MMR®
COMP5425

MULTIMEDIA RETRIEVAL

THE UNIVERSITY OF SYDNEY

Week02

Semester 1, 2025

Text Retrieval

- Background
- Textual Information Retrieval

Slides adapted from Christopher D. Manning and Wolfgang Hürst

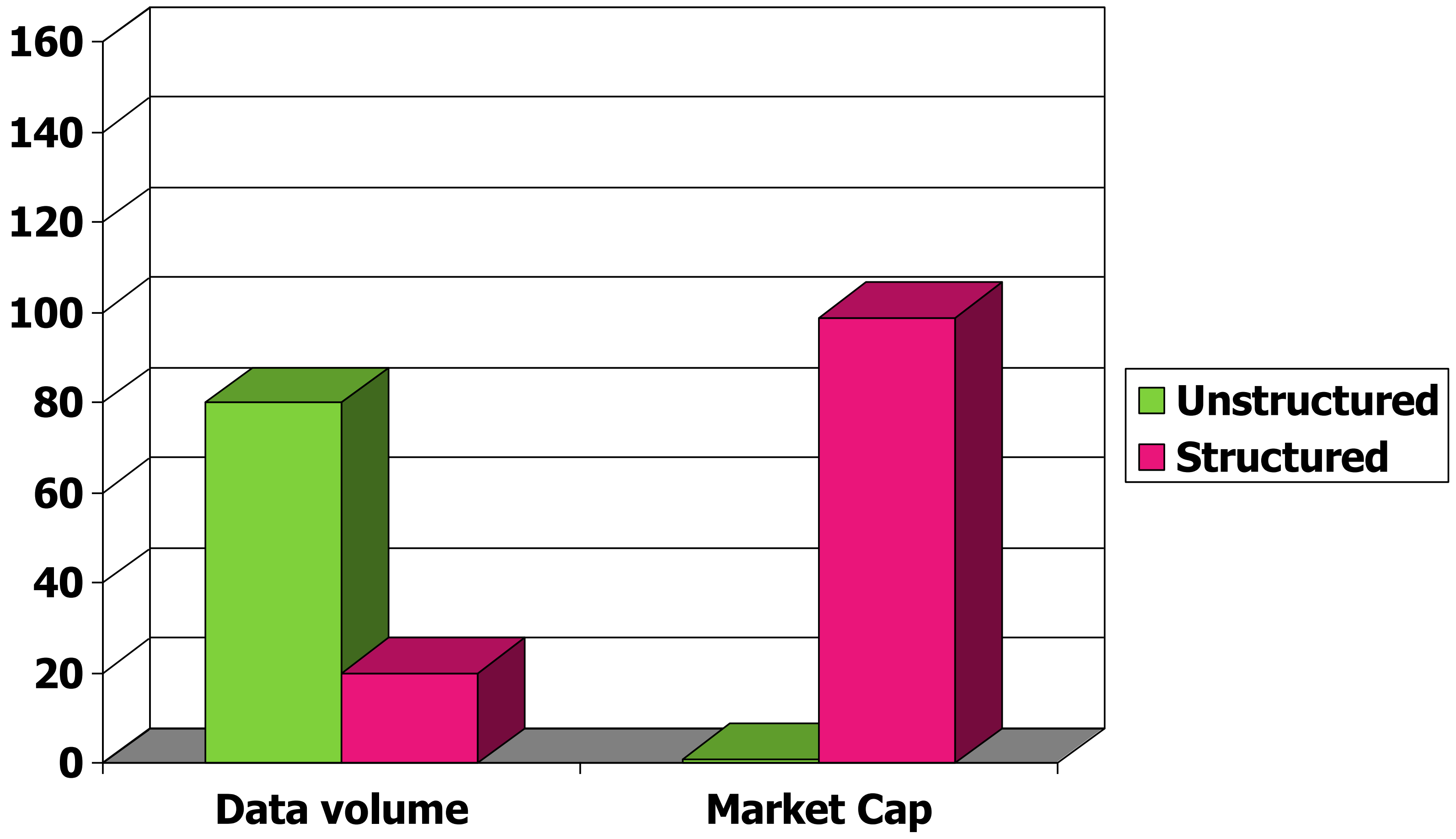
Background

Find a needle from haystacks!!!

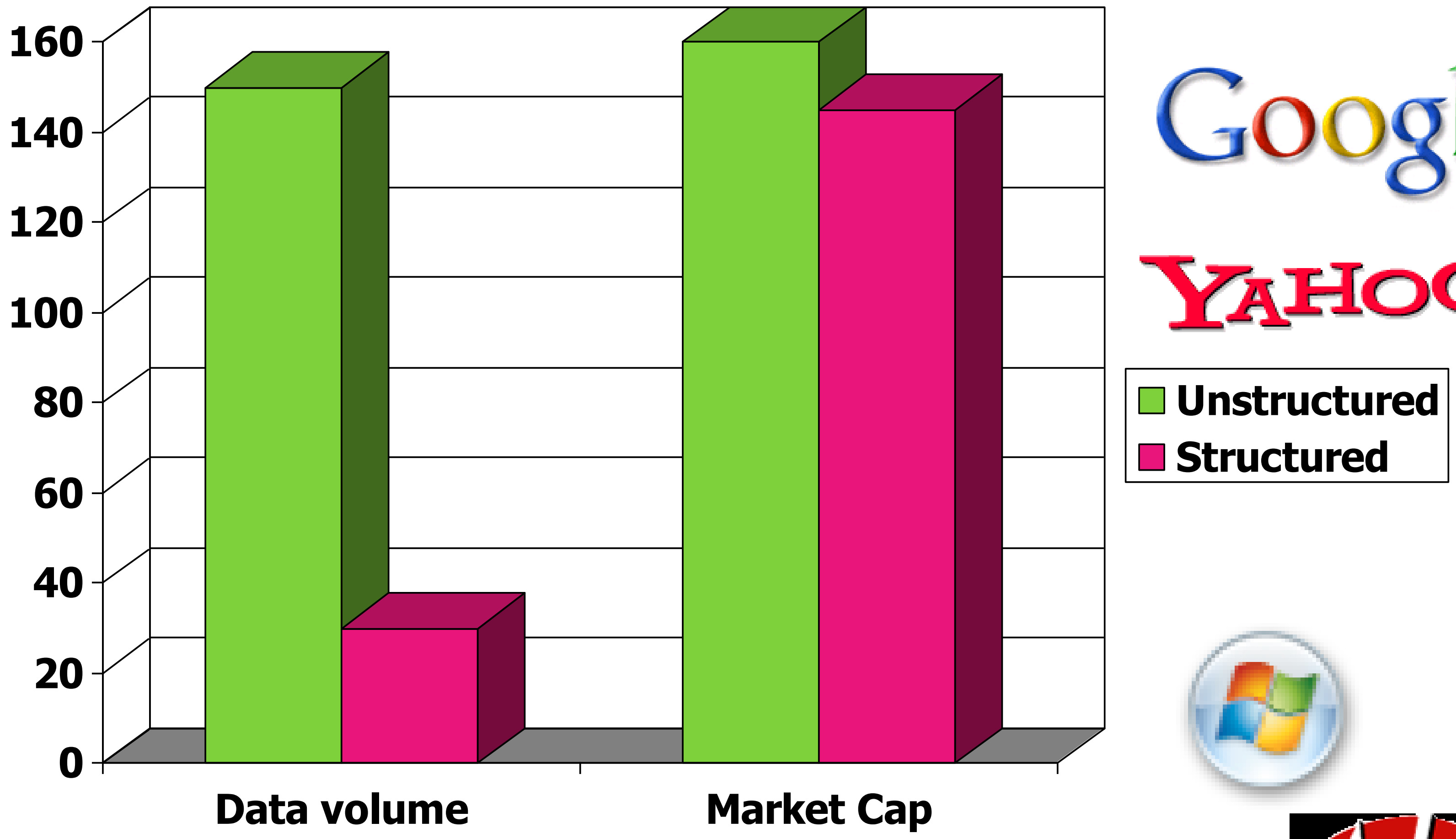
Copyright TREC 2003

- Information is of no use unless you can actually access it.

