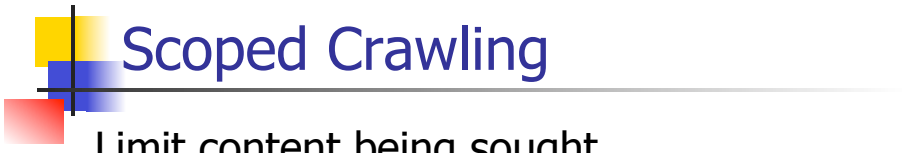
Collection Building II



Scoped Crawling

Limit content being sought
- Topic
- Location
- Format
- Genre
- Language

Weight function w(p) will reflect degree to which p is in scope

Evaluate recall
- % of hand-labeled pages in result
- overlap starting from different seeds

Topical Crawlers

- 1st generation: explore depth d from each topical page
- 2nd generation: differential exploration
  rank potential of link p→q

Topical Crawler II

- 3rd generation: train topic classifier from directory
  - User gives topics
  - Follow links from pages the classifier finds most relevant
  - Also use parent topics
  - Identify authoritative hub pages, favor their links

Incremental Crawling Weighted Freshness (WF)

$WF(t) = \sum_{p \in C(t)} w(p) * f(p,t)$

Where

- $C(t)$ = pages crawled up to time t
- $w(p)$ = weight of page p
- $f(p,t)$ = freshness of local copy of p at time t

Want to maximize steady-state average of $WF(t)$

Choice

- Download new page: improve coverage
- Revisit old page: improve currency

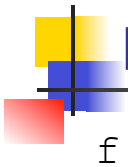
Both can improve link info

Maximizing Freshness

Optimal strategy depends on freshness function

General process

- Estimate change rate of each page p
- Assign portion of crawl rate per page: r(p)
- Scheduling: Find a crawl order that matches the page rates



Binary Freshness

`f(p,t)` = 1 if stored copy = live, else 0

- Revisiting proportional to change rates is sub-optimal
- Studied by Cho and García-Molina: Uniform does better

Binary Freshness Example

Two pages

- e1 changes 9 times/day
- e2 changes 1 time/day

uniformly random in interval

fetch at mid-interval

Look at effect on freshness of different # of fetches per day

(duration of freshness)*
(prob. changes before mid-interval)

Refresh Rates

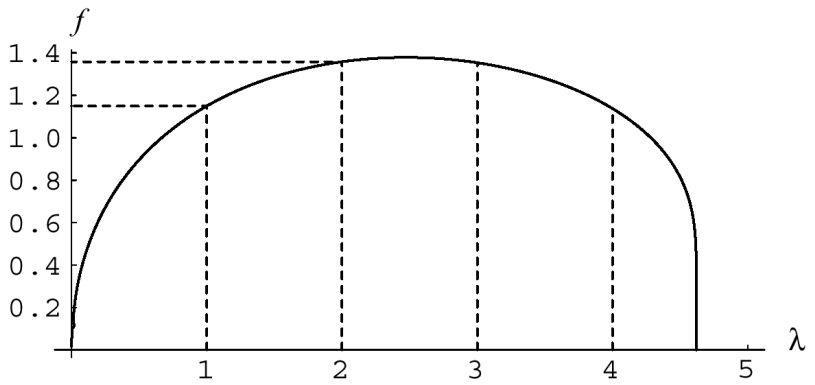
row	$f_1 + f_2$	f_1	f_2	benefit	best	
(a)	1	1	0	$\frac{1}{2} \times \frac{1}{18} = \frac{1}{36}$	0	1
(b)		0	1	$\frac{1}{2} \times \frac{1}{2} = \frac{9}{36}$		
(c)	2	2	0	$\frac{1}{2} \times \frac{1}{18} + \frac{1}{2} \times \frac{1}{18} = \frac{2}{36}$	0	2
(d)		1	1	$\frac{1}{2} \times \frac{1}{18} + \frac{1}{2} \times \frac{1}{2} = \frac{10}{36}$		
(e)		0	2	$\frac{1}{3} \times \frac{2}{3} + \frac{1}{3} \times \frac{1}{3} = \frac{12}{36}$		
(f)	5	3	2	$\frac{3}{36} + \frac{12}{36} = \frac{30}{72}$	2	3
(g)		2	3	$\frac{2}{36} + \frac{6}{16} = \frac{31}{72}$		
(h)	10	9	1	$\frac{9}{36} + \frac{1}{4} = \frac{36}{72}$	7	3
(i)		7	3	$\frac{7}{36} + \frac{6}{16} = \frac{41}{72}$		
(j)		5	5	$\frac{5}{36} + \frac{15}{36} = \frac{40}{72}$		

Observations

- Uniform method is ALWAYS better than proportional method for freshness
- Optimal refresh rate graph ALWAYS has the same shape

