/\text{CTR}_1 = 3.00 \times 0.03/0.06 = 1.50$
$p_2 = \text{bid}_3 \times \text{CTR}_3/\text{CTR}_2 = 1.00 \times 0.08/0.03 = 2.67$
$p_3 = \text{bid}_4 \times \text{CTR}_4/\text{CTR}_3 = 4.00 \times 0.01/0.08 = 0.50$

Keywords with high bids

According to http://www.cwire.org/highest-paying-search-terms/

\$69.1	mesothelioma treatment options
\$65.9	personal injury lawyer michigan
\$62.6	student loans consolidation
\$61.4	car accident attorney los angeles
\$59.4	online car insurance quotes
\$59.4	arizona dui lawyer
\$46.4	asbestos cancer
\$40.1	home equity line of credit
\$39.8	life insurance quotes
\$39.2	refinancing
\$38.7	equity line of credit
\$38.0	lasik eye surgery new york city
\$37.0	2nd mortgage
\$35.9	free car insurance quote

Search ads: A win-win-win?

- The search engine company gets revenue every time somebody clicks on an ad.
- The user only clicks on an ad if they are interested in the ad.
  - Search engines punish misleading and nonrelevant ads.
  - As a result, users are often satisfied with what they find after clicking on an ad.
- The advertiser finds new customers in a cost-effective way.

Exercise

- Why is web search potentially more attractive for advertisers than TV spots, newspaper ads or radio spots?
- The advertiser pays for all this. How can the advertiser be cheated?
- Any way this could be bad for the user?
- Any way this could be bad for the search engine?

Not a win-win-win: Keyword arbitrage

- Buy a keyword on Google
- Then redirect traffic to a third party that is paying much more than you are paying Google.
  - E.g., redirect to a page full of ads
- This rarely makes sense for the user.
- Ad spammers keep inventing new tricks.
- The search engines need time to catch up with them.

Not a win-win-win: Violation of trademarks

- Example: geico
- During part of 2005: The search term "geico" on Google was bought by competitors.
- Geico lost this case in the United States.
- Louis Vuitton lost similar case in Europe.
- See http://google.com/tm complaint.html
- It's potentially misleading to users to trigger an ad off of a trademark if the user can't buy the product on the site.

