Introduction to Informatics

Lecture 16: Web Basics

Brief (non-technica ry

- Early keyword-based engines ca. 1995-1997
  - Altavista, Excite, Infoseek, Inktomi, Lycos

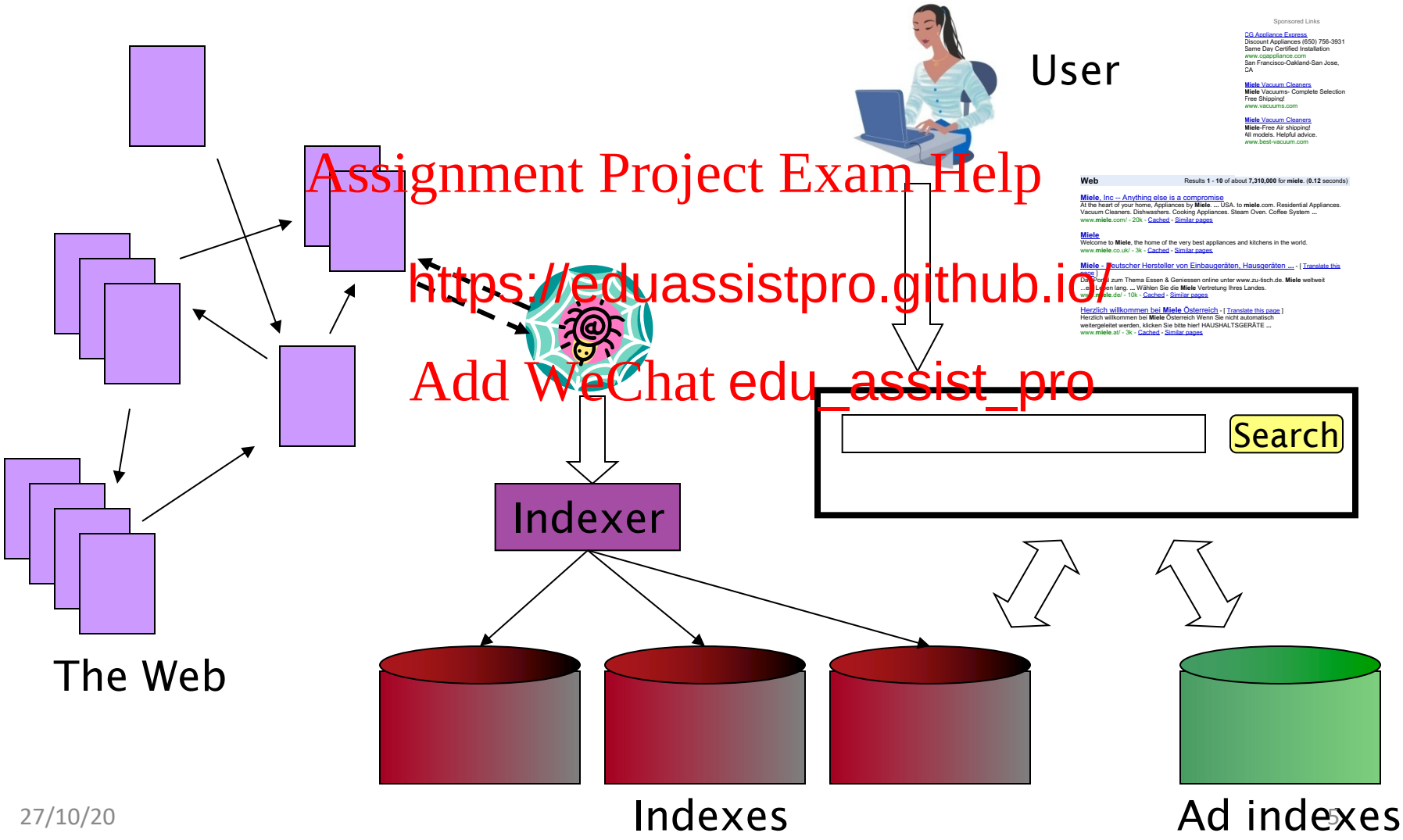
- Paid search ed into Overture.co

  - Your search ranking depended on ho paid
  - Auction for keywords: ***casino*** was expensive!

