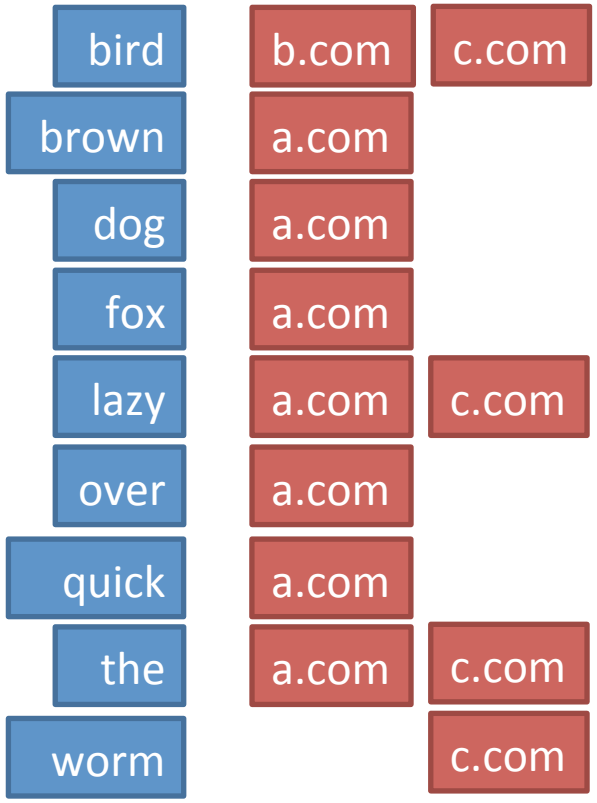
on the web: basic text indexing

a.com "the quick brown fox jumps over the lazy dog"

b.com "a bird in hand is worth two in a bush"

c.com "the lazy bird misses the worm"

bird



looking up a posting takes $O(\log m)$:

keep the term-lists in a sorted structure [hashing - can do better $O(1+m/K)$]

still need to assemble results of a *q-term* query

$O(r\,q)$ if r = # intermediate results in all; what if r is huge ??

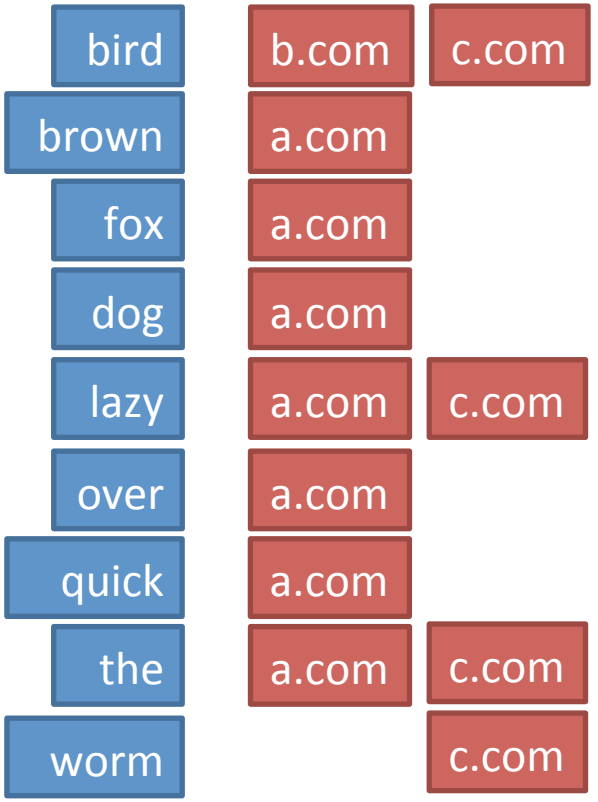
how to create a text index

a.com "the quick brown fox jumps over the lazy dog"

b.com "a bird in hand is worth two in a bush"

c.com "the lazy bird misses the worm"

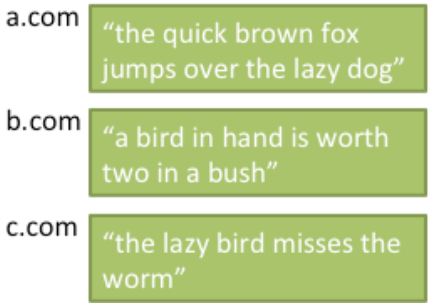
```
class index:
  def create(D) :
    for d in D :
      for w in d :
        i = index.lookup(w)
        if  i < 0 :
          j = index.add(w)
          index.append(j,d.id)
        else:
          index.append(i,d.id)
```



complexity of index creation

n documents, *m* words, *w* words per document

- every word in each document needs to be read, so the complexity is at least $O(n\,w)$



additionally, as *each* word is read:

- we need to lookup the sorted structure of at most *m* words to find out if it has already been inserted before;

  this cost is $O(\log m)$ or $O(1)$ if we use a good hash table
- we must insert the url in the document list for the word

  (after creating a new entry if needed)

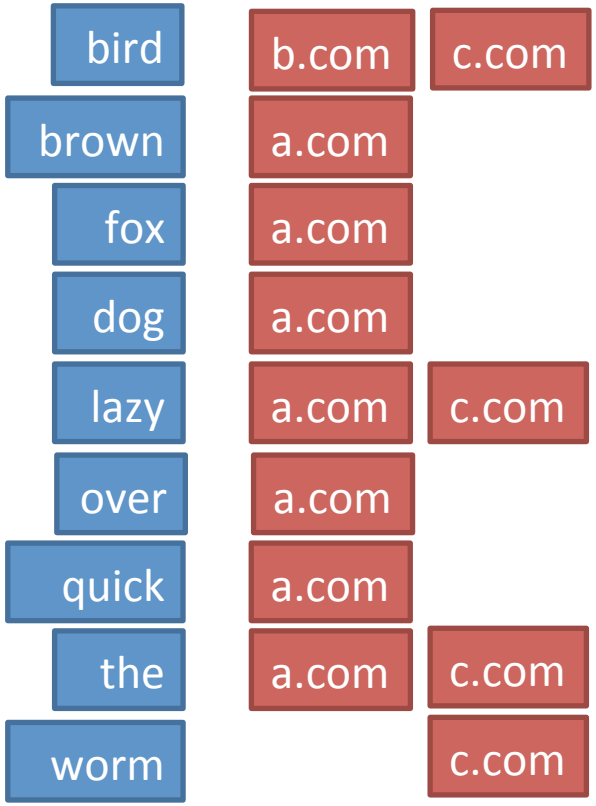
  each of these represents but a *constant* cost per word*

  **there is an important assumption here – HW…*

therefore the complexity of our procedure is $O(n\,w \log m)$

(using a balanced binary tree to store words)

or $O(n\,w)$ (using a hash table to store words)



now that we know what an index is ..

how many web-pages are indexed?

- 2-5 billion
- ✔ 30-40 billion
- 200-300 billion
- trillions

search for a common word, such as ‘a’, or ‘in’ on Google and see how many results are returned

how to arrange the results of search?

what if the result set is *very* large?

- e.g. search for `a' in Google
- also – how to assemble results of a *q-term* query

  O(*r q*) *if r = # intermediate results in all;*
- search for `Clinton plays India cards':

  " Clinton to visit India but Islamabad was not on the cards…"

  OR "Clinton Cards acquired, will save hundreds of jobs in India …"

similarity (from search index) vs importance

- *name the first word that comes to mind …*

  starting with "A"? starting with "G"?

  are some words more important than others; just the common words?
- *top 10 documents matching* `Clinton plays India cards'

  importance = PageRank + … …. but is there anything deeper?

page rank

imagine a `random surfer'

what is the relative probability of visiting a *particular* page?