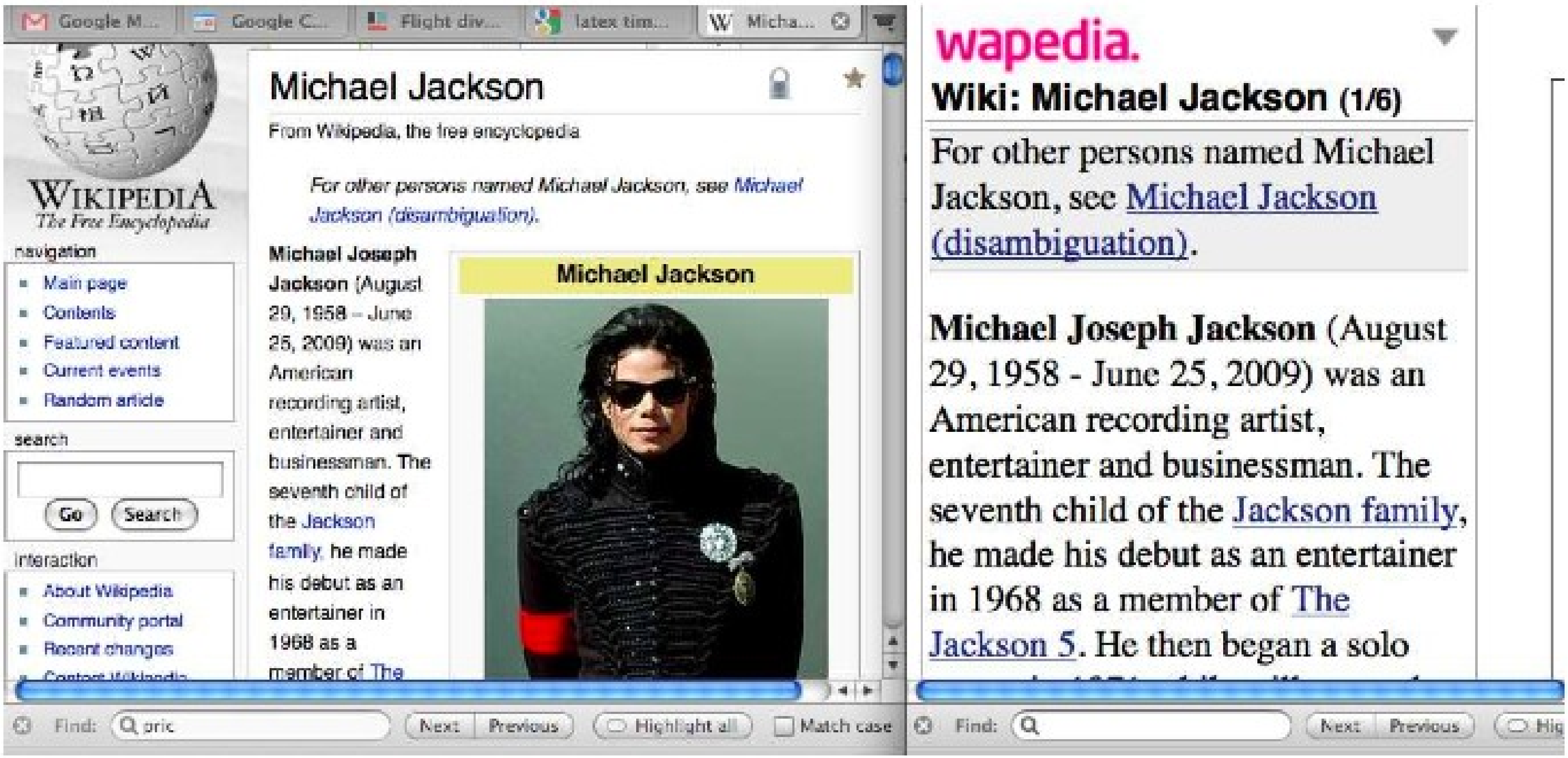
Outline

1. Recap
2. Big picture
3. Ads
4. Duplicate detection

Duplicate detection

- The web is full of duplicated content.
- More so than many other collections
- Exact duplicates
  - Easy to eliminate
  - E.g., use hash/fingerprint
- Near-duplicates
  - Abundant on the web
  - Difficult to eliminate
- For the user, it's annoying to get a search result with near-identical documents.
- Marginal relevance is zero: even a highly relevant document becomes nonrelevant if it appears below a (near-)duplicate.
- We need to eliminate near-duplicates.

Near-duplicates: Example



Exercise

How would you eliminate near-duplicates on the web?

Detecting near-duplicates

- Compute similarity with an edit-distance measure
- We want "syntactic" (as opposed to semantic) similarity.
  - True semantic similarity (similarity in content) is too difficult to compute.
- We do not consider documents near-duplicates if they have the same content, but express it with different words.
- Use similarity threshold θ to make the call "is/isn't a near-duplicate".
- E.g., two documents are near-duplicates if similarity

  $> \theta = 80\%$.

Represent each document as set of **shingles**

- A shingle is simply a word n-gram.
- Shingles are used as features to measure syntactic similarity of documents.
- For example, for $n = 3$, "a rose is a rose is a rose" would be represented as this set of shingles:
  - { a-rose-is, rose-is-a, is-a-rose }
- We can map shingles to $1..2^m$ (e.g., $m = 64$) by fingerprinting.
- From now on: s_k refers to the shingle's fingerprint in $1..2^m$.
- We define the similarity of two documents as the Jaccard coefficient of their shingle sets.

Recall: Jaccard coefficient

- A commonly used measure of overlap of two sets
- Let *A* and *B* be two sets
- Jaccard coefficient:

$$\text{JACCARD}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

$(A \neq \emptyset \text{ or } B \neq \emptyset)$

- JACCARD(A,A) = 1
- JACCARD(A,B) = 0 if $A \cap B = 0$
- *A* and *B* don't have to be the same size.
- Always assigns a number between 0 and 1.

Jaccard coefficient: Example

- Three documents:

  d_1: "Jack London traveled to Oakland"

  d_2: "Jack London traveled to the city of Oakland"

  d_3: "Jack traveled from Oakland to London"
- Based on shingles of size 2 (2-grams or bigrams), what are the Jaccard coefficients $J(d_1, d_2)$ and $J(d_1, d_3)$?
- $J(d_1, d_2) = 3/8 = 0.375$
- $J(d_1, d_3) = 0$
- Note: very sensitive to dissimilarity