Unstructured (text) vs. structured (database) data in 1996



Unstructured (text) vs. structured (database) data in 2006



Background

- Information Retrieval (IR) is concerned with developing algorithms and models for retrieving information from document repositories according to an information need.
- With the remarkable growth in the use of computers to process multimedia data, there is also an ever-increasing demand for retrieval of such data.
- Retrieval: Text document and Multimedia document

Some Historical Remarks on IR

- 1950s: Basic idea of searching text with a computer
- 1960s: Key developments, e.g.
  The SMART system (G. Salton, Harvard/Cornell)
  The Crainfield evaluations
- 1970s and 1980s: Advancements of basic ideas
  But: mainly with small test collections
- 1990s: Establishment of TREC (Text Retrieval Conference) series (since 1992 till today)
  - Large text collections
  - Expansion to other fields and areas, e.g. spoken document retrieval, non-english or multi-lingual retrieval, information filtering, user interactions, WWW, video retrieval, etc.

Amit Singhal, Modern Information Retrieval: a Brief Overview, IEEE Bulletin, 2001.

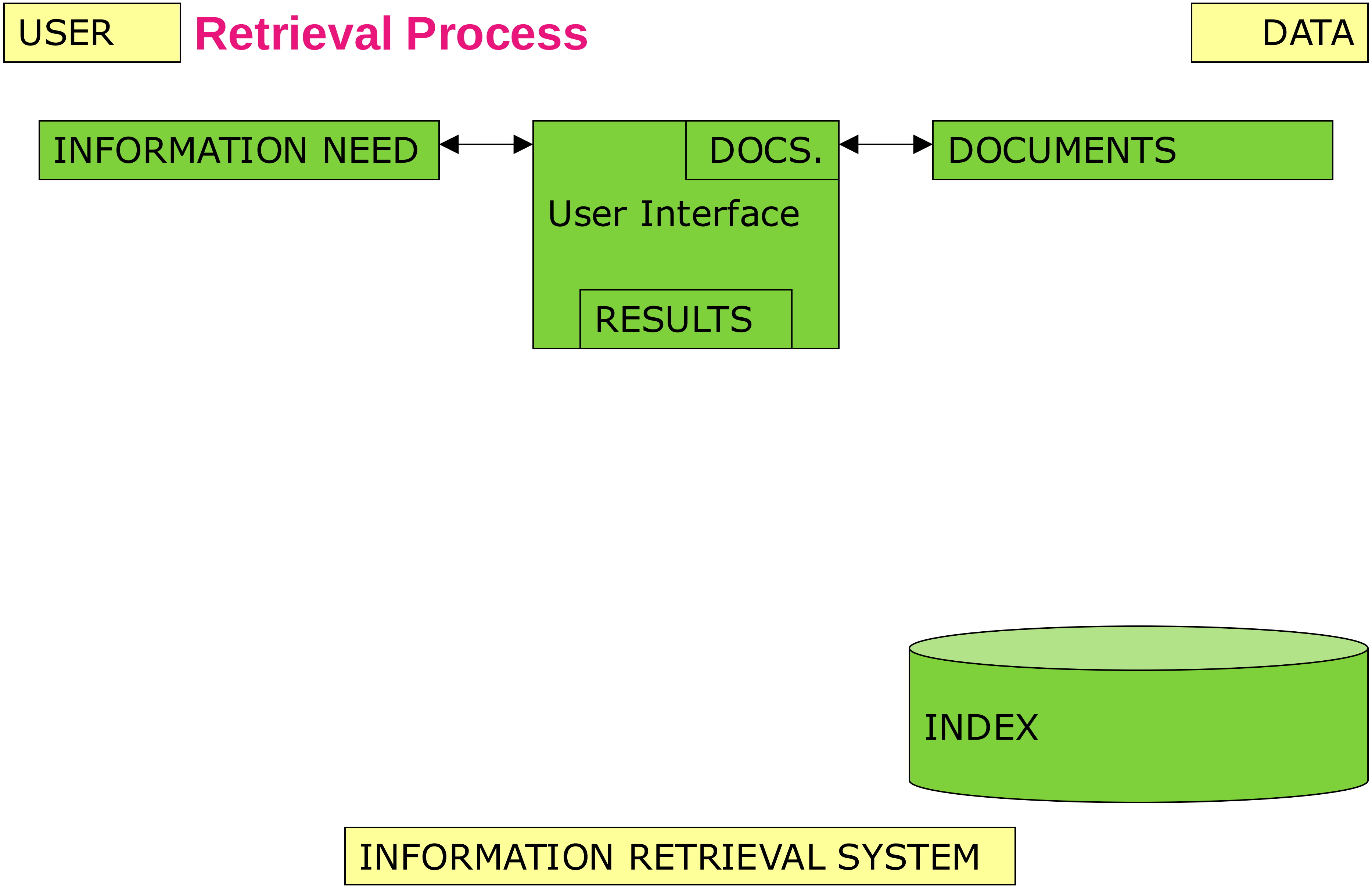
Major Issues in IR

Information Retrieval (IR) deals with the representation, storage, organization of, and access to information items.