= *page-rank of the page*

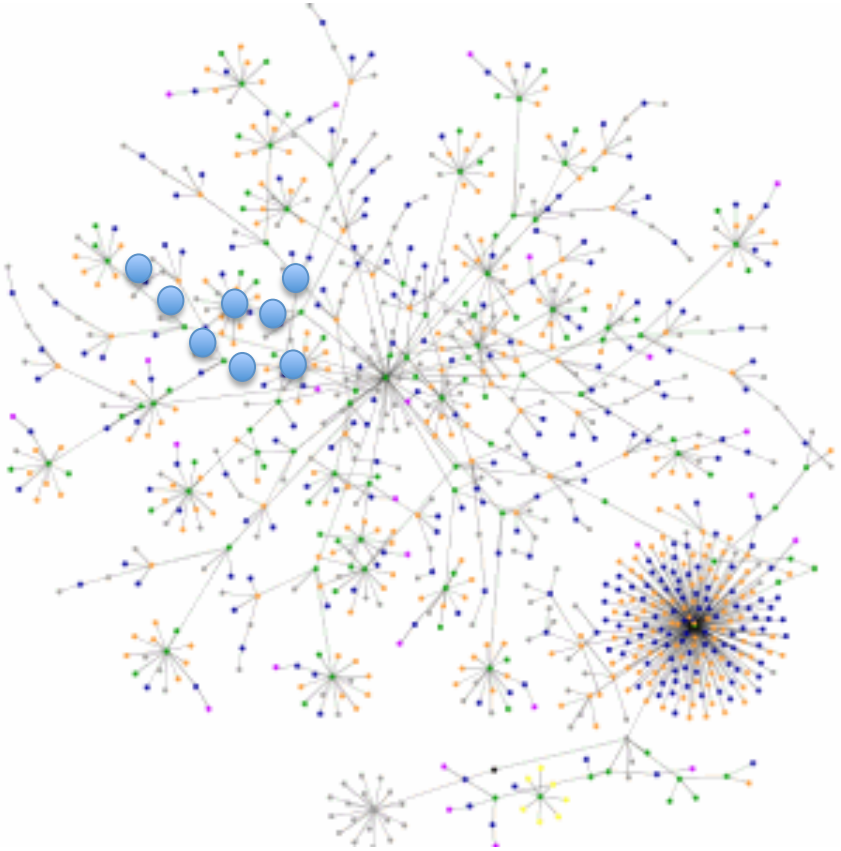
is the number of hyper-links of a page sufficient to compute its page-rank?

yes

✔ *no*

no – because the surfer can re-visit a page via cycles in the graph

page-rank is a *global* property



page-rank is computed iteratively, continuously and in parallel
page-rank is related to the largest eigenvector of an adjacency matrix

page rank and memory

search results ordered by page-rank have proved `intuitive' (=> *$$*)

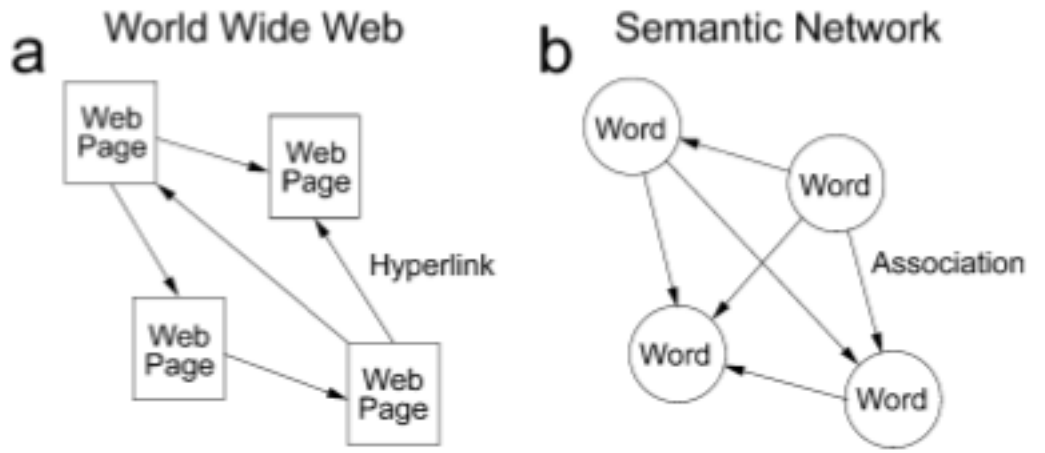
does page-rank provide more insight, say into human memory?

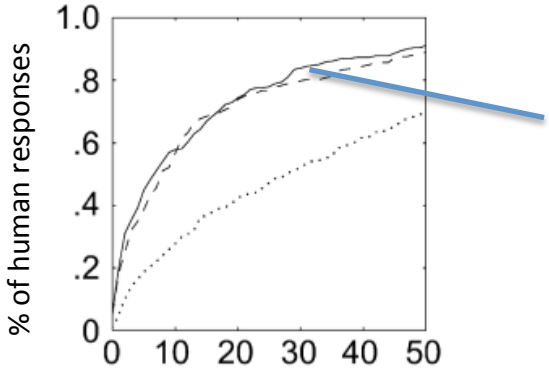
"Google and the Mind" Psychological Science, 2007

