Algorithms for duplicate documents

Andrei Broder
IBM Research
abroder@us.ibm.com

Fingerprinting (discussed last week)

- Fingerprints are short tags for larger objects.
- Notations

  $\Omega =$ The set of all objects

  $k =$ The lenght of the fingerprint

  $f : \Omega \rightarrow \{0,1\}^k$ A fingerprinting function
- Properties

$$f(A) \neq f(B) \Rightarrow A \neq B$$

$$\Pr\left(f(A) = f(B) \middle| A \neq B\right) \approx \frac{1}{2^k}$$

Fingerprinting schemes

- Fingerprints vs hashing
  - For hashing I want good distribution so bins will be equally filled
  - For fingerprints I don't want any collisions = much longer hashes but the distribution does not matter!
- Cryptographically secure:
  - MD2, MD4, MD5, SHS, etc
  - relatively slow
- Rabin's scheme
  - Based on polynomial arithmetic
  - Very fast (1 table lookup + 1 xor + 1 shift) /byte
  - Nice extra-properties

Rabin's scheme

[Rabin '81], [B '93]

- View each string A as a polynomial over $\mathbf{Z_2}$:

  $$A = 1\ 0\ 0\ 1\ 1 \quad \Rightarrow \quad A(x) = x^4 + x + 1$$

- Let $P(t)$ be an irreducible polynomial of degree k chosen uar
- The fingerprint of A is

  $$f(A) = A(t) \bmod P(t)$$

- The probability of collision among n strings of average length t is about

  $$n\text{^}2\ t\ /\ 2\text{^}k$$

Nice extra properties

- Let ♦ = catenation. Then

$$f(a \blacklozenge b) = f(f(a) \blacklozenge b)$$

- Can compute extensions of strings easily.
- Can compute fprs of sliding windows.

1995 – AltaVista was born at Digital SRC

- First large scale web search engine
  - “Complete web” then = 30 million documents!!!
  - Current estimate = 11.5 B docs [Gullio & Signorini 05]
- First web annoyance: duplication of documents was immediately visible

Background on web indexing

- Web search engines (Google, MSN, Yahoo, etc…)
  - Crawler – starts from a set of seed URLs, fetches pages, parses, and repeats.
  - Indexer -- builds the index.
  - Search interface -- talks to users.
- AltaVista (Nov 2001)
  - Explored ~ 2-3 B URL -> global ranking
  - Processed ~ 1B pages -> filtering
  - Indexed fully ~ 650 M pages > 5 TB of text

Reasons for duplicate filtering

- Proliferation of almost but not quite equal documents on the Web:
  - Legitimate: Mirrors, local copies, updates, etc.
  - Malicious: Spammers, spider traps, dynamic URLs, “cookie crumbs”
  - Mistaken: Spider errors
- Costs:
  - RAM and disks
  - Unhappy users
- Approximately 30% of the pages on the web are (near) duplicates. [B,Glassman,Manasse & Zweig ‘97, Shivakumar & Garcia-Molina ’98]
- In enterprise search even larger amount of duplication.

Cookie crumbs

- Some sites create some session and/or user id that becomes part of the URL = “cookie crumb”
- Real cookies are stored in user space and persistent across sessions.
- Crawler comes many times to the same page with a different cookie crumb
- Page is slightly modified between different visits.
- Example
  - http://www.crutchfield.com/S-fXyiE5bZS43/
  - http://www.crutchfield.com/S-LcNLKgc7bMg/

Cookie crumbs

Observations

- Must filter both duplicate and near-duplicate documents
- Computing pair-wise edit distance would take forever
- Natural approach = sampling substrings (letters, words, sentences, etc.)