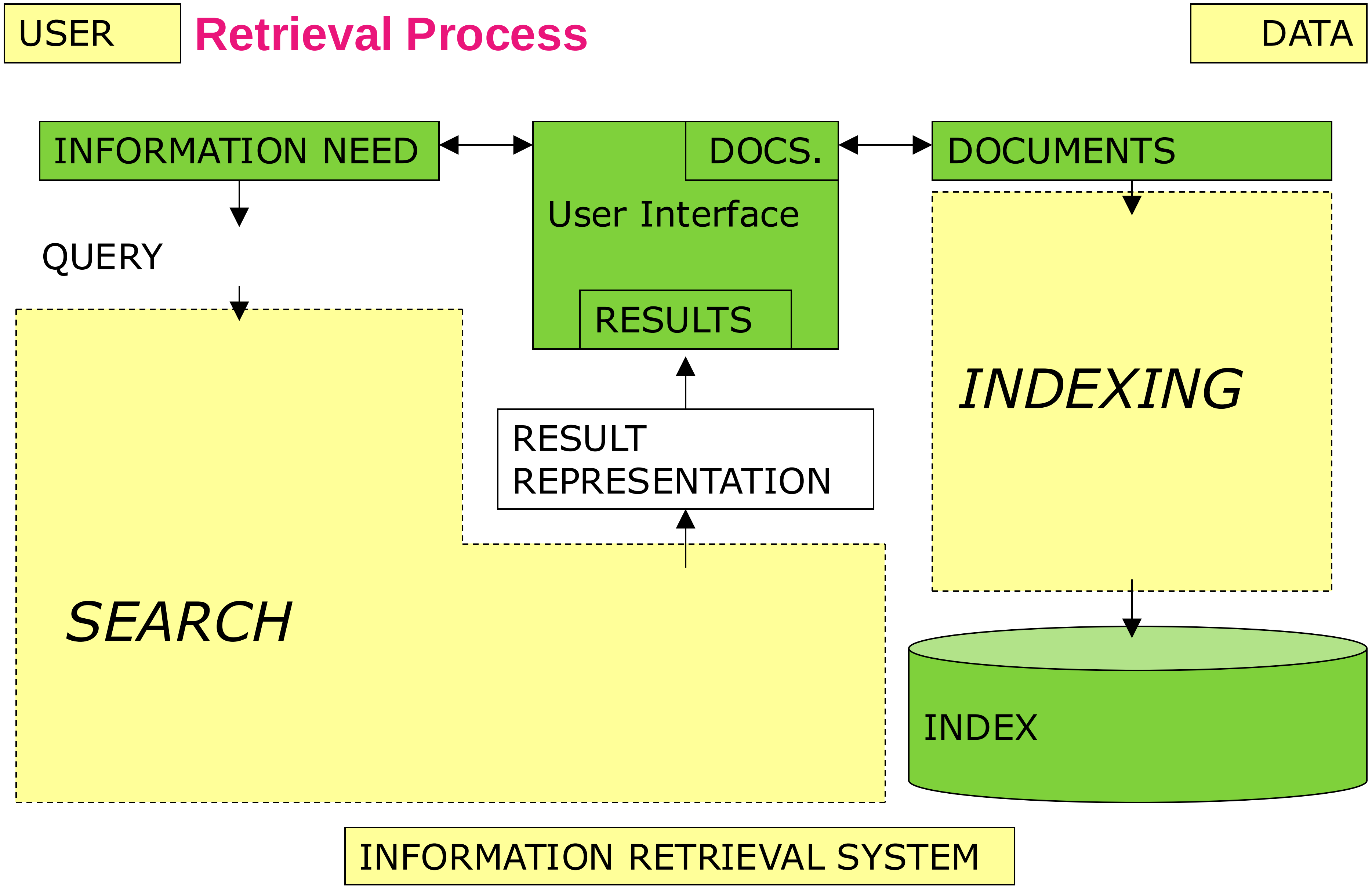
Baeza-Yates and Ribeiro-Neto

- Content analysis
- Indexing
- Ranking
- Interaction
- Results presentation (visualization)

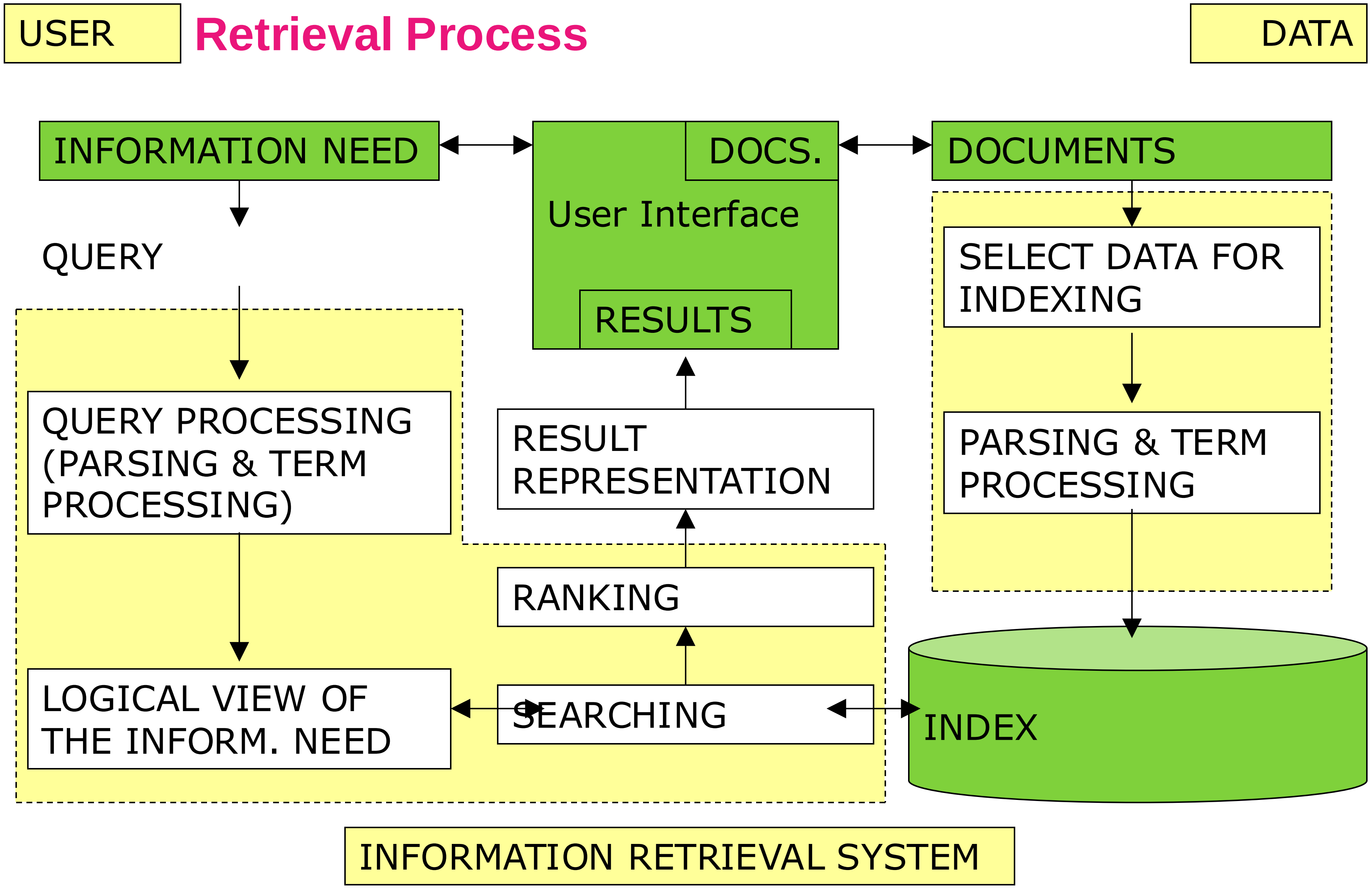
Retrieval Process



Retrieval Process



Retrieval Process



Document Representation

- Terms
  - *hello*, *world*, *information*, *retrieval*, *multimedia*, …
- Boolean Vector
  - Dictionary / Vocabulary
  - The i-th dimension indicates whether the i-th term in the vocabulary appears in the given document

Example

Query: capital AND France (Boolean Query)

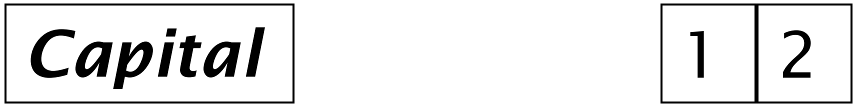
Doc. 1: The capital of France is called Paris.

Doc. 2: Paris is the capital of France.

Doc. 3: The capitals of France and England are called Paris and London, respectively.

Index (Inverted Index)

- For each term *T*, we must store a set of all documents that contain *T*.



- Do we use an array or a list for this?

Term-Document Incidence Matrix

	DOC. 1	DOC. 2	DOC. 3
and	0	0	1
are	0	0	1
called	1	0	1
capital	1	1	0
capitals	0	0	1
England	0	0	1
France	1	1	1
is	1	1	0
London	0	0	1
of	1	1	1
Paris	1	1	1
respectively	0	0	1
The	1	0	1
the	0	1	0

1 = Word appears
0 = Word does not appear

Query: capital AND France

	DOC. 1	DOC. 2	DOC. 3
and	0	0	1
are	0	0	1
called	1	0	1
capital	1	1	0
capitals	0	0	1
England	0	0	1
France	1	1	1
is	1	1	0
London	0	0	1
of	1	1	1
Paris	1	1	1
respectively	0	0	1
The	1	0	1
the	0	1	0

Query: capital AND France

	DOC. 1	DOC. 2	DOC. 3
and	0	0	1
are	0	0	1
called	1	0	1
capital	1	1	0
capitals	0	[illegible]	1
England	0	[illegible]	1
France	1	1	1
is	1	1	0
London	0	0	1
of	1	[illegible]	1
Paris	1	[illegible]	1
respectively	0	0	1
The	1	0	1
the	0	1	0

AND

(1 , 1 , 0)

Term-Document Incidence Matrix

- Search: Very easy, but not very efficient (e.g. 1 000 000 docs, 1 000 terms = Matrix with 1 000 000 000 cells)

- The good news: This matrix is very sparse (i.e. lots of 0’s, only few 1’s)