1. people asked to form word-word associations
   - *a semantic network*

2. people asked to form letter-word associations

Q: could human response in 2. be predicted from the semantic net of 1.?





page-rank did best

does this mean anything?

search vs. memory

is human memory similar to Google's massive 'index'?

yes

✔ no

most of us are poor at remembering facts

"when was Napolean's defeat at Waterloo?"

we often need context to augment recall

not recognizing a work colleague when seen in a mall …

memories are linked in time

what one did first thing in the morning … and thereafter, etc.

an incident from one's first day at school / college / work …

memories are `fuzzy' – *can you recall every item in your room?*

can be triggered by very sparse matches – *such as a mere smell*

Google and the mind: co-evolution?

page-rank is intuitive, so the more we rely on it

how does this affect accuracy of page-rank?

- *page-rank gets better*
- ✔ *page-rank gets worse*
- *no effect at all*

page-rank relies on hyperlinks

why include hyperlinks? easier to just `Google' anything!

so newer pages have fewer hyperlinks: ***bad*** for page-rank ☒

we find it hard to remember facts, so we increasingly use Google

if our supposedly associative memories rely on building associations, which are strengthened when traversed during recall

➢ the more we use Google the less we can remember! ☒

"The Shallows: What the Internet is doing to our Brains", Nicholas Carr, 2010

Google and the mind: co-evolution? ☑

`mere' indexing is poor at capturing deeper associations between documents, words, and `concepts'

however, as we search and retrieve, we also divulge information on the relative *relevance* of search-results vis-à-vis a query

exploiting such relevance feedback can improve search (augmenting page-rank) ☑

what about us?

exercising recall abilities is not the only time connections are built

we use and create fresh connections when *reasoning*

but reasoning relies on a lot of *facts*

and Google provides these abundantly and easily, encouraging more reasoning, so building more, probably deeper associations! ☑

desktops, email, etc. - `private' search

- ✓ indexing works
- ➢ but what about relevance?
  - no *links* => cannot directly use page-rank
- ➢ need to capture and use other associations
  - named entities (people, places)
  - relevance feedback (by tracking user behavior)
- ➢ duplicate detection and handling
  - multiple versions / formats of the same document
- ❑ is 'search' the only paradigm?
  - topic & activity mining, contextual suggestions

databases & `enterprise search'

all the challenges of `private' search and more:

- context includes the role being played
  - *people play multiple roles*
- taxonomies and classification:
  - manual vs automatic; combinations?
- what about security – role-based access…
- what about `structured' data
  - *SQL is not an answer: text in structured records, linking unstructured documents to structured, `searching' structured records and getting a list of `objects', i.e. related records ….*

searching structured data

consider a *LYRICS* database:

Artist (A)

ArtistID	Name
a1	D12
a2	Eminem
a3	Code Red
a4	jojo

Artist-Album (AB)

AB-ID	ArtistID	AlbumID
ab1	a1	b1
ab2	a2	b1
ab3	a4	b3

Album (B)

AlbumID	Title
b1	D12 World
b2	Off the Wall
b3	jojo

Album-Song (BS)

BS-ID	AlbumID	SongID
bs1	b1	s1
bs2	b2	s2
bs3	b3	s3

Song (S)

SongID	Title	Lyrics
s1	How Come	How come we don't … by... more..
s2	The Kids	And everyone should get along ...
s3	Leave(Get out)	I've been waiting all day…

SQL to get albums with "*World*" in the title:

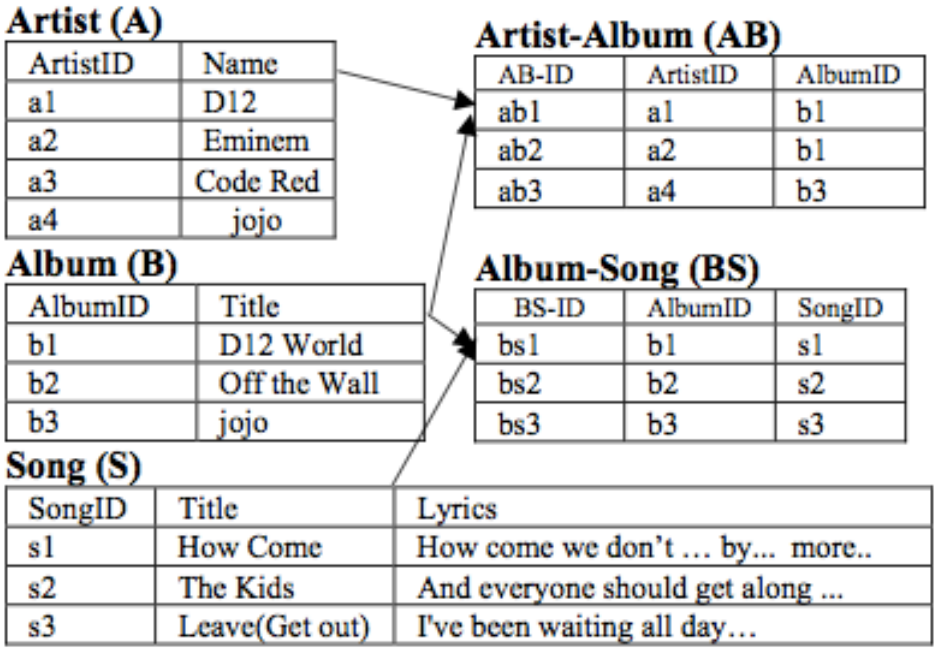
```
Select * from Album B
Where Contains (B.title, 'World' 1) > 0
Order by score(1) desc
```

"Effective keyword search in relational databases", Liu et. SIGMOD06

quiz: searching structured data

how many SQL queries will it take to retrieve

the *names* of each artist and the *lyrics* of every song in an album that has "*World*" in its title



Artist (A)

ArtistID	Name
a1	D12
a2	Eminem
a3	Code Red
a4	jojo

Artist-Album (AB)

AB-ID	ArtistID	AlbumID
ab1	a1	b1
ab2	a2	b1
ab3	a4	b3

Album (B)

AlbumID	Title
b1	D12 World
b2	Off the Wall
b3	jojo

Album-Song (BS)

BS-ID	AlbumID	SongID
bs1	b1	s1
bs2	b2	s2
bs3	b3	s3

Song (S)

SongID	Title	Lyrics
s1	How Come	How come we don't … by... more..
s2	The Kids	And everyone should get along ...
s3	Leave(Get out)	I've been waiting all day…

quiz: searching structured data

how many SQL queries will it take to retrieve

the *names* of each artist and the *lyrics* of every song in an album that has "*World*" in its title

Artist (A)

ArtistID	Name
a1	D12
a2	Eminem
a3	Code Red
a4	jojo

Artist-Album (AB)

AB-ID	ArtistID	AlbumID
ab1	a1	b1
ab2	a2	b1
ab3	a4	b3

Album (B)

AlbumID	Title
b1	D12 World
b2	Off the Wall
b3	jojo

Album-Song (BS)

BS-ID	AlbumID	SongID
bs1	b1	s1
bs2	b2	s2
bs3	b3	s3

Song (S)

SongID	Title	Lyrics
s1	How Come	How come we don't … by... more..
s2	The Kids	And everyone should get along ...
s3	Leave(Get out)	I've been waiting all day…

Query 1: *'World' from Album* *

Query 2: *"lyrics how come by D12"*

Query 3: *"album by D12 and Eminem"*

* *"Effective keyword search in relational databases",* Liu et. SIGMOD06

searching structured data

compare writing SQLs with issuing a 'search' query: *"off the world"*

- partial matches are missed, e.g. "*World*" , "*off the wall"*
- schema needs to be understood
- many queries, or a complex join are needed

but there is more:

- suppose there were multiple databases, each with a different schema, and with partial, or duplicated data?
- most important – some unstructured data in documents, other structured in databases: how to search both *together*

➢ *'searching' structured data well remains a research problem*

other kinds of search

index a object (document) by features (words)

assumption is that query is a bag of words, i.e. features

what if the query is an object

e.g. an image *(Google Goggles)*, fingerprint + iris *(UID*) …*

is an inverted index the best way to search for objects?

yes

✓ *no*

why? – think about this and discuss!

there is another, very powerful method, called:

*Locality Sensitive Hashing***

"compare n pairs of objects in O(n) time"