  … but sampling twice even from the same document will not produce identical samples. (Birthday paradox in reverse – need sqrt(n) samples before a collision)

Desiderata

- Store only small sketches for each document.
- On-line processing. (Once sketch is done, source is unavailable)
- Good mathematics. (Small biases might have large impact.)
- At most $n \log n$ time for n documents.
- Ridiculous rule of thumb: At web size you can not do anything that is not linear in n except sorting

The basics of our solution

[B ‘97], [B, Glassman, Manasse, & Zweig ‘97], [B ‘00]

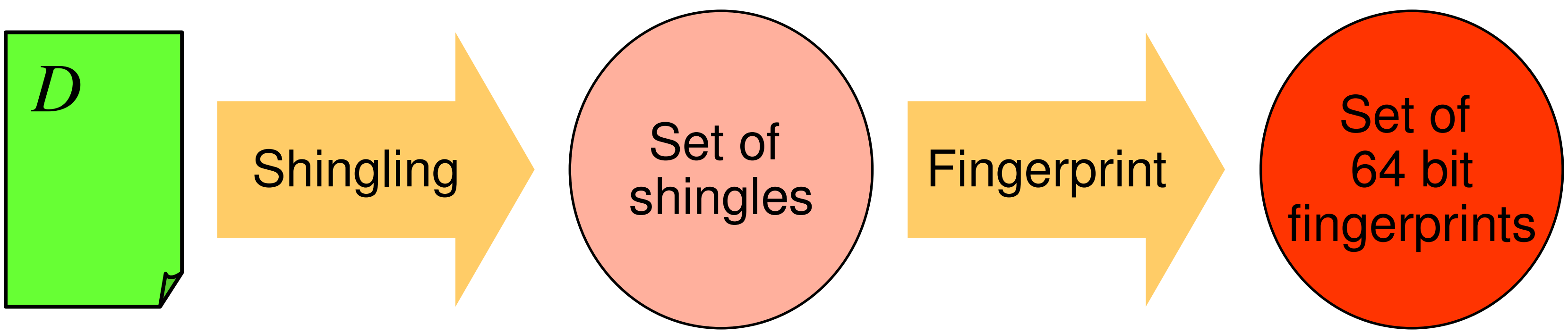
1. Reduce the problem to a set intersection problem

2. Estimate intersections by sampling minima

Shingling

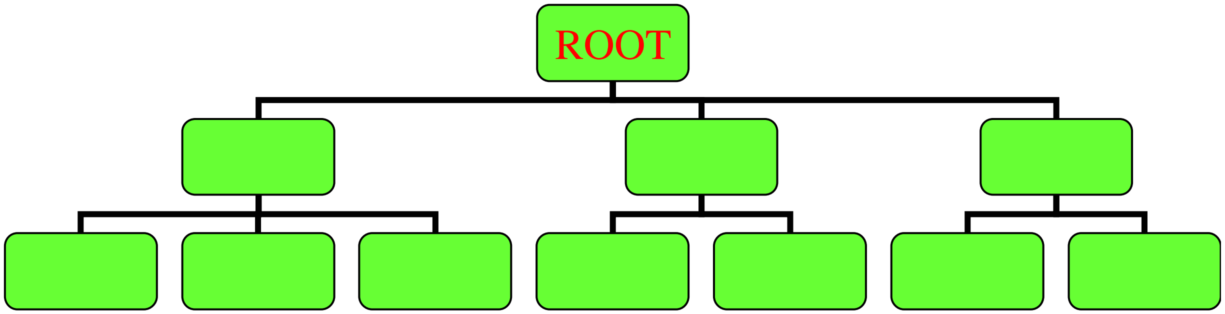
- Shingle = Fixed size sequence of w contiguous words (q-gram)

```
a rose is a rose is a rose
a rose is a
  rose is a rose
       is a rose is
          a rose is a
            rose is a rose
```



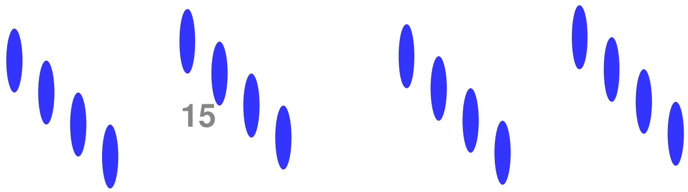
Trees, rain, & shingles (joke!)