Brief (non-technica ry

- 1998+: Link-based ranking pioneered by Google
  - Blew away all early engines save Inktomi
  - Great user experience in search of a business model
  - Meanwhile Got were nearing $1 billion
- Result: Google a the side, independent of search results
  - Yahoo followed suit, acquiring Ov placement) and Inktomi (for search)
- 2005+: Google gains search share, dominating in Europe and very strong in North America
  - 2009: Yahoo! and Microsoft propose combined paid search offering

Web search basics



User Needs

- Need [Brod02, RL04]
  - **Informational** – want to learn about something (~40% / 65%)
  - **Navigational** – want to go to that page (~25% / 15%)
  - **Transactional** – ted) (~35% / 20%)
    - Access a service
    - Downloads
    - Shop
  - **Gray areas**
    - Find a good hub
    - Exploratory search "see what's there"

Low hemoglobin

Airlines

Seatt

M images

Canon S410

Car rental Brasil

How far do people look for results?

(Source: iprospect.com WhitePaper_2006_SearchEngineUserBehavior.pdf)

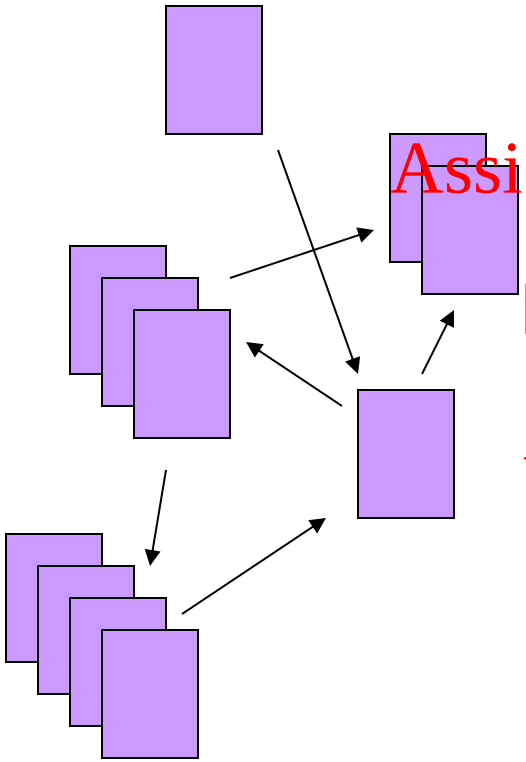
Users' empirical eval f results

- Quality of pages varies widely
  - Relevance is not enough
  - Other desirable qualities (non IR!!)
    - Content: Trustworthy, diverse, non-duplicated, well maintained
    - Web readability: display correctly & fast
    - No annoya
- Precision vs. r
  - On the web, recall seldom matte
- What matters
  - Precision at 1? Precision above the fold?
  - Comprehensiveness – must be able to deal with obscure queries
    - Recall matters when the number of matches is very small
- User perceptions may be unscientific, but are significant over a large aggregate

Users' empirical eval f engines

- Relevance and validity of results
- UI – Simple, no clutter, error tolerant
- Trust – Results are objective
- Coverage of top
- Pre/Post proces
  - Mitigate user errors (auto spell ch ist,…)
  - Explicit: Search within results, mor ne-…
  - Anticipative: related searches
- Deal with idiosyncrasies
  - Web specific vocabulary
    - Impact on stemming, spell-check, etc
  - Web addresses typed in the search box
- "The first, the last, the best and the worst …"

The Web document collection



The Web

- No design/co-ordination
- Distributed content creation, linking, democratization of publishing
- Content includes truth, lies, obsolete dictions … html, …), semi-structure tated photos), structure)…
- Scale much larger than previous text collections … but corporate records are catching up
- Growth – slowed down from initial "volume doubling every few months" but still expanding
- Content can be *dynamically generated*

Spam

- (Search Engine Optimization)

Size of the web

What is the size of the web ?

- Issues
  - The web is really infinite
    - Dynamic content, e.g., calendar
    - Soft 404: w valid page
  - Static web c on, mostly due to mirroring (~3
  - Some servers are seldom co
- Who cares?
  - Media, and consequently the user
  - Engine design
  - Engine crawl policy. Impact on recall.

What can we attempt to measure?

- The relative sizes of search engines
  - The notion of a page being indexed is still *reasonably* well defined.
  - Already there
    - Document extension: e.g. engine not yet crawled, by indexing anchortext.
    - Document restriction: All engines restrict what is indexed (first *n* words, only relevant words, etc.)
- The coverage of a search engine relative to another particular crawling process.