**Indyk and Motwani '98; Ullman and Rajaraman, Ch 3

*http://uidai.gov.in/

locality sensitive hashing (LSH)

basic idea – object *x* is hashed *h*(*x*) so that

if *x* = *y* or *x* close-to *y* , then *h*(*x*) = *h*(*y*) with *high probability*,

and conversely

if *x* ≠ *y* (*x* far-from *y*) then *h*(*x*) ≠ *h*(*y*) with *high probability*

constructing the hash functions is tricky …

combining random functions from a "locally sensitive" *family*

see Ullman and Rajaraman – Chapter 3

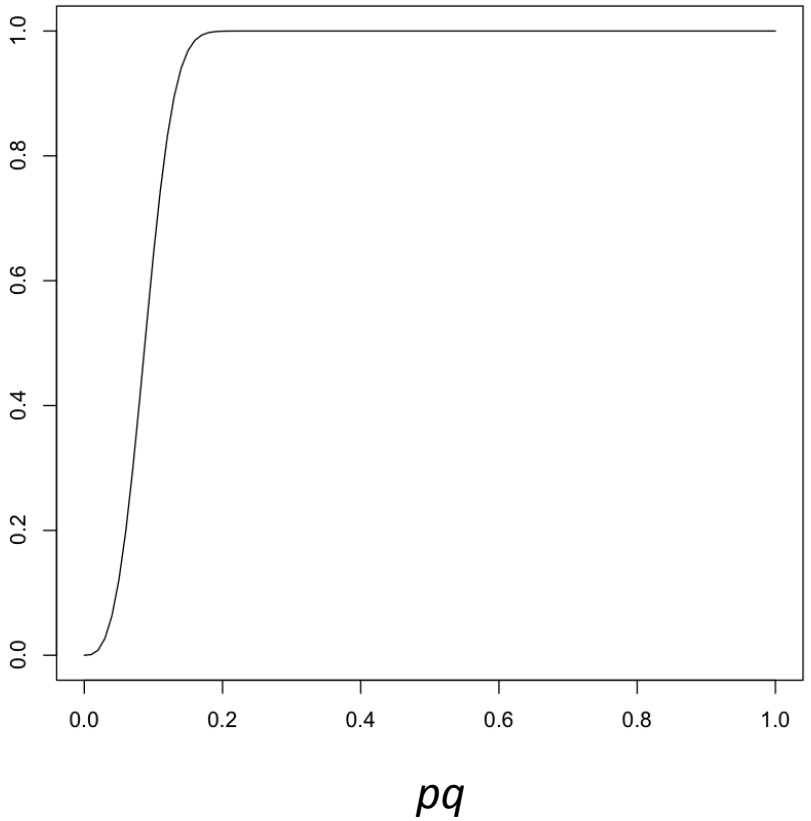
example application: biometric matching

e.g. UID, of a billion+ people, *280+ million* enrolled so far …*

*disclaimer: what UID uses is proprietary, this is merely a motivating example

combining locality-sensitive functions

$$1-(1-(pq)^k)^b$$



pq is the probability of a match in one function; even if moderate the LSH expression *amplifies* this match probability while driving the false-match probability to zero as long as it is reasonably smaller

some `big data' applications of LSH

grouping similar tweets without comparing all pairs

near-duplicates / versions of the same root document

finding patterns in time-series (e.g. sensor data)

resolving identities of people from multiple inputs

…

LSH and ‘dimensionality reduction’

intuition

the `space’ of objects (prints) is *d*-dimensional, (e.g. 1000)