CS Tree

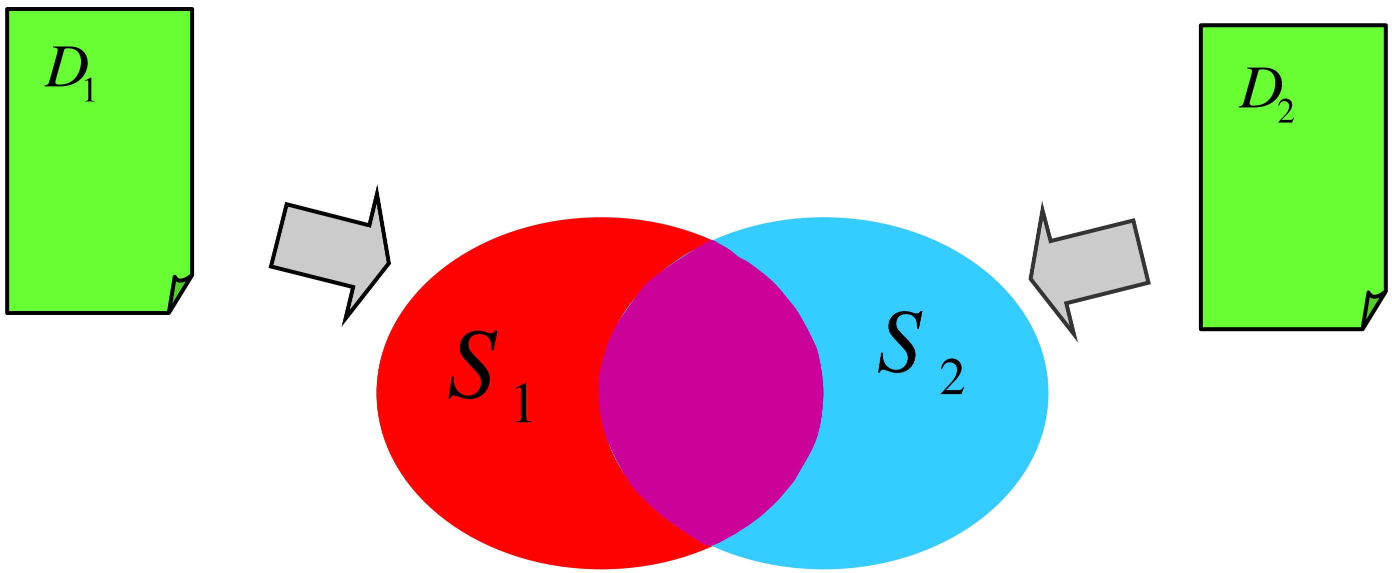


CS Shingles

CS Rain



Defining resemblance



$$resemblance = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

a.k.a. Jaccard distance

Impact of shingle size

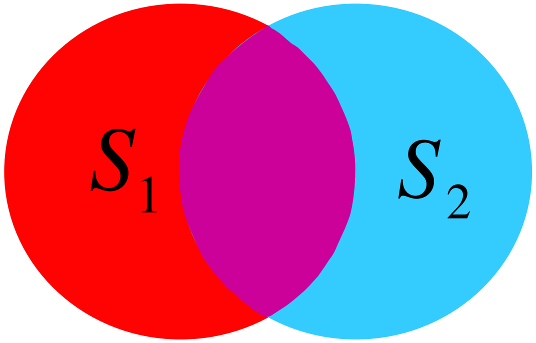
- Long shingles $\Rightarrow$ small random changes have large impact.
- Short shingles $\Rightarrow$ unrelated documents can have too much commonality.
- Good sizes: 3 --10
- See also results about q-gram distance vs. edit distance [Ukkonen '91]
- See also discussion in Schleimer & al., "Winnowing: Local Algorithms for Document Fingerprinting" SIGMOD 2003

Sampling minima

- Apply a random permutation σ to the set $[0..2^{64}]$
- Crucial fact

Let $\alpha = \sigma^{-1}(\min(\sigma(S_1))) \quad \beta = \sigma^{-1}(\min(\sigma(S_2)))$

$$\Pr(\alpha = \beta) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$



- More generally, we look at the k smallest elements in $S_1 \cup S_2$ and check how many are in common.