New definition?

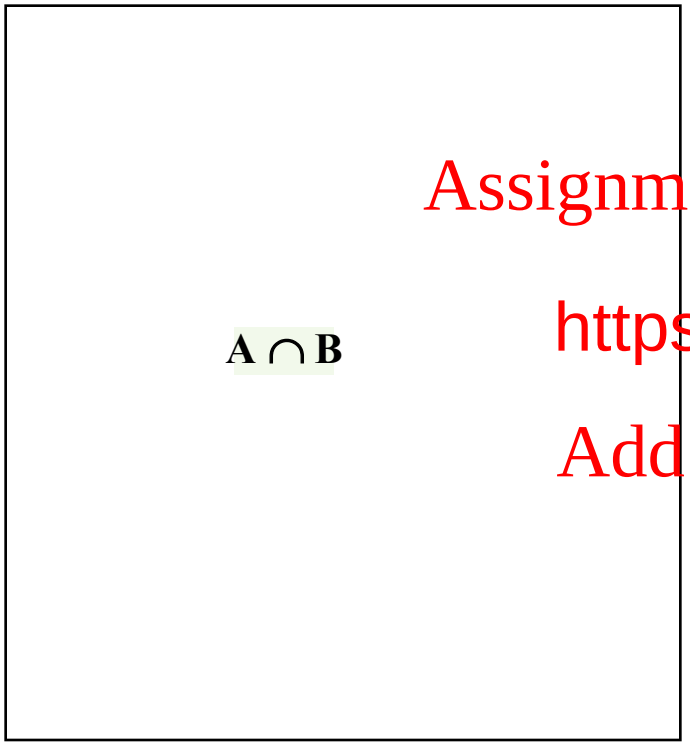
(IQ is whatever the IQ tests measure.)

- The statically indexable web is whatever search engines in

- Different engi ferences
  - max url depth, max count/h m rules, priority rules, etc.
- Different engines index different things under the same URL:
  - frames, meta-keywords, document restrictions, document extensions, ...

Relative Size from O

Given two engines



Sample URLs randomly from A

Check if contained in B and vice versa

$A \cap B$ = $\frac{1}{2}$ * Size A

$A \cap B$ = $\frac{1}{6}$ * Size B

(1/2)*Size A = (1/6)*Size B

∴ Size A / Size B =

(1/6)/(1/2) = 1/3

Each test involves: (i) Sampling (ii) Checking

Sampling URLs

- Ideal strategy: Generate a random URL and check for containment in each index.
- Problem: Ran find! Enough to generate a ran a given Engine.
- Approach 1: Generate a ra contained in a given engine
  - Suffices for the estimation of relative size
- Approach 2: Random walks / IP addresses
  - In theory: might give us a true estimate of the size of the web (as opposed to just relative sizes of indexes)

Statistical methods

- Approach 1
  - Random queries
  - Random searches
- Approach 2
  - Random IP addresses
  - Random walks

Random URLs from r queries

- Generate random query: how?
  - **Lexicon:** 400,000+ words from a web crawl
  - **Conjunctive Queries:** w and w
    e.g., vocalists A
- Get 100 result URLs from engin
- Choose a random URL as the c check for presence in engine B
- This distribution induces a probability weight W(p) for each page.
- Conjecture: $W(SE_A) / W(SE_B) \sim |SE_A| / |SE_B|$

Not an English dictionary

Query Based Che

- *Strong Query* to check whether an engine *B* has a document *D*:
  - Download *D*. Get list of words.
  - Use 8 low freq y to *B*
  - Check if *D* is pr
- Problems:
  - Near duplicates
  - Frames
  - Redirects
  - Engine time-outs
  - Is 8-word query good enough?

Advantages & d tages

- Statistically sound under the induced weight.
- Biases induced by random query
  - Query Bias: Favors content-rich pages in the language(s) of the lexicon
  - Ranking Bias: *So* s & fetch all
  - Checking Bias: Duplicates, impover mitted
  - Document or query restriction ight not deal properly with 8 words conjunctive query
  - Malicious Bias: Sabotage by engine
  - Operational Problems: Time-outs, failures, engine inconsistencies, index modification.