2^d, i.e., lots … of possible objects

LSH *reduces* the dimension to just *b* hash values (e.g. 1024),

further, random hash functions turn out to be locality sensitive*

so similar objects map to `similar’ hash values

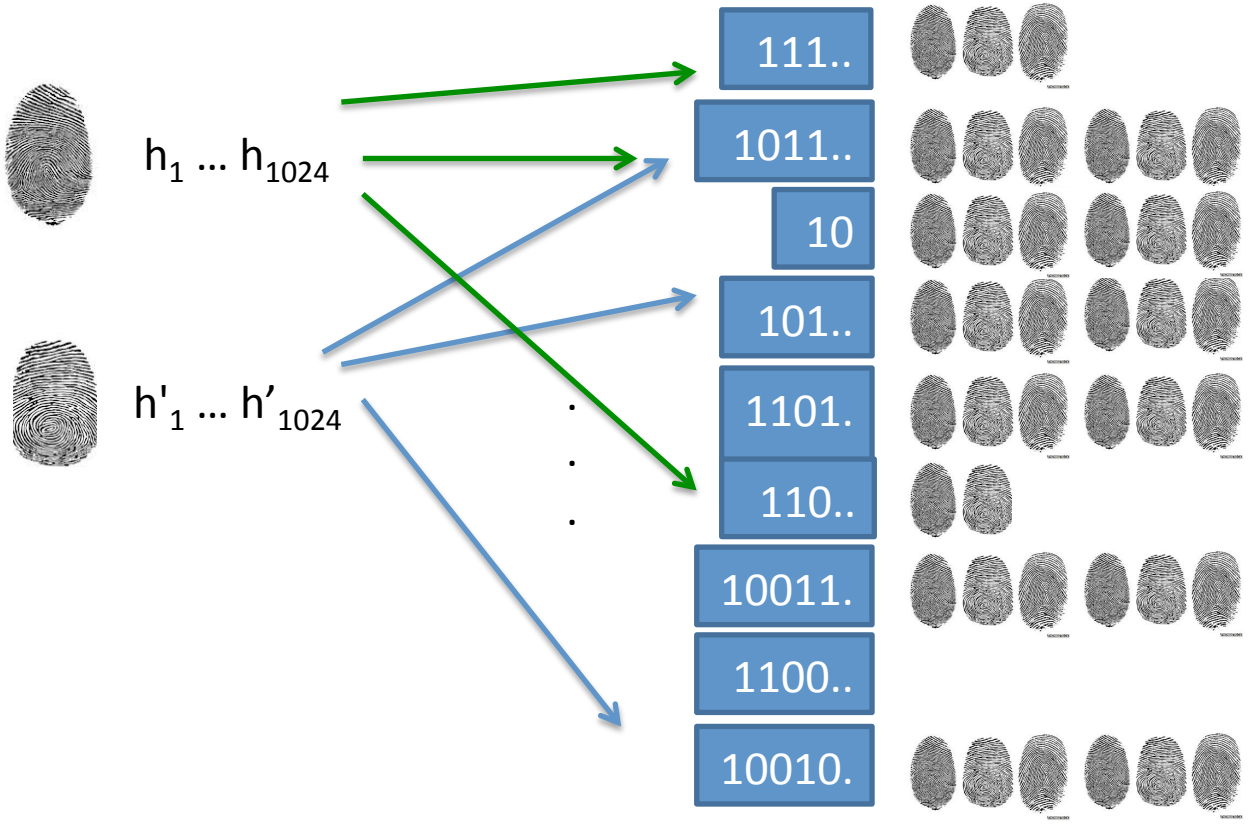
- closely related to other kinds of ‘dimensionality-reduction’
- bit tricky to implement, especially in parallel - …

LSH-based indexing

it might appear that LSH ‘groups’ similar items

instead it computes the *neighborhood* of each item:

e.g. – represent each object (print) by its *b* hash-values



approximate recall: associative memory

do we *store* all objects (images, experiences …)?

"sparse distributed memory" *

pre-dating LSH; also related to high-dimensional spaces, *exploits* vs reduce

consider the space of all 1000-bit vectors; there are lots .. 2^{1000}!

average distance between any two 1000-bit vectors? 500

now – consider a particular vector *x* chosen at random

half of all other vectors differ by < 500 bits, half by more .. obvious

how many differ from *x* by less than 450 bits?