Observations

- Min Hash = example of locally sensitive hash [Indyk & Motwani '99] (week 5)
  - Hashing such that two items are more likely to collide if they are close under certain metric.
- 1 – Res(A,B) obeys the triangle inequality
  - Can be proven directly (painful …)
  - Follows from general properties of LSH [Charikar '02]

Can it be done differently?

Any family of functions $\{f\}$ such that $f(S) \in S$

that satisfies

$$\Pr(f(S_1) = f(S_2)) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

is such that every f is defined by

$$f(S) = \pi_f^{-1}(\min(\pi_f(S)))\}$$

[B & Mitzenmacher 99]

Implementation

- Choose a random permutations of $\pi(U)$.
- For each document keep a sketch $S(D)$ consisting of t minimal elements of $\pi(D)$.
- Estimate resemblance of A and B by counting common minimal elements within the first t elements of $\pi(A\ U\ B)$.
- Details in [B ‘97]

Alternative implementation

- Choose a random permutations of $\pi(U)$.
- For each document keep a sketch $S(D)$ consisting of all elements of $\pi(D)$ that are $0 \bmod m$.
- Estimate resemblance of A and B by counting common elements.
- Disadvantage: proportional to the length of original document.

Clustering the Web

[B, Glassman, Manasse, & Zweig ‘97]

- We took the 30 million documents found by AltaVista in April 1996
- We found all clusters of similar documents.

Cluster formation

Doc 1 Doc 2 Doc 3 … Doc N

(shingle-ID) (shingle-ID) (shingle-ID) (shingle-ID)
... ...
(shingle-ID) (shingle-ID) (shingle-ID) (shingle-ID)

Sketch, sorted on shingle.

(shingle-ID)
...
(shingle-ID)

Merge-sort.

(ID-ID Count) (ID-ID Count) (ID-ID Count) (ID-ID Count) (ID-ID Count)
... ...
(ID-ID Count) (ID-ID Count) (ID-ID Count) (ID-ID Count) (ID-ID Count)

Sort on ID-ID.

(ID-ID Count)
...
(ID-ID Count)

Merge-sort.

Clusters

Union-Find.

Still, not very easy ...

- On a farm of Alphas (in `97)
  - Sketching: 4.6 alpha-days
  - Exact Duplicate Elimination: 0.3
  - Shingle Merging: 1.7
  - ID-ID Pair Formation: 0.7
  - ID-ID Merging: 2.6
- On a large memory MIPS machine
  - Cluster Formation: 0.5 mips-days
- TOTAL: ~10 alpha-days (~ 150KB/sec)

What did we learn in '97?

- Most documents were unique but also there were lots of duplicates.
  - 18 million unique documents (roughly 60%)
- Most clusters were small
  - ~70% of the clusters had 2 documents
- The average cluster was small
  - ~3.4 documents/cluster
- A few clusters were big
  - 3 clusters had between 10000 and 40000 documents
- This distribution of cluster sizes was still roughly correct in 2001 (based on AV data from 2001)

Filtering

- In many cases value of resemblance not needed.
- Check only if the resemblance is above a certain (high) threshold, e.g. 90%
- Might have false positive and false negatives

New approach – Use multiple perms

- [B ‘98]
- Advantages
  - Simpler math $\Rightarrow$ better understanding.
  - Better for filtering
- Disadvantage
  - Time consuming
- Similar approach independently proposed by [Indyk & Motwani ‘99]

Sketch construction

- Choose a set of t random permutations of U
- For each document keep a sketch $S(D)$ consisting of t minima = samples
- Estimate resemblance of A and B by counting common samples
- Need to worry about quality of randomness
- The permutations should be from a min-wise independent family of permutations.

Min-wise independent permutations

- A truly random permutation on 2^{64} elements is undoable.
- Need an easy-to-represent polynomial size family of permutations such that

  For every set X

  every element x in X

  has an equal chance to become the minimum
- See [B, Charikar, Frieze, & Mitzenmacher ‘97].