Random searches

- Choose random searches extracted from a local log [Lawrence & Giles 97] or build "random searches" [Notess]
  - Use only que
  - Count norma
  - Use ratio statistics

Advantages & disa es

- Advantage
  - Might be a better reflection of the human perception of coverage
- Issues
  - Samples are correlated wit log
  - Duplicates
  - Technical statistical problems (must have non-zero results, ratio average not statistically sound)

Random searches

- 575 & 1050 queries from the NEC RI employee logs
- 6 Engines in 1998, 11 in 1999
- Implementation:
  - Restricted lts in total
  - Counted U r verifying query match
  - Computed size ratio & ove vidual queries
  - Estimated index size ratio & overlap by averaging over all queries

Queries from Lawrence study

- *adaptive access control*
- *neighborhood preservation topographic*
- *hamiltonian structures*
- *right linear gramm*
- *pulse width modula*
- *unbalanced prior probabilities*
- *ranked assignment method*
- *internet explorer favourites importing*
- *karvel thornber*
- *zili liu*
- *softmax activation function*
- *bose multidimensional system theory*
- *gamma mlp*
- *dis*
- *exploring neural*
- *marking*
- *counterpropagation network*
- *fat shattering dimension*
- *abelson amorphous computing*

Random IP address

- Generate random IP addresses
- Find a web server at the given address
  - If there's one
- Collect all pag
  - From this, choose a page at r

Random IP addresse

- HTTP requests to random IP addresses
  - Ignored: empty or authorization required or excluded
  - [Lawr99] Estimated 2.8 million IP addresses running crawlable we l) from observing 2500 servers
  - OCLC using IP sampling foun s in 2001
    - Netcraft [Netc02] accessed 37.2 million hosts in July 2002
- [Lawr99] exhaustively crawled 2500 servers and extrapolated
  - Estimated size of the web to be 800 million pages
  - Estimated use of metadata descriptors:
    - Meta tags (keywords, description) in 34% of home pages, Dublin core metadata in 0.3%

Advantages & disa es

- Advantages
  - Clean statistics
  - Independent of crawling strategies
- Disadvantages
  - Doesn't deal w
  - Many hosts mi ccept requests
  - No guarantee all pages are lin age.
    - Eg: employee pages
  - Power law for # pages/hosts generates bias towards sites with few pages.
    - But bias can be accurately quantified IF underlying distribution understood
  - Potentially influenced by spamming (multiple IP's for same server to avoid IP block)

Random walks

- View the Web as a directed graph
- Build a random walk on this graph
  - Includes various "jump" rules back to visited sites
    - Does not get s
    - Can follow all l
  - Converges to a stationary distri
    - Must assume graph is finite and i f the walk.
    - Conditions are not satisfied (cookie crumbs, flooding)
    - Time to convergence not really known
  - Sample from stationary distribution of walk
  - Use the "strong query" method to check coverage by SE

Advantages & disa es

- Advantages
  - "Statistically clean" method at least in theory!
  - Could work even for infinite web (assuming convergence) under certai
- Disadvantage
  - List of seeds is a problem.
  - Practical approximation might not be valid.
  - Non-uniform distribution
    - Subject to link spamming

Conclusions

- No sampling solution is perfect.
- Lots of new ideas ...
- ....but the pro r
- Quantitative and a good research problem

Duplicate detectio

Duplicate docume

- The web is full of duplicated content
- Strict duplicate detection = exact match
  - Not as com

- But many, m plicates
  - E.g., Last modified date t fference between two copies of a page

Duplicate/Near-Duplic ction

- *Duplication*: Exact match can be detected with fingerprints
- *Near-Duplication*: Approximate match

  - Overview

    - Compute syntactic similari dit-distance measure

    - Use similarity threshold to detect near-duplicates
      - E.g., Similarity > 80% => Documents are "near duplicates"
      - Not transitive though sometimes used transitively