- Idea: Just store the ‘hits’ (term incidences) using lists

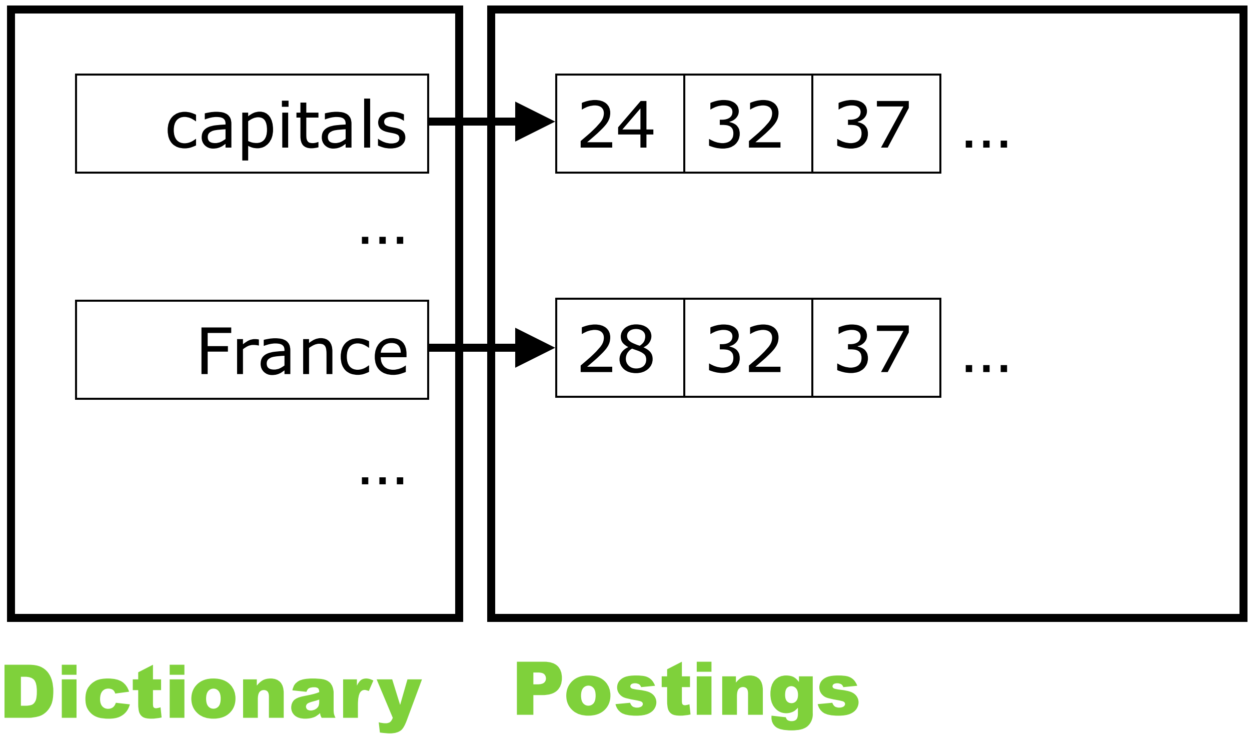
Inverted File

	DOC1	DOC2	DOC3			
and	0	0	1	3		
are	0	0	1	3		
called	1	0	1	1	3	
capital	1	1	0	1	2	
capitals	0	0	1	3		
England	0	0	1	3		
France	1	1	1	1	2	3
is	1	1	0	1	2	
London	0	0	1	3		
of	1	1	1	1	2	3
Paris	1	1	1	1	2	3
respectively	0	0	1	3		
The	1	0	1	1	3	
the	0	1	0	2		

Inverted File

- Main advantage
  - Easy, efficient search
- Disadvantages
  - Storage (10%-100% of doc. size)
  - Modifications (updates, …)
- Often other information is stored as well
  - to support advanced queries
    (e.g. position for phrases)
  - to speed up the search process
    (e.g. frequency for query optimization)
- Bag-of-Words approach
  - Ignores word order
  - What terms / tokens should go into the dictionary?

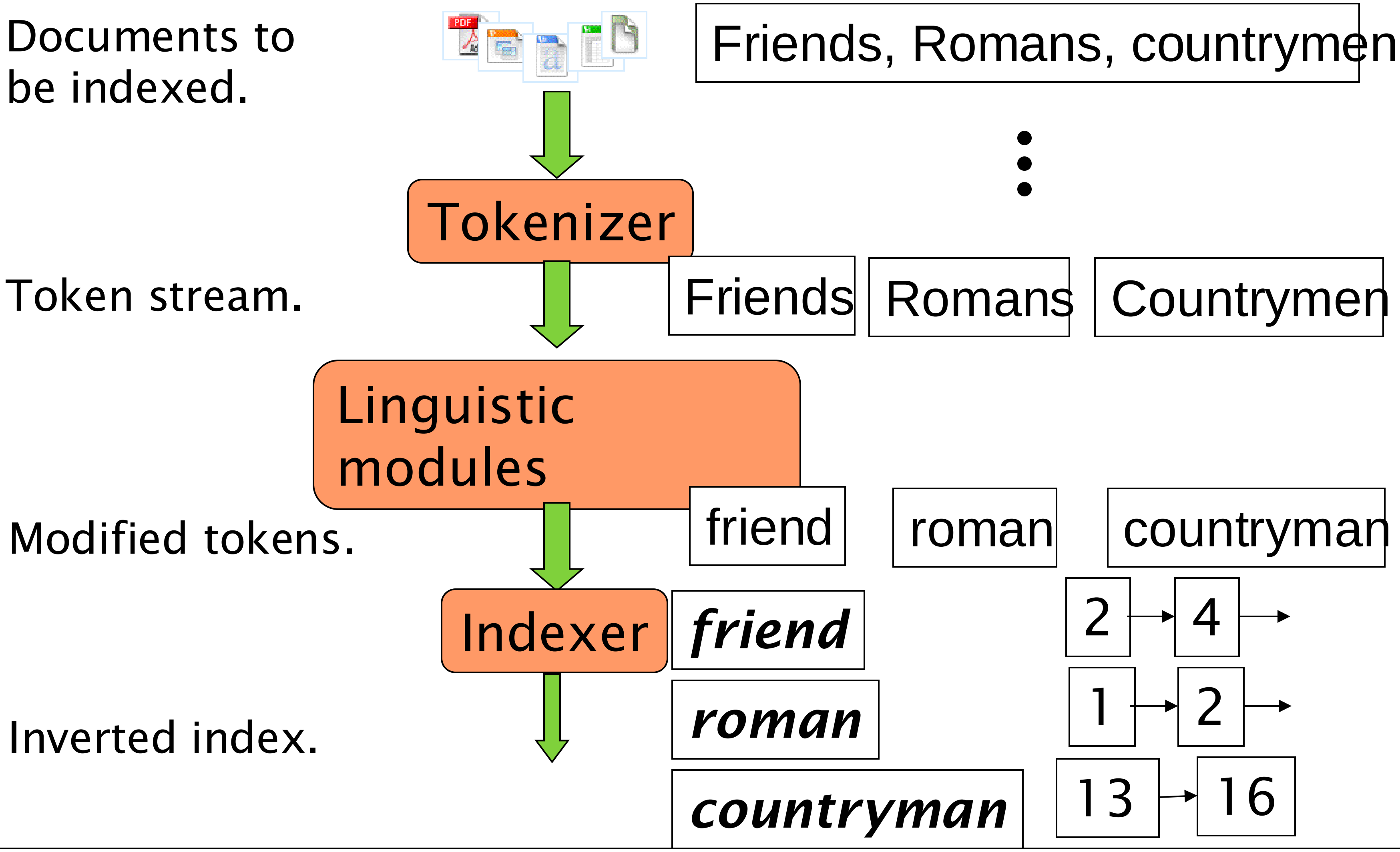
Inverted File



Dictionary: Usually kept in memory (fast!)