MWI Issues

- Size of MWI families
- How good are easy-to-implement families? (e.g. linear transformation)

Minimum size of MWI families

- Exact case $P = 1/|X|$
  - exponential UB = LB = lcm(1, 2, …,n)
    - LB [BCFM ‘98], UB [Takei, Itoh, & Shinozaki]
    - See also [Norin ‘02]
- Approximate case $P = (1 \pm \varepsilon)/|X|$
  - polynomial (non-constructive)
  - $O(n^{1/\varepsilon})$ [Indyk ‘98, Saks & al. ‘99]
- “Application”: Derandomization of the Rajagopalan-Vazirani approximate parallel set cover [B, Charikar, & Mitzenmacher ‘98]

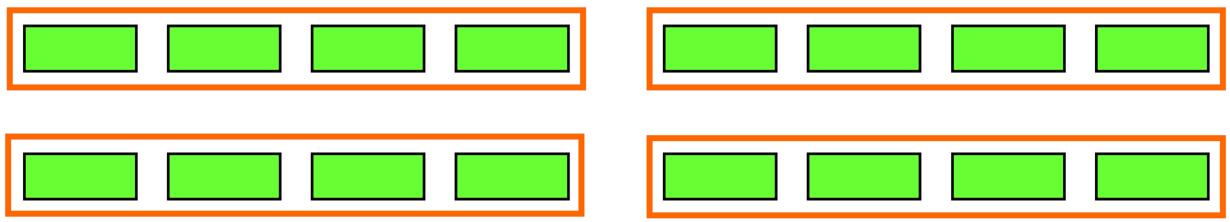
Quality of MWI families

- Linear transformation are not good in the worst case but work reasonable well in practice.
  - See [BCFM ‘97], [Bohman, Cooper, & Frieze ’00]
- Matrix transformations
  - [B & Feige ‘00]
- Some code available from http://www.icsi.berkeley.edu/~zhao/minwise/ [Zhao ’05]

The filtering mechanism

Sketch 1:

Sketch 2:



- Divide into k groups of s elements. (t = k * s)
- Fingerprint each group => feature
- Two documents are fungible if they have more than r common features.

Real implementation

- $\rho = 90\%$. In a 1000 word page with shingle length = 8 this corresponds to
  - Delete a paragraph of about 50-60 words.
  - Change 5-6 random words.
- Sketch size $t = 84$, divided into $k = 6$ groups of $s = 14$ samples
- 8 bytes fingerprints $\rightarrow$ store $6 \times 8 = 48$ bytes/document
- Threshold $r = 2$
- Variant: 200 samples, divided into 8 groups of 25. Threshold $r = 1$.

Probability that two documents are deemed fungible

Two documents with resemblance ρ

- Using the full sketch

$$P = \sum_{i=t}^{k \cdot s} \binom{k \cdot s}{i} \rho^{i} (1-\rho)^{k \cdot s - i}$$