Represent each document as a sketch

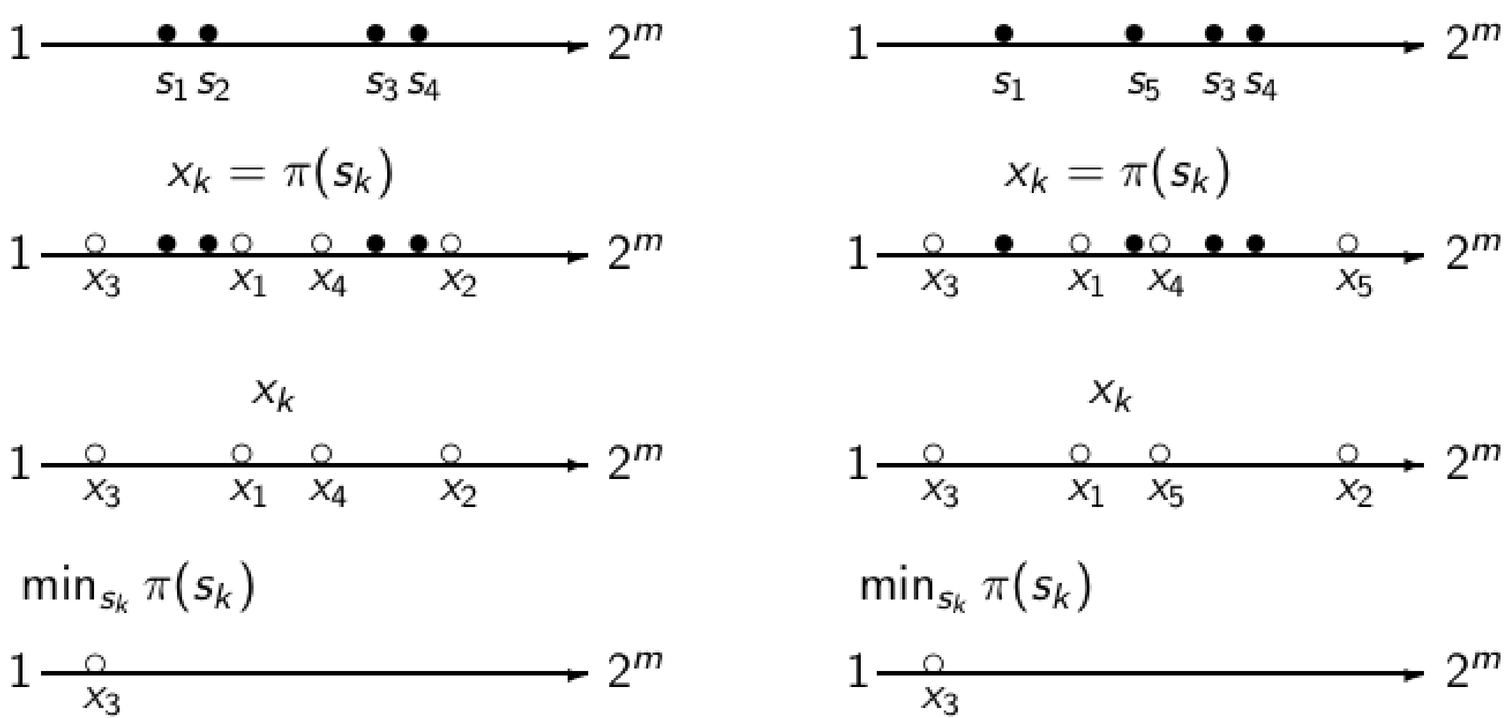
- The number of shingles per document is large.
- To increase efficiency, we will use a sketch, a cleverly chosen subset of the shingles of a document.
- The size of a sketch is, say, $n = 200$ . . .
- . . . and is defined by a set of permutations $\pi_1 \ldots \pi_{200}$.
- Each π_i is a random permutation on $1..2^m$
- The sketch of d is defined as:

  $< \min_{s \in d} \pi_1(s), \min_{s \in d} \pi_2(s), \ldots, \min_{s \in d} \pi_{200}(s) >$

  (a vector of 200 numbers).

The Permutation and minimum: Example

document 1: $\{s_k\}$ document 2: $\{s_k\}$



We use $\min_{s\in d1} \pi(s)$ = mins∈d_2 $\pi(s)$ as a test for: are d_1 and d_2 near-duplicates? In this case: permutation π says: $d_1 \approx d_2$

Computing Jaccard for sketches

- Sketches: Each document is now a vector of $n = 200$ numbers.
- Much easier to deal with than the very high-dimensional space of shingles
- But how do we compute Jaccard?