Postings: Kept on disks, access via offset

Inverted Index Construction



Indexer steps: Example

- Sequence of (Modified token, Document ID) pairs.

Doc 1

I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

Doc 2

So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

→

Term	Doc #
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

- Multiple term entries in a single document are merged.
- Frequency information is added.

Why frequency?
Will discuss later.

Term	Doc #
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

$\rightarrow$

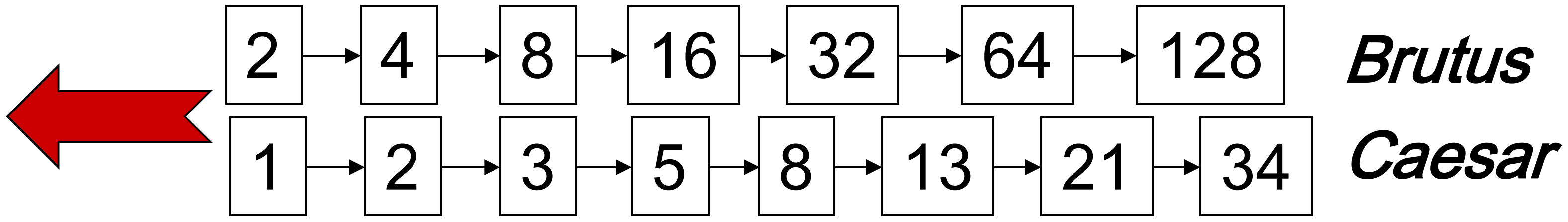
Term	Doc #	Term freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1

Query processing: AND

- Consider processing the query:

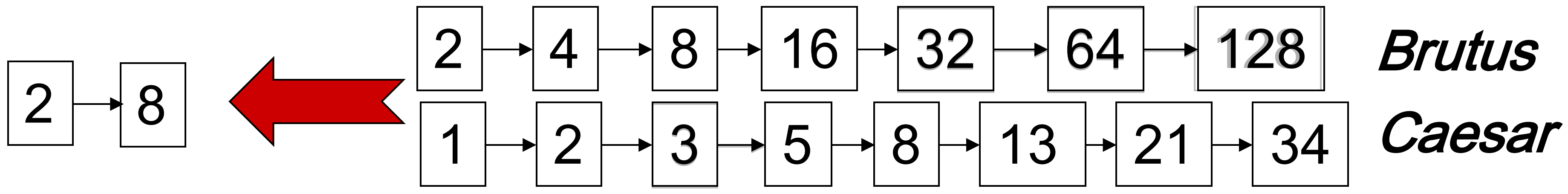
  Brutus *AND* ***Caesar***

  - Locate ***Brutus*** in the Dictionary;
    - Retrieve its postings.
  - Locate *Caesar* in the Dictionary;
    - Retrieve its postings.
  - "Merge" the two postings:



The merge

- Walk through the two postings simultaneously, in time linear in the total number of postings entries



If the list lengths are *x* and *y*, the merge takes O(*x*+*y*) operations.

Crucial: postings sorted by docID.

Boolean queries: Exact match

- The Boolean Retrieval model is being able to ask a query that is a Boolean expression:
  - Boolean Queries are queries using *AND, OR* and *NOT* to join query terms
    - Views each document as a <u>set</u> of words
    - Is precise: document matches condition or not.
- Primary commercial retrieval tool for 3 decades.
- Professional searchers (e.g., lawyers) still like Boolean queries:
  - You know exactly what you’re getting.

Query processing How to parse query

- Parsing expression(not our focus):
  - E.g. (A OR B) AND C
  - Recursive descent parser[1] OR
  - Reverse Polish notation[2] OR
  - Etc.

[1] Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. 2006. Compilers: Principles, Techniques, and Tools (2nd Edition). Addison-Wesley Longman Publishing Co., Inc., USA.
[2] The Calculator Home Page. "Reverse Polish Notation." http://www.calculator.org/rpn.html.Stone, A. "Reverse Polish Notation." http://www-stone.ch.cam.ac.uk/documentation/rrf/rpn.html.

Example: WestLaw

http://www.westlaw.com/

- Largest commercial (paying subscribers) legal search service (started 1975; ranking added 1992)
- Tens of terabytes of data; 700,000 users
- Majority of users *still* use boolean queries
- Example query:
  - What is the statute of limitations in cases involving the federal tort claims act?
  - LIMIT! /3 STATUTE ACTION /S FEDERAL /2 TORT /3 CLAIM
- /3 = within 3 words, /S = in same sentence

Example: WestLaw

http://www.westlaw.com/

- Another example query:
  - Requirements for disabled people to be able to access a workplace
  - disabl! /p access! /s work-site work-place (employment /3 place
- Note that SPACE is disjunction, not conjunction!
- Long, precise queries; proximity operators; incrementally developed; not like web search
- Professional searchers often like Boolean search:
  - Precision, transparency and control
- But that doesn't mean they actually work better….

Beyond Boolean Retrieval

- Boolean retrieval is accurate for precise information need, however, not flexible.
  - Either match, or not; Either too few, or too much
- Position information
  - Phrases: ***Stanford University***
  - Proximity: Find ***Gates*** *NEAR* ***Microsoft***.
  - Need index to capture position information in docs.
- Zones/clusters in documents: Find documents with (*author* = ***Ullman***) *AND* (text contains ***automata***).
- Ranking: which is more relevant to the given query?
  - Most users don't want to wade through 1000s of results.
    - This is particularly true of web search.
  - Need to measure proximity from query to each doc.
  - Need to decide whether documents presented to user are singletons, or a group of documents covering various aspects of the query.

Evidence Accumulation

- 1 vs. 0 occurrence of a search term
  - 2 vs. 1 occurrence
  - 3 vs. 2 occurrences, etc.
  - Usually more seems better
- Need term frequency information in docs
  - If the query term does not occur in the document: score should be 0
  - The more frequent the query term in the document, the higher the score (should be)

Term-document Count Matrices

- Consider the number of occurrences of a term in a document:
  - Each document is a count vector in $\mathbb{N}^{v}$: a column below

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Bag of Words Model

- Vector representation doesn't consider the ordering of words in a document
  - *John is quicker than Mary* and *Mary is quicker than John* have the same vectors
- This is called the bag of words model.
- In a sense, this is a step back: The positional index was able to distinguish these two documents.

Term Frequency *tf*

- The term frequency $tf_{t,d}$ of term *t* in document *d* is defined as the number of times that *t* occurs in *d*.
- We want to use *tf* when computing query-document match scores. But how?
- Raw term frequency is not what we want:
  - A document with 10 occurrences of the term is more relevant than a document with one occurrence of the term.
  - But not 10 times more relevant.
- Relevance does not increase proportionally with term frequency.

Log-frequency Weighting

- The log frequency weight of term t in d is

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

- 0 → 0, 1 → 1, 2 → 1.3, 10 → 2, 1000 → 4, etc.
- Score for a document-query pair: sum over terms *t* in both *q* and *d*:
- score $= \sum_{t \in q \cap d} (1 + \log \mathrm{tf}_{t,d})$

- The score is 0 if none of the query terms is present in the document.

Document Frequency

- Rare terms are more informative than frequent terms
  - Frequent terms: *a, the, to*, …
- Consider a term in the query that is rare in the collection (e.g., *arachnocentric*)
- A document containing this term is very likely to be relevant to the query *arachnocentric*
- → We want a high weight for rare terms like *arachnocentric*.