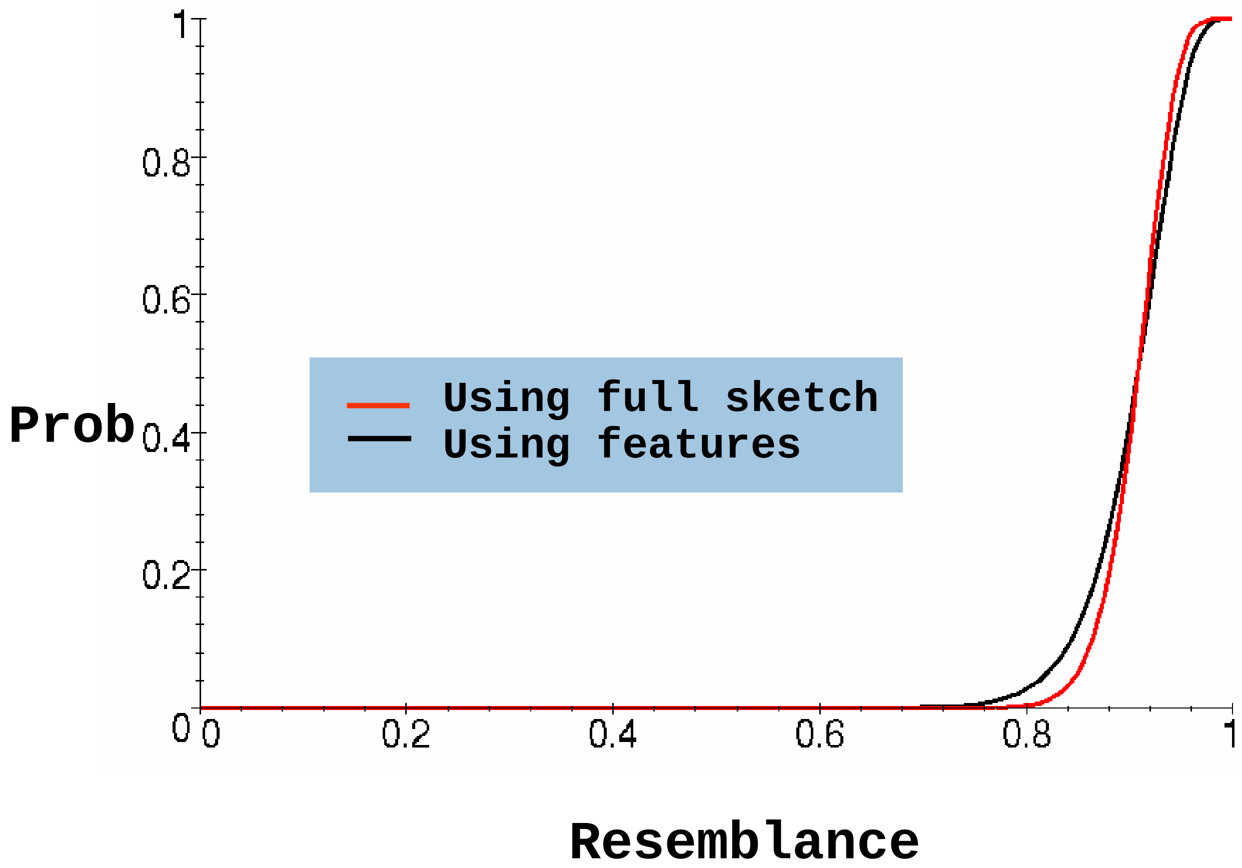
- Using features

$$P = \sum_{i=r}^{k} \binom{k}{i} \rho^{s \cdot i} \left(1-\rho^{s}\right)^{k-i}$$

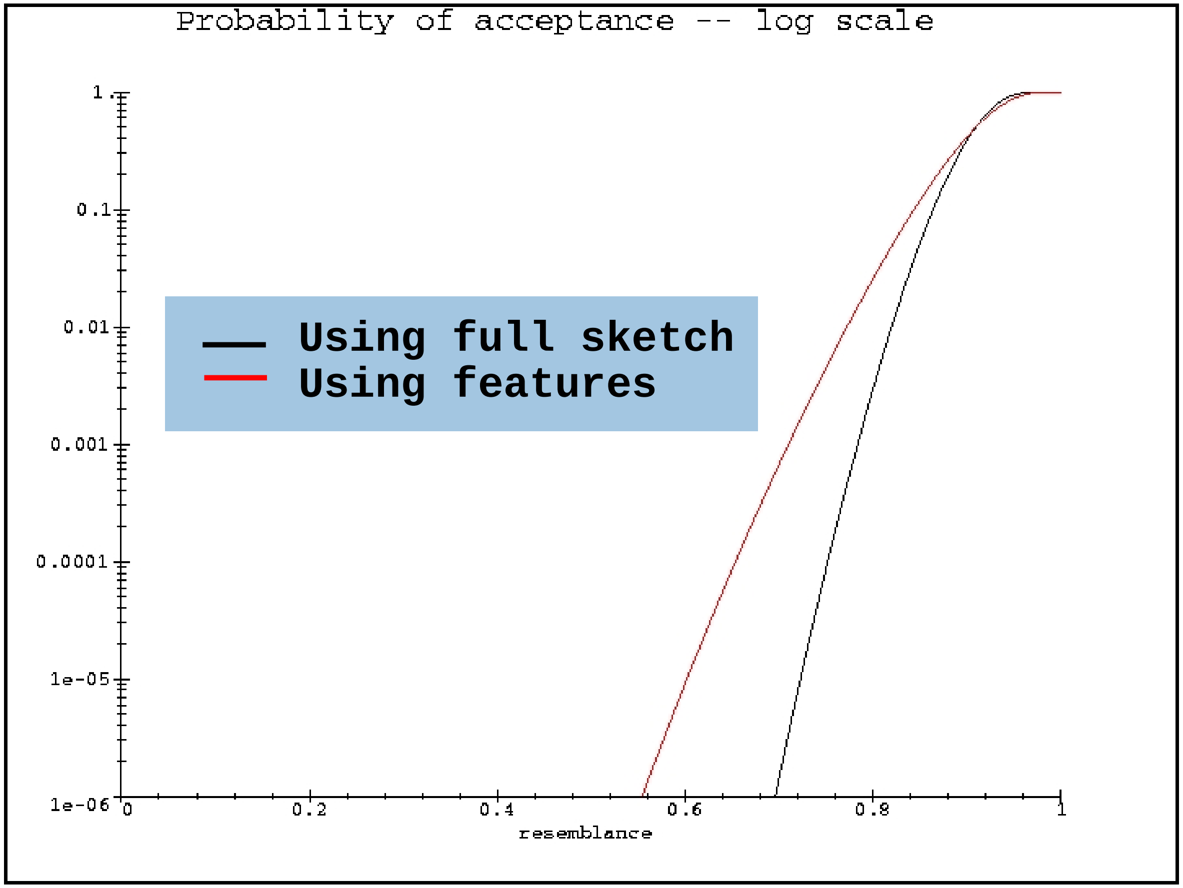
- The second polynomial approximates the first