Freshness: Optimal Rate

5 pages that change 1, 2, 3, 4, and 5 times a day



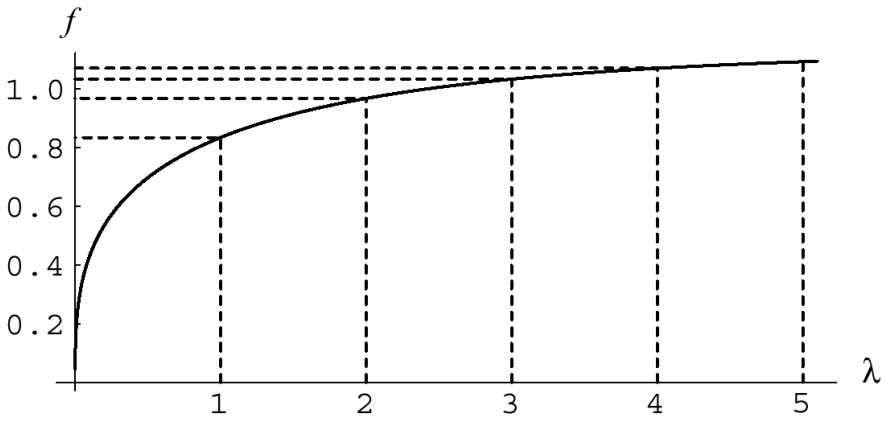
Temporal Freshness

`age(p,t)` = length of time cached p has been different from on-line p at time t

- Rate freshness inversely to age

Optimal strategy crawls faster-changing pages more often

Age: Optimal Rate



Other Metrics

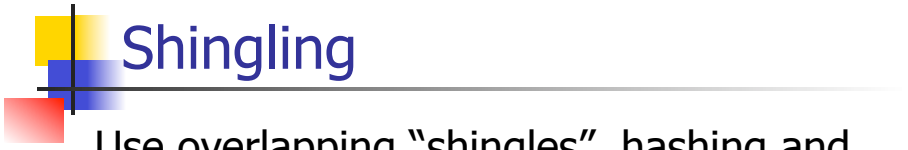
- Percent of content in common with current page
  - Need to model longevity of new content
- Embarrassment level: chance that the user will click on a stale page

Undesirable Content: Duplicates

Reduce caching requirements, or prune search results

Real issue is near-duplicates

- Date stamp, formatting, home link

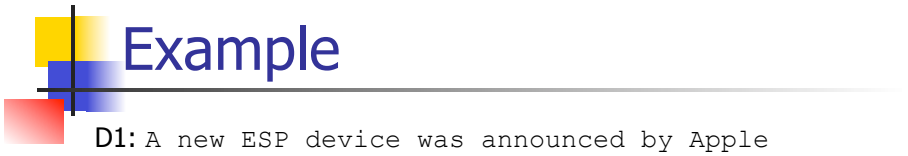


Shingling

Use overlapping "shingles", hashing and permutations to produce a *sketch*, from which similarity can be derived

w-shingle: w consecutive terms from a document

Overlaps with next shingle by w-1 terms



Example

D1: `A new ESP device was announced by Apple Computer at their annual meeting on Tuesday`

D2: `A new ESP device was announced by Apple Corporation at their annual meeting on Tuesday`

3-shingles of D1:

`A new ESP, new ESP device, ESP device was, device was announced, was announced by, …`

S(D) = set of (unique) shingles of document D

Document similarity = Jaccard Coefficient

$SIM(D1,D2) = |S(D1) \cap S(D2)| / |S(D1) \cup S(D2)|$

Fast Estimate of SIM(D1,D2)

First, apply hash function h to all shingles

H(D) = {h(s)|s ∈ S(D)}

Assume h → 1..1000 [2^{64} more usual]

H(D1) =
{150,460,26,821,337,698,14,875,512,504,219,417,733}

H(D2) =
{150,460,26,821,337,698,711,43,208,504,219,417,733}

Fast Estimate, cont.

sorted H(D1) =
{14,26,150,219,337,417,460,504,512,698,733,821,875}

sorted H(D2) =
{26,43,150,208,219,337,417,460,504,698,711,733,821}

- Consider the smallest in each

min(H(D1)) = 14

min(H(D2)) = 26

- Take (n + 200) mod 1000, then take smallest

+200 +400 +600 +800

H(D1)

H(D2)

Sketch of Document

Collect the list of mins

For H(D1) `[14,21,98,17,19]`

For H(D2) `[26,21,98,17,8]`

% similarity of sketch estimates SIM(D1,D2)

How Contrived Was That?

Was trying to sample the hash sets of both

(But I didn't want to sample over 1..1000)

	H1	H2
14	1	0
26	1	1
43	0	1
150	1	1
208	0	1
219	1	1
337	1	1
417	1	1
460	1	1
512	1	0
698	1	1
711	0	1
733	1	1
821	1	1
875	1	0

Getting a Random Sample

My sampling wasn't random

Take a bunch of random permutations

$\Pi_1, \Pi_2, \ldots, \Pi_{200}$

$m_i(D) = \min(\{\Pi_i(v) | v \in H(D)\})$

Sketch of D =

$[m_1(D), m_2(D), \ldots, m_{200}(D)]$

Estimate Pairwise Similarities

If $m_j(D1) = m_j(D2)$, generate (D1, D2, 1)

Notes:

- Random permutation: only needs to be "min-wise" independent – any element has equal chance of being mapped to smallest
- Pre-filter: Sketch of sketches

Crawler Traps

Populate a large URL space with generated content

- calendar pages
- malicious

BEAST: Budget Enforcement with Anti-Spam Techniques

Give each site a "crawl budget" & prioritize based on remaining budget

Web Spam

1. Keyword stuffing
2. Link spam: trying to influence rank
3. Cloaking: different content to crawlers vs. people

Handling 1 & 2

- Classification
- Which features?
  - Hyperlink features
  - Term & phrase frequency
  - DNS lookup stats
  - HTML markup structure

Handling Cloaking

- Giving different content based on `user-agent`

  Need to come in with different user agent or from different IP and compare results
- Redirection Spam

  Script will forward user to another URL

  Use some kind of time-bound evaluation of scripts?