Computing Jaccard for sketches (2)

- How do we compute Jaccard?
- Let U be the union of the set of shingles of d_1 and d_2 and I the intersection.
- There are $|U|!$ permutations on U.
- For $s' \in I$, for how many permutations π do we have $\operatorname{argmin}_{s \in d1} \pi(s) = s' = \operatorname{argmin}_{s \in d2} \pi(s)$?
- Answer: $(|U| - 1)!$
- There is a set of $(|U| - 1)!$ different permutations for each s in I. $\Rightarrow$ $|I|(|U| - 1)!$ permutations make $\operatorname{argmin}_{s \in d1} \pi(s) = \operatorname{argmin}_{s \in d2} \pi(s)$ true
- Thus, the proportion of permutations that make $\min_{s \in d1} \pi(s) = \min_{s \in d2} \pi(s)$ true is:

$$\frac{|I|(|U| - 1)!}{|U|!} = \frac{|I|}{|U|} = J(d_1, d_2)$$

Estimating Jaccard

- Thus, the proportion of successful permutations is the Jaccard coefficient.
  - Permutation π is successful iff $\min_{s \in d1} \pi(s) = \min_{s \in d2} \pi(s)$
- Picking a permutation at random and outputting 1 (successful) or 0 (unsuccessful) is a Bernoulli trial.
- Estimator of probability of success: proportion of successes in n Bernoulli trials. ($n = 200$)
- Our sketch is based on a random selection of permutations.
- Thus, to compute Jaccard, count the number k of successful permutations for $< d_1, d_2 >$ and divide by $n = 200$.
- $k/n = k/200$ estimates $J(d_1, d_2)$.

Implementation

- We use hash functions as an efficient type of permutation: $h_i : \{1..2^m\} \rightarrow \{1..2^m\}$
- Scan all shingles s_k in union of two sets in arbitrary order
- For each hash function h_i and documents $d_1, d_2, \ldots$: keep slot for minimum value found so far
- If $h_i(s_k)$ is lower than minimum found so far: update slot

Example

	d_1	d_2
s_1	1	0
s_2	0	1
s_3	1	1
s_4	1	0
s_5	0	1

$h(x) = x \bmod 5$

$g(x) = (2x + 1) \bmod 5$

$\min(h(d_1)) = 1 \neq 0 =$
$\min(h(d_2)) \quad \min(g(d_1)) =$

$2 \neq 0 = \min(g(d_2))$

$\hat{J}(d_1, d_2) = \frac{0+0}{2} = 0$

	d_1 slot		d_2 slot	
h		∞		∞
g		∞		∞
$h(1) = 1$	1	1	–	∞
$g(1) = 3$	3	3	–	∞
$h(2) = 2$	–	1	2	2
$g(2) = 0$	–	3	0	0
$h(3) = 3$	3	1	3	2
$g(3) = 2$	2	2	2	0
$h(4) = 4$	4	1	–	2
$g(4) = 4$	4	2	–	0
$h(5) = 0$	–	1	0	0
$g(5) = 1$	–	2	1	0

final sketches

Exercise

	d_1	d_2	d_3
s_1	0	1	1
s_2	1	0	1
s_3	0	1	0
s_4	1	0	0

$h(x) = 5x + 5 \bmod 4$

$g(x) = (3x + 1) \bmod 4$

Estimate $\hat{J}(d_1, d_2)$, $\hat{J}(d_1, d_3)$, $\hat{J}(d_2, d_3)$

Solution (1)

	d_1	d_2	d_3
s_1	0	1	1
s_2	1	0	1
s_3	0	1	0
s_4	1	0	0

$h(x) = 5x + 5 \bmod 4$

$g(x) = (3x + 1) \bmod 4$

	d_1 slot		d_2 slot		d_3 slot	
		∞		∞		∞
		∞		∞		∞
$h(1) = 2$	–	∞	2	2	2	2
$g(1) = 0$	–	∞	0	0	0	0
$h(2) = 3$	3	3	–	2	3	2
$g(2) = 3$	3	3	–	0	3	0
$h(3) = 0$	–	3	0	0	–	2
$g(3) = 2$	–	3	2	0	–	0
$h(4) = 1$	1	1	–	0	–	2
$g(4) = 1$	1	1	–	0	–	0

final sketches

Solution (2)

$$\hat{J}(d_1, d_2) = \frac{0+0}{2} = 0$$

$$\hat{J}(d_1, d_3) = \frac{0+0}{2} = 0$$

$$\hat{J}(d_2, d_3) = \frac{0+1}{2} = 1/2$$

Shingling: Summary

- Input: N documents
- Choose n-gram size for shingling, e.g., $n = 5$
- Pick 200 random permutations, represented as hash functions
- Compute N sketches: 200 $\times$ N matrix shown on previous slide, one row per permutation, one column per document
- Compute $\frac{N \cdot (N-1)}{2}$ pairwise similarities
- Transitive closure of documents with similarity > θ
- Index only one document from each equivalence class

Efficient near-duplicate detection

- Now we have an extremely efficient method for estimating a Jaccard coefficient for a single pair of two documents.
- But we still have to estimate $O(N^2)$ coefficients where N is the number of web pages.
- Still intractable
- One solution: locality sensitive hashing (LSH)
- Another solution: sorting (Henzinger 2006)