Features vs. full sketch

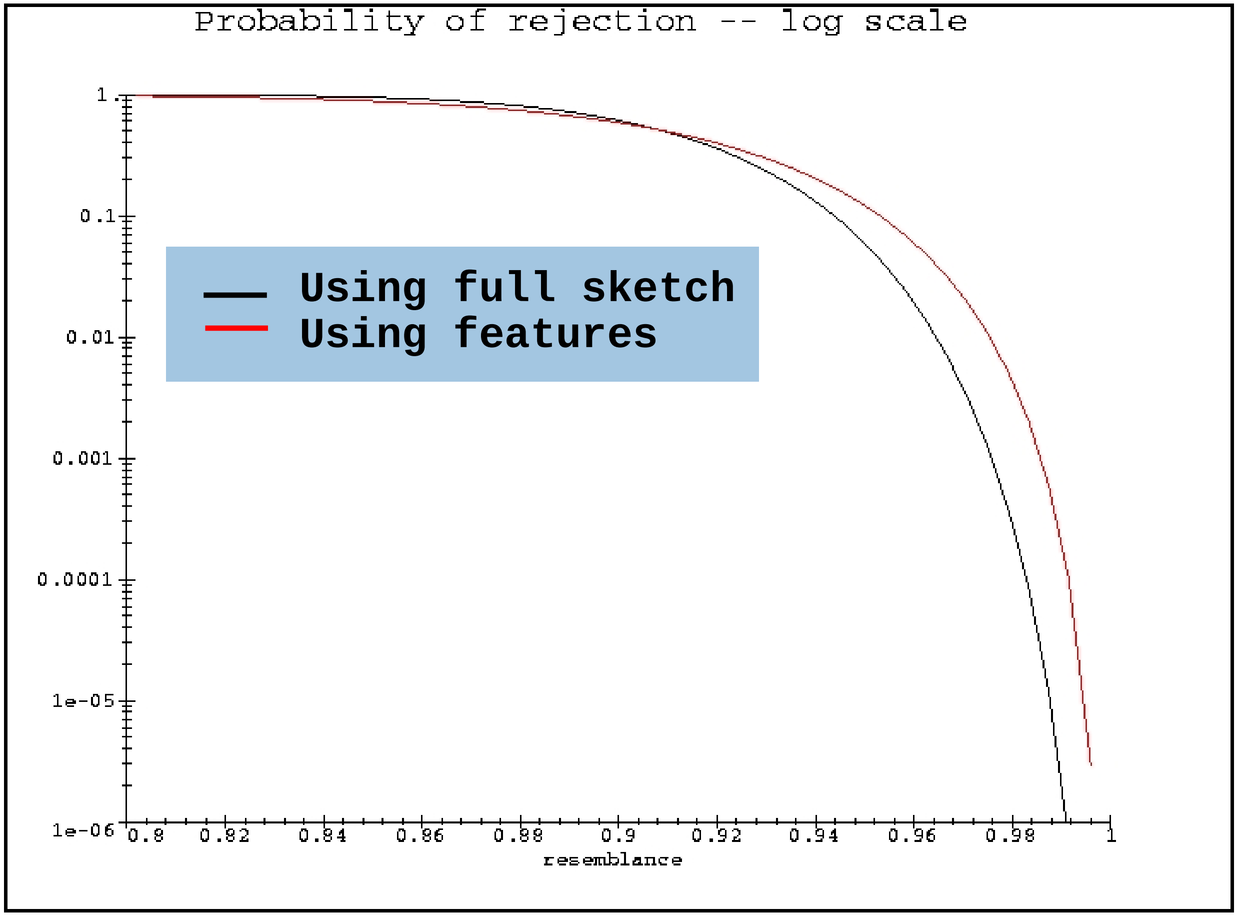
Probability that two pages are deemed fungible