Computing Similari

- Features:
  - Segments of a document (natural or artificial breakpoints)
  - Shingles (Word N-Grams)
  - ***a rose is a ros***

    a_rose_is_a

    rose_is_a_rose

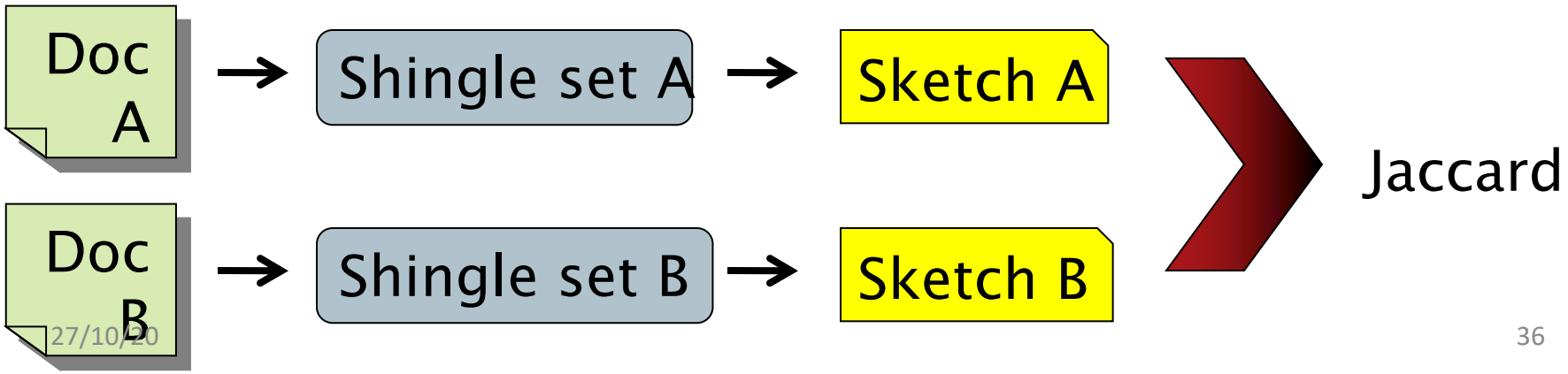
    is_a_rose_is

    a_rose_is_a

- Similarity Measure between two docs (= sets of shingles)
  - Set intersection
  - Specifically (Size_of_Intersection / Size_of_Union)

Shingles + Set Inter

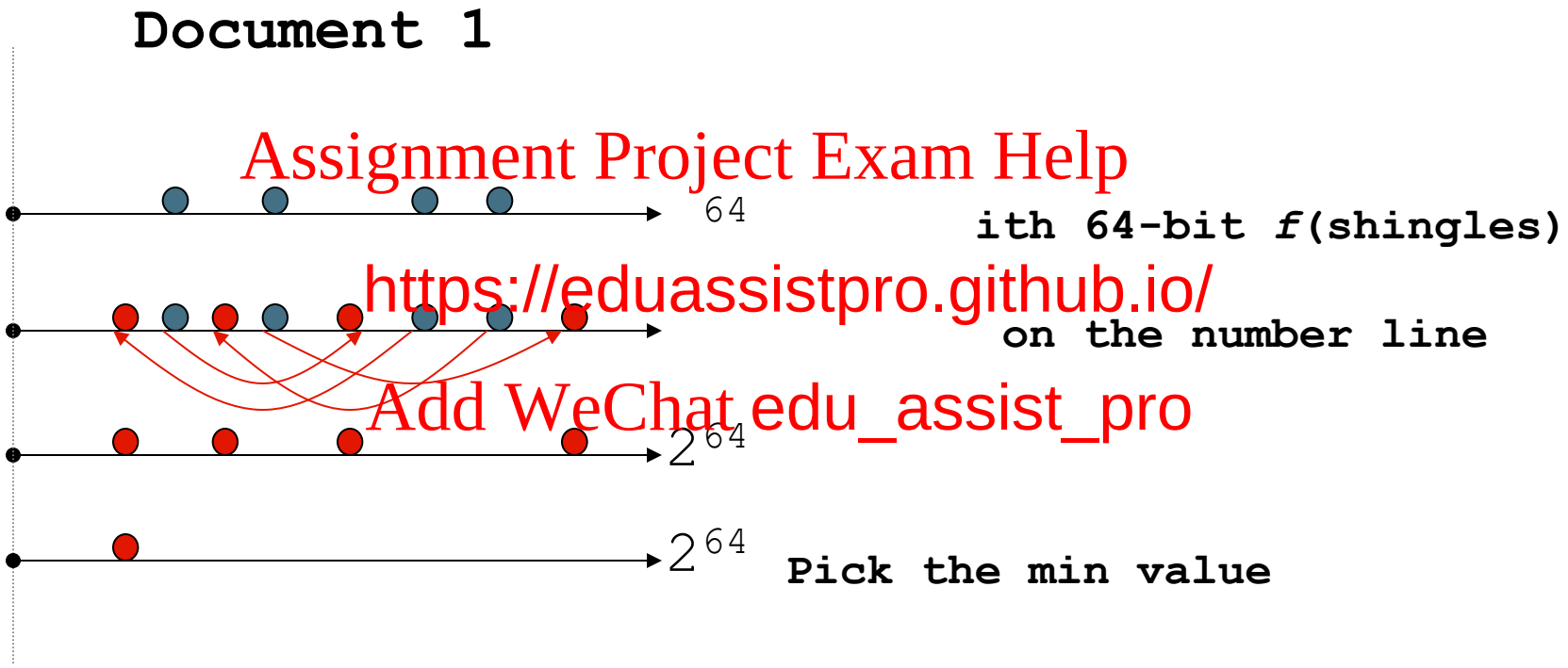
- Computing exact set intersection of shingles between all pairs of documents is expensive/intractable
  - Approximate using a set of shingles from each (a *sketch*)
- Estimate (size_of_intersection / size_of_union) based on a short sketch