binomial distribution with mean 500, n=1000, so σ = √npq = √250 = 15.8

using a normal approximation – only .0007th are less than 450 bits from x

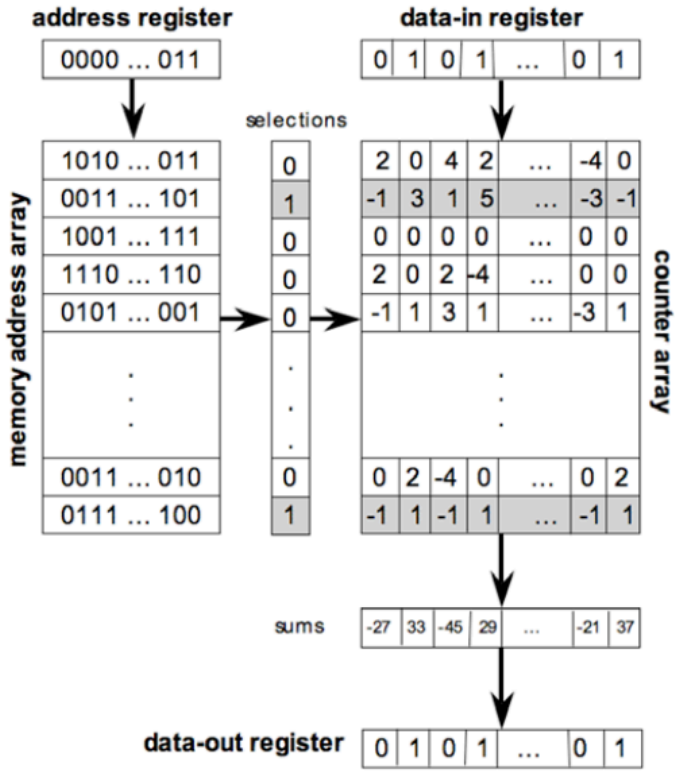
or, <u>most</u> vectors (.998, all but < 2/1000^{ths}), are within 450 and 550 bits away!

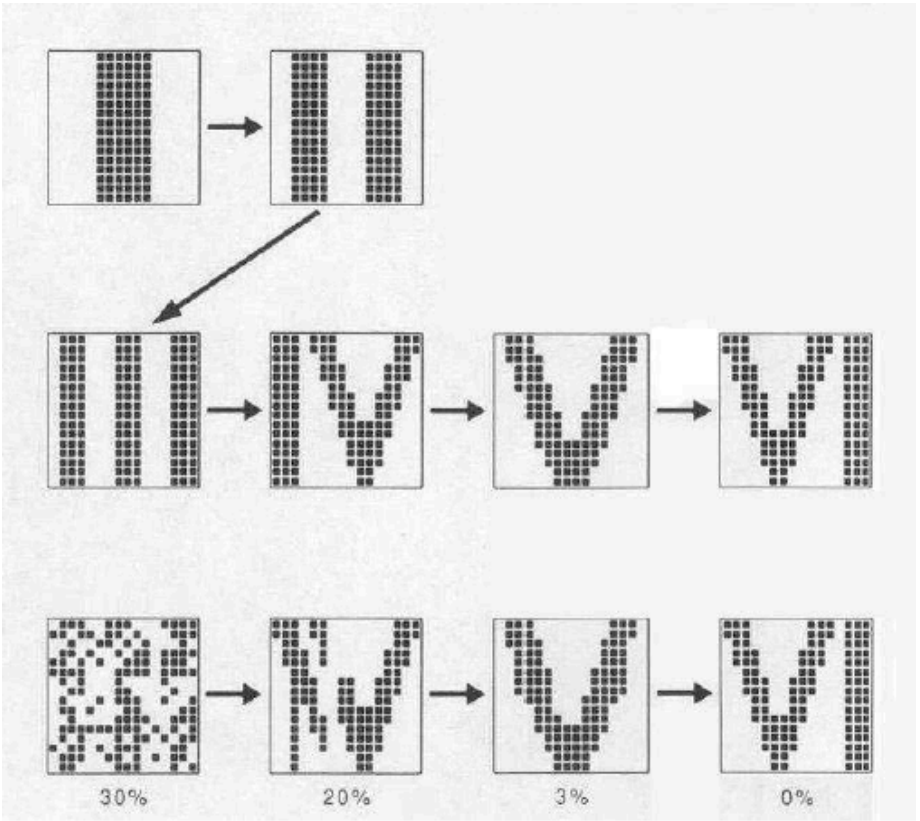
in SDM, concepts are represented by *m* random vectors:

- 'nearby' instances, i.e., even differing in 400 bits, are easily identified
- moreover, SDM shows how to recall by construction – instances accumulate rather than being individually stored

*Pentti Kanerva

sparse-distributed memory at work





P. J. Denning, American Scientist 77 (July-August 1989)

observed 'documents', 'images' or 'objects' are *not* stored
instead these are *reconstructed* 'from memory'
SDMs can store objects address by *themselves*
SDMs can store *sequences* of objects, addressed by preceding elements

`looking' vs searching

- seeing: recognizing objects and activities
- browsing a bookshelf, flipping pages of a book
- looking at data: time-series, histogram, charts

"*visualizing* a scene"
"getting a *feel* for a document, collection or data"

- ❑ clustering and classification
- ❑ topic discovery, summarization
- ❑ correlation and `interestingness'