Prob of acceptance - LOG scale



Prob of rejection - LOG scale



Timing

[B, Burrows, & Manasse 98]

- 85M documents
- 1000 word/doc
- 300 MHz machines

⇨ 1 μsec/word ~ 1 CPU day

Using many math and programming tricks plus DCPI tuning we got it down to 1.5 μsec/word !!

- Speed ~ 3 MB/sec (20 X vs full sketch)
  - Speed by 2001 ~ 10-20 MB/sec

One trick based on left-to-right minima [B, Burrows, Manasse]

- For each shingle instead of a permutation **p(s)** compute an injection **h(s)**
- The injection **h(s)** consists of 1 byte + 8 bytes = **p(s)**
- Given **s** compute the lead byte for 8 permutations in parallel via a random linear transformation
- Compute the remaining 8 bytes only if needed
- No theory, but it works! J

How often do we have to compute (or store) the tail ?

- Eventually first byte = 0 so 1/256 of the time.
- Up until the time this happens, roughly the expected number of left to right minima in a permutation with 256 elements, $H_{256} =$ 6.1243… (Because of repetitions, actual number is 7.1204…)

Small scale problems …

- Most duplicates are within the same host
  - Aliasing
    - Unix ln –s is a big culprit!
  - Cookie crumbs problem

8 bytes are enough!

- Same idea with a few twists, threshold = 3 common bytes out of 8.
  - Works only on small scale (say less than 50K documents)
- On a large scale we can use 7 out of 8 bytes
  - Why 7 common bytes is a good idea?
  - Filter is not so sharp

Open problems

- Practical efficient min-wise permutations
- Better filtering polynomials
- Weighted sampling methods
- Document representation as text (using semantics)
- Extensions beyond text: images, sounds, etc. (Must reduce problem to set intersection)
- Extraction of grammar from cookie crumbs URLs (variants are NP-hard)

Conclusions

- Resemblance of documents can be estimated via
  - Translation into set intersection problem
  - Sampling minima
- Filtering is easier than estimating resemblance.
- 30-50 bytes/document is enough for a billion documents, 8 bytes enough for small sets and/or less sharp filters
- Mixing theory & practice is a lot of fun

Further applications & papers

- Chen & al, Selectively estimation for Boolean queries, PODS 2000
- Cohen & al, Finding Interesting Associations, ICDE 2000
- Haveliwala & al, Scalable Techniques for Clustering the Web, *WebDB* 2000
- Chen & al, Counting Twig Matches in a Tree, ICDE 2001
- Gionis & al, Efficient and tunable similar set retrieval, SIGMOD 2001
- Charikar, Similarity Estimation Techniques from Rounding Algorithms, STOC 2002
- Fogaras & Racz, Scaling link based similarity search, WWW 2005 (to appear)
- A bunch of math papers on “Min-Wise Independent Groups”