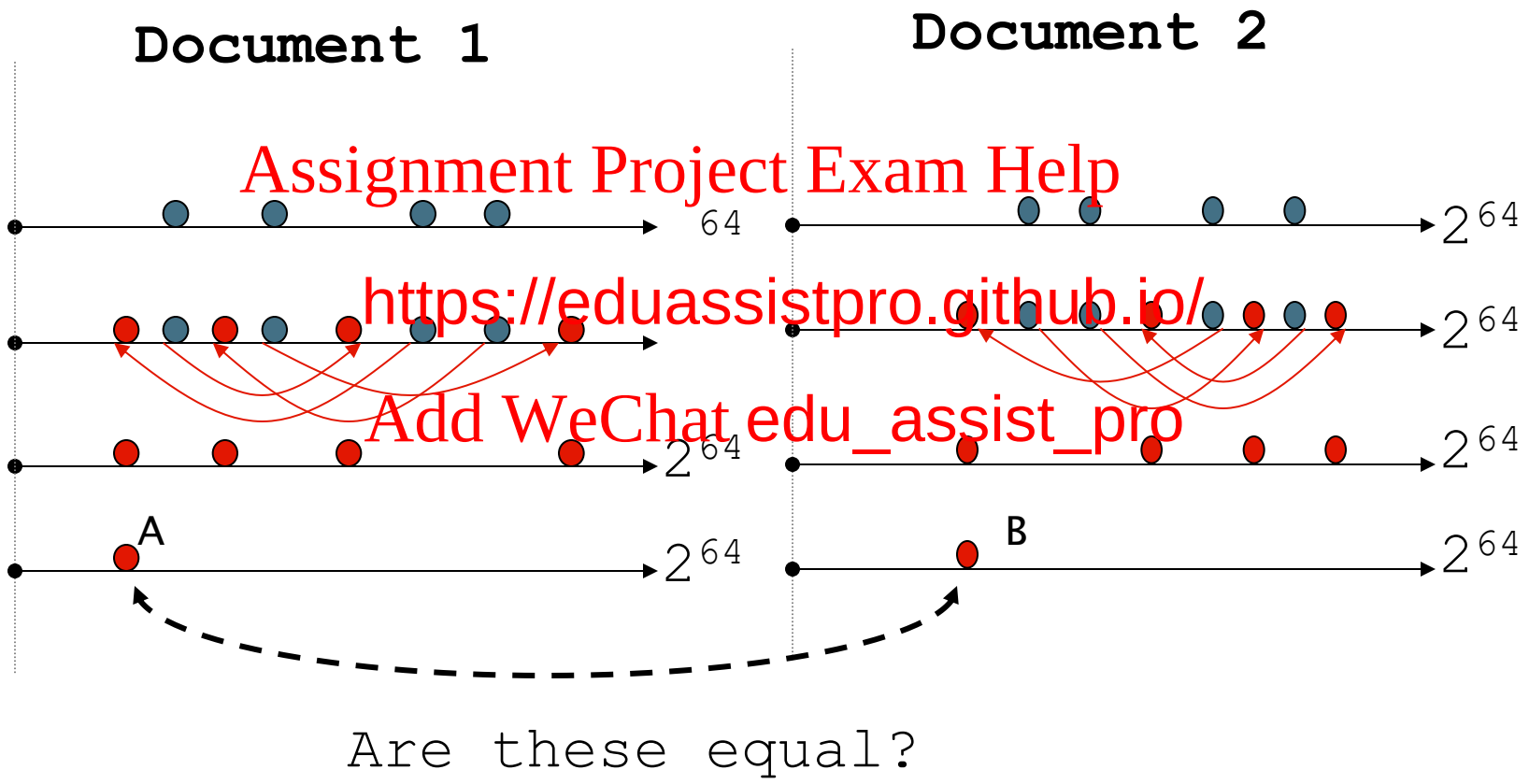
Sketch of a docum

- Create a "sketch vector" (of size ~200) for each document
  - Documents that share ≥ t (say 80%) corresponding vector elem
  - For doc D, sketch$_D$[i] is :
    - Let f map all shingles in t o [0, 2^m-1] (e.g., f = fingerprinting)
    - Let π_i be a *random permutation* on [0, 2^m-1]
    - Pick MIN {π_i(f(s))} over all shingles s in D