Document Frequency

- Consider a query term that is frequent in the collection (e.g., *high, increase, line*)
- A document containing such a term is more likely to be relevant than a document that doesn't, but it's not a sure indicator of relevance.
- For frequent terms, we want positive weights for words like *high, increase, and line*, but lower weights than for rare terms.
- We will use document frequency (df) to capture this in the score.
- df ($\leq N$) is the number of documents that contain the term

idf Weight

- df_t is the <u>document</u> frequency of *t*: the number of documents that contain *t*
  - df is a measure of the informativeness of *t*
- We define the idf (inverse document frequency) of *t* by

$$\mathrm{idf}_t = \log_{10} N/\mathrm{df}_t$$

  - We use log N/df_t instead of N/df_t to "dampen" the effect of idf.

Will turn out the base of the log is immaterial.

idf example, suppose *N*= 1 million

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1,000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

There is one idf value for each term *t* in a collection.

Collection vs. Document Frequency

- The collection frequency of *t* is the number of occurrences of *t* in the collection, counting multiple occurrences.
- Example:

Word	Collection frequency	Document frequency
insurance	10440	3997
try	10422	8760

- Which word is a better search term (and should get a higher weight)?

tf-idf Weighting

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$\mathrm{w}_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \times \log N / \mathrm{df}_t$$

- Best known weighting scheme in information retrieval
- Note: the “-” in tf-idf is a hyphen, not a minus sign!
- Alternative names: tf.idf, tf x idf
- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection

Binary → Count → Weight matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Each document is now represented by a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$

Documents as vectors

- So we have a |V|-dimensional vector space
- Terms are axes of the space
- Documents are points or vectors in this space
- Very high-dimensional: hundreds of millions of dimensions when you apply this to a web search engine
- This is a very sparse vector - most entries are zero.

Queries as vectors

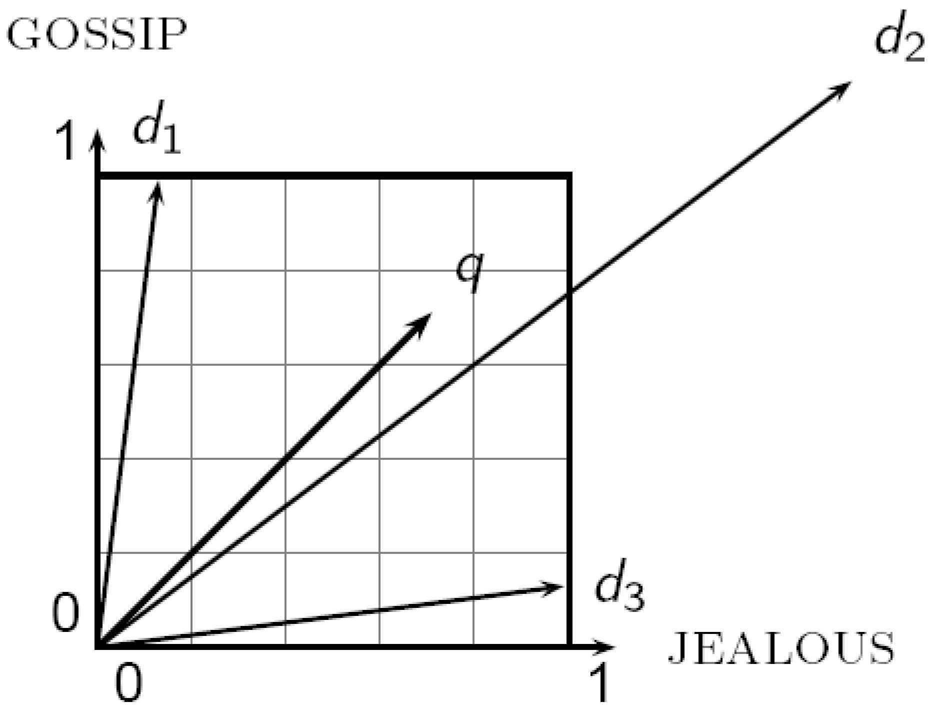
- Key idea 1: Do the same for queries: represent them as vectors in the space
- Key idea 2: Rank documents according to their proximity to the query in this space
  - proximity = similarity of vectors
  - proximity ≈ inverse of distance
- Recall: We do this because we want to get away from the you're-either-in-or-out Boolean model.
- Instead: rank more relevant documents higher than less relevant documents

Formalizing Vector Space Proximity

- First cut: distance between two points
  - (= distance between the end points of the two vectors)
- Euclidean distance?
- Euclidean distance is a bad idea . . .
- . . . because Euclidean distance is large for vectors of different lengths.

Why distance is a bad idea

The Euclidean distance between $\vec{q}$ and $\vec{d_2}$ is large even though the distribution of terms in the query $\vec{q}$ and the distribution of terms in the document $\vec{d_2}$ are very similar.



From Angles to Cosines

- The following two notions are equivalent.
  - Rank documents in <u>decreasing</u> order of the angle between query and document
  - Rank documents in <u>increasing</u> order of cosine(query,document)
- Cosine is a monotonically decreasing function for the interval [0^o, 180^o]

Cosine (query, document)

Dot product

Unit vectors

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q}\bullet\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|}\bullet\frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

q_i is the tf-idf weight of term *i* in the query
d_i is the tf-idf weight of term *i* in the document
$\cos(\vec{q},\vec{d})$ is the cosine similarity of $\vec{q}$ and $\vec{d}$ … or, equivalently, the cosine of the angle between $\vec{q}$ and $\vec{d}$.

Cosine similarity amongst 3 documents

How similar are the novels
SaS: *Sense and Sensibility*
PaP: *Pride and Prejudice*, and
WH: *Wuthering Heights*?

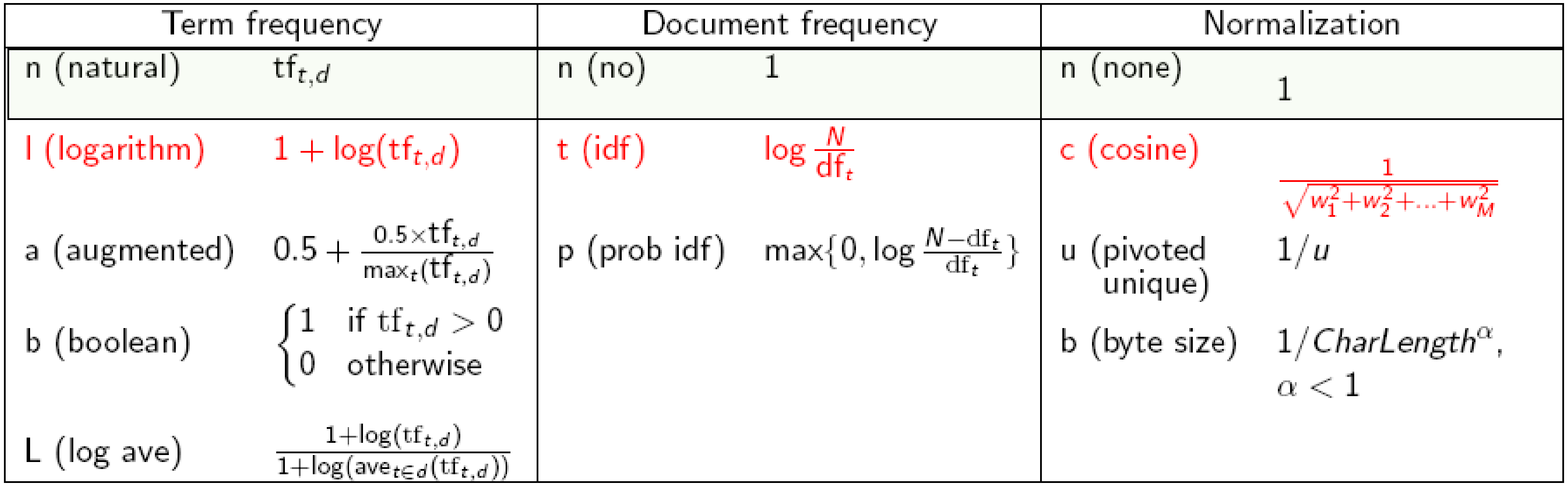
term	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6
wuthering	0	0	38

Term frequencies (counts)

Computing Cosine Scores

```
CosineScore($q$)
 1  float $Scores[N] = 0$
 2  float $Length[N]$
 3  for each query term $t$
 4  do calculate $w_{t,q}$ and fetch postings list for $t$
 5       for each pair($d$, $tf_{t,d}$) in postings list
 6       do $Scores[d] += w_{t,d} \times w_{t,q}$
 7  Read the array $Length$
 8  for each $d$
 9  do $Scores[d] = Scores[d]/Length[d]$
10  return Top $K$ components of $Scores[]$
```

tf-idf weighting has many variants



Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2 + w_2^2 + \ldots + w_M^2}}$
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N - \mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/\mathit{CharLength}^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1 + \log(\mathrm{tf}_{t,d})}{1 + \log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$				

Columns headed ‘n’ are acronyms for weight schemes.