Computing Sketch[1]

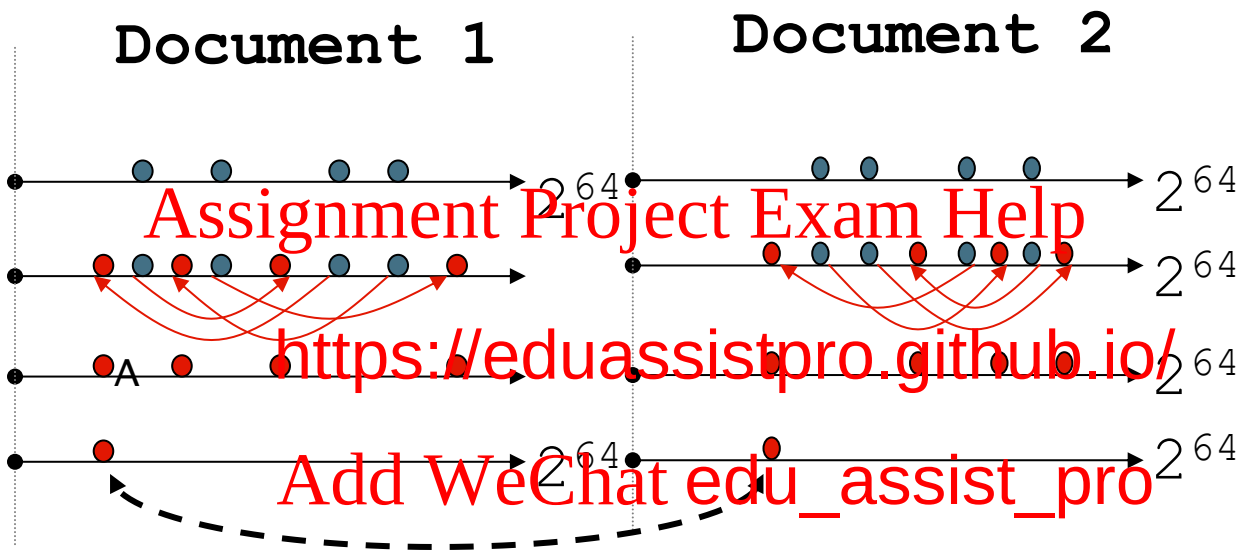


Test if Doc1.Sketch[i] = Doc2.Sketch[i]



Test for 200 random permutations: $\pi_1, \pi_2, \ldots \pi_{200}$

However…



A = B iff the shingle with the MIN value in the union of Doc1 and Doc2 is common to both (i.e., lies in the intersection)

Claim: This happens with probability

Why?

Size_of_intersection / Size_of_union

Set Similarity of se

$$\text{Jaccard}(C_i, C_j) = \frac{|C_i \cap C_j|}{|C_i \cup C_j|}$$

- View sets as c e row for each element in the tes presence of item i in set j

- Example

C_1	C_2
0	1
1	0
1	1
0	0
1	1
0	1

Jaccard(C_1, C_2) = 2/5 = 0.4

Key Observation

- For columns C_i, C_j, four types of rows

	C_i	C_j
type A	1	1
type B	1	0
type C	0	1
type D	0	0

- Overload notation: A = # of rows of type A
- **Claim**

$$\text{Jaccard}(C_i, C_j) = \frac{A}{A + B + C}$$

"Min" Hashing

- Randomly permute rows
- Hash $h(C_i)$ = index of first row with 1 in column C_i

- Surprising Pro

$$\Pr\left[h(C_i) = h(C_j) \right] = ?$$

- Why?
  - Both are A/(A+B+C)
  - Look down columns C_i, C_j until first non-Type-D row
  - $h(C_i) = h(C_j)$ ←→ type A row

Min-Hash sketches

- Pick *P* random row permutations
- MinHash sketch

sketch(C) = list of *k* indexes of first rows with 1 in column C

- Similarity of signatures

  - Let sim[sketch(C_i),sketch(C_j)] = fraction of permutations where MinHash values agree
  - Observe **E**[sim(sig(C_i),sig(C_j))] = Jaccard(C_i,C_j)

Practical Implementation

- Random permutation is hard to obtain; simulate them using universal hashing instead
  - h: {0, 1, 2, …, U} → {0, 1, 2, …, M}
  - h(x) = ((a*x +
  - where
    - P >> U and is a prime number
    - a, b are two randomly chosen integers modulo P and a != 0
  - sketch(C) = { $\text{argmin}_{e \in C} \{ h_i(e) \} \mid 1 \le i \le k\}$

Example

	C_1	C_2	C_3
R_1	1	0	1
R_2	0	1	1
R_3	1	0	0
R_4	1	0	1
R_5	0	1	0

Signatures

	S_1	S_2	S_3
Perm 1 = (12345)	1	2	1
Perm 2 = (54321)	4	5	4
2)	3	5	4

Similarities

	1-2	1-3	2-3
Col-Col	0.00	0.50	0.25
Sig-Sig	0.00	0.67	0.00

Example Using the Sal Hashing

$h(x) = (7x+1 \bmod 31) \bmod 9$
$g(x) = (17x+8 \bmod 31) \bmod 9$

	C_1	C_2	C_3
R_1	1	0	1
R_2	0	1	1
R_3	1	0	0
R_4	1	0	1
R_5	0	1	0

$S_1 = \{R_1, R_3, R_4\}$
$h(e) = \{8, 4, 2\}$ ➔ min_elem = R_4
$g(e) = \{7, 1, 5\}$ ➔ min_elem = R_3
sketch(S1) = $\{R_4, R_3\}$

$S_2 = \{R_2, R_3, R_5\}$
h(e → min_elem = R_5
g(e → min_elem = R_5
sketch(S1) = $\{R_5, R_5\}$

Note: this example results in different sketches from the previous slide

Therefore, estimated similarity between S_1 and S_2 is 0/2 = 0.0

All signature pairs

- Now we have an extremely efficient method for estimating a Jaccard coefficient for a single pair of documents.

- But we still ha ficients where *N* is the number

  - Still slow
- One solution: locality sensitive hashing (LSH)
- Another solution: Sorting (Henzinger 2006)

SimHash

- Generalization of LSH to other similarity measures
  [Charikar, STOC 02]
  - $\theta(\mathbf{x}, \mathbf{y})$: related to cosine
  - $h_{\mathbf{u}}(\mathbf{x}) = sig$ om unit vector
  - then $\Pr[h_{\mathbf{u}}$ $/\pi$

Practical Implementation

- Near duplicate Web page detection from google [Henzinger, SIGIR06] [Manku et al, WWW07]
  - Document D → set of tokens with idf weighting → form a set of "features" V(D)
  - Each feature is hashed to a 64-dimensional binary vector of [-1,1]
  - Sum up the weighted projection vectors: resulting v(D) → **r**(D)
  - a *f*-bit signature sig(D) ← sign(**r**(D))
- Results (in comparison with Shingling)
  - Fairly accurate and stable
  - Does not capture order among tokens

Simhash

Simhash Example

2 - 2 + 1
- 1 - 1 +
1 - 1 + 1
+ 1 - 1 +
1 -1 + 1
= 1

count-min sketch, approximately preserves the inner product of the raw feature vectors

More resources

- IIR Chapter 19