Why is the base of the log in idf immaterial?

Summary – vector space ranking

- Represent the query as a weighted tf-idf vector
- Represent each document as a weighted tf-idf vector
- Compute the cosine similarity score for the query vector and each document vector
- Rank documents with respect to the query by score
- Return the top K (e.g., $K = 10$) to the user

More than tf-idf

- Limitation of TF-IDF
  - TF-IDF assume linear relationship between freq and importance. Higher freq != more important.
    - E.g. machine learning vs. gradient descent
  - TF-IDF does not consider document length
    - Longer document can be ranked higher
  - TF-IDF can be overly sensitive to skewness of corpus
    - In the machine learning themed corpus, the importance of machine learning related terms can be underestimated.

BM25

- Best Matching 25[1]
- Addresses issues of TF-IDF
  - Issues: linear relationship
    - Solution: BM25 introduces non-linear tunable saturation
  - Issues: Bias towards
    - Solution: BM25 introduces non-linear length normalization

[1]Robertson, S.E., & Zaragoza, H. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr., 3*, 333-389.

BM25

$$\mathrm{IDF}(q_i) = \log\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}\right) \tag{1}$$

$$\mathrm{TF}_{\mathrm{BM25}}(q_i, d) = \frac{f_{i,d} \cdot (k_1 + 1)}{f_{i,d} + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{\mathrm{avgdl}}\right)} \tag{2}$$

$$\mathrm{BM25}(Q, d) = \sum_{q_i \in Q} \mathrm{IDF}(q_i) \cdot \mathrm{TF}_{\mathrm{BM25}}(q_i, d) \tag{3}$$

- N: Total number of documents
- $n(q_i)$: Number of documents containing term q_i
- $f_{i,d}$: Frequency of term q_i in document d
- $|d|$: Document length (number of terms)
- avgdl: Average document length in corpus
- $k_1 \in [1.2, 2.0]$: Term frequency saturation parameter
- $b \in [0, 1]$: Document length normalization parameter (default: 0.75)

Beyond All of those

- More advanced methods are available
  - Language model
  - Semantic model
  - … …
- IR is still a very active area
  - SIGIR is the annual conference.
  - IR performance keeps improving.

Relevance Feedback & Query Expansion

- Personalized Service
  - Focus on relevance feedback
- The complete landscape
  - Global methods
    - Query expansion
    - Thesauri
    - Automatic thesaurus generation
  - Local methods
    - Relevance feedback
    - Pseudo relevance feedback

Relevance Feedback

- Relevance feedback: user feedback on relevance of docs in initial set of results
  - User issues a (short, simple) query
  - The user marks some results as relevant or non-relevant.
  - The system computes a better representation of the information need based on feedback.
  - Relevance feedback can go through one or more iterations.
- Idea: it may be difficult to formulate a good query when you don’t know the collection well, so iterate

Key concept: Centroid

- The centroid is the center of mass of a set of points
- Recall that we represent documents as points in a high-dimensional space
- Definition: Centroid

$$\vec{\mu}(C) = \frac{1}{|C|} \sum_{d \in C} \vec{d}$$

where C is a set of documents.

Rocchio Algorithm

- The Rocchio algorithm uses the vector space model to pick a relevance fed-back query
- Rocchio seeks the query $\vec{q}_{opt}$ that maximizes

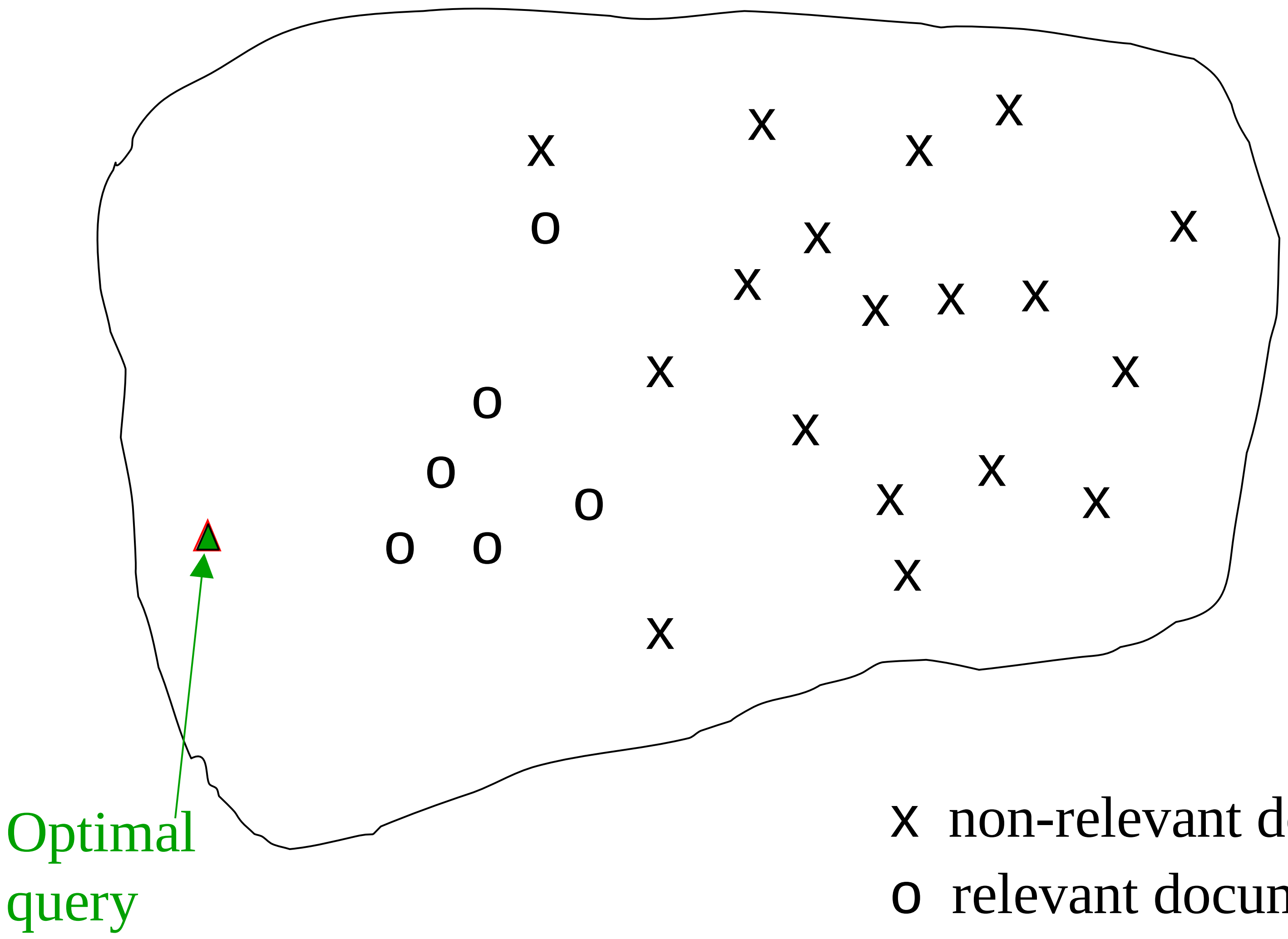
$$\vec{q}_{opt} = \arg\max_{\vec{q}} [\cos(\vec{q}, \vec{\mu}(C_r)) - \cos(\vec{q}, \vec{\mu}(C_{nr}))]$$

- Tries to separate docs marked relevant and non-relevant

$$\vec{q}_{opt} = \frac{1}{|C_r|} \sum_{\vec{d}_j \in C_r} \vec{d}_j - \frac{1}{|C_{nr}|} \sum_{\vec{d}_j \notin C_r} \vec{d}_j$$

- Problem: we don't know the truly relevant docs

The Theoretically Best Query



x non-relevant documents

o relevant documents

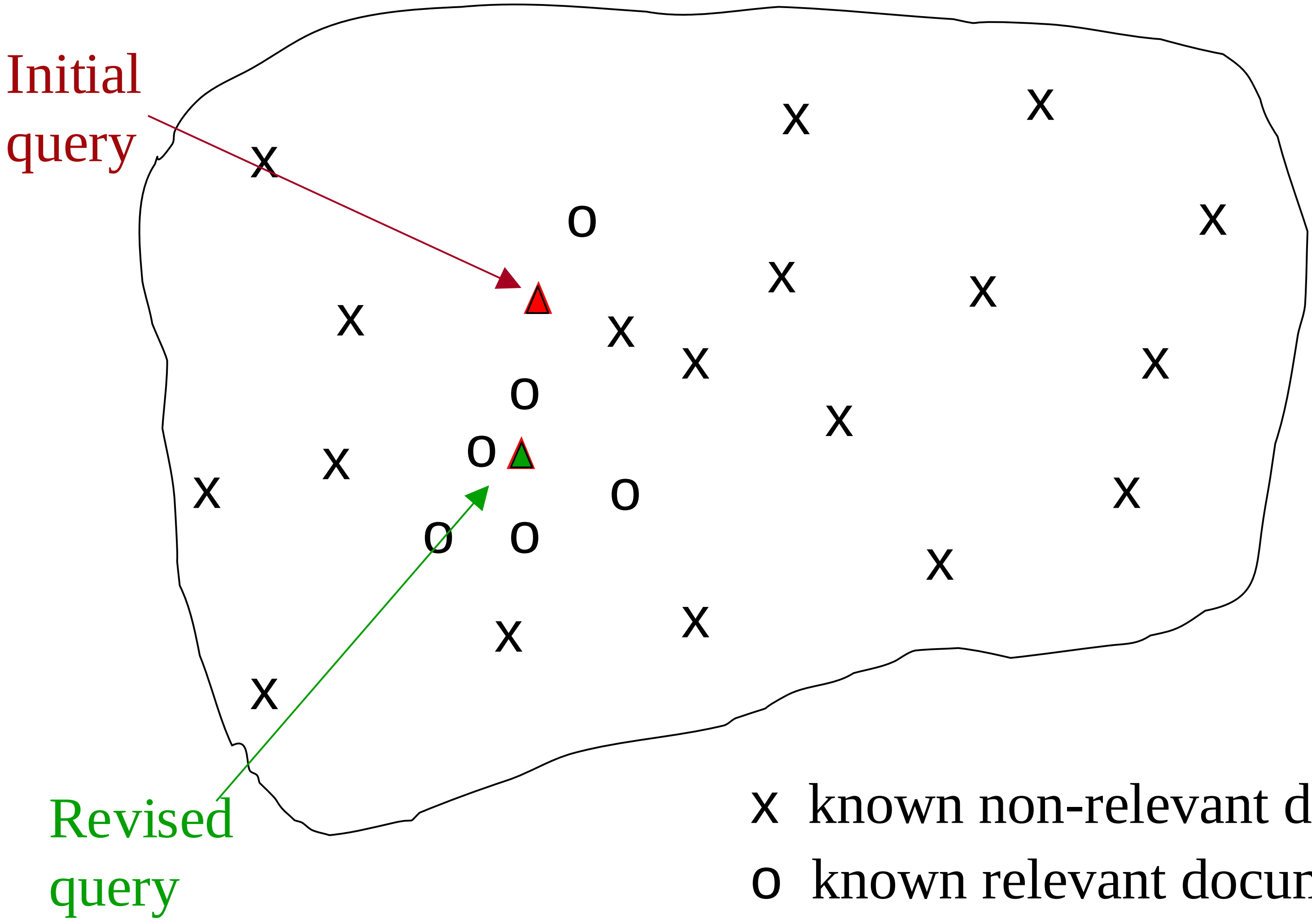
Rocchio 1971 Algorithm (SMART)

- Used in practice:

$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j$$

- *D_r* = set of known relevant doc vectors
- *D_{nr}* = set of known irrelevant doc vectors
  - Different from C_r and C_{nr} ⚠
- *q_m* = modified query vector; *q_0* = original query vector; *α,β,γ*: weights (hand-chosen or set empirically)
- New query moves toward relevant documents and away from irrelevant documents

Relevance Feedback on Initial Query



x known non-relevant documents

o known relevant documents

Relevance Feedback: Problems

- Long queries are inefficient for typical IR engine.
  - Long response times for user.
  - High cost for retrieval system.
  - Partial solution:
    - Only reweight certain prominent terms
      - Perhaps top 20 by term frequency
- Users are often reluctant to provide explicit feedback
- It’s often harder to understand why a particular document was retrieved after applying relevance feedback

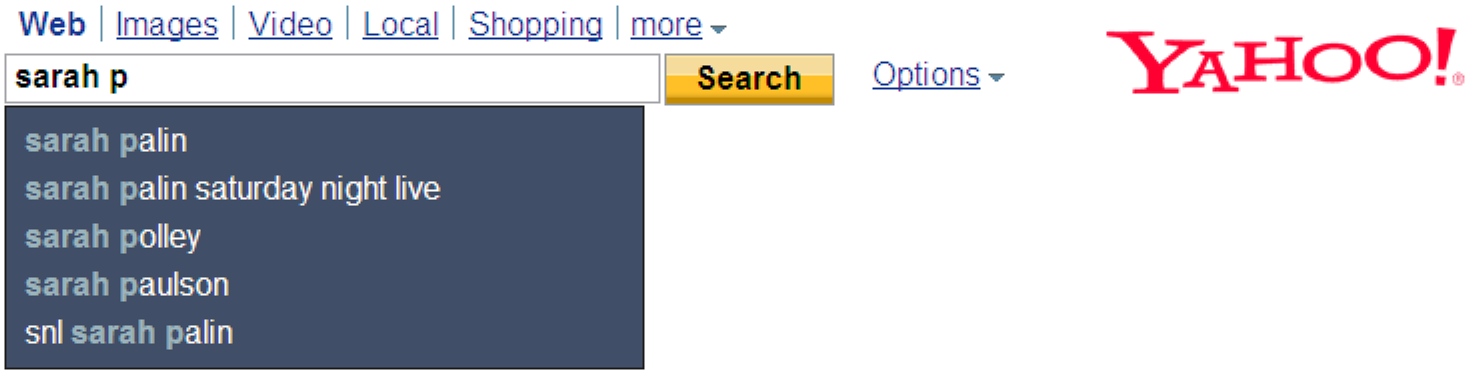
Pseudo Relevance Feedback

- Pseudo-relevance feedback automates the "manual" part of true relevance feedback.
- Pseudo-relevance algorithm:
  - Retrieve a ranked list of hits for the user's query
  - Assume that the top k documents are relevant.
  - Do relevance feedback (e.g., Rocchio)
- Works very well on average
- But can go horribly wrong for some queries.
- Several iterations can cause query drift.

Query Expansion

- In relevance feedback, users give additional input (relevant/non-relevant) on documents, which is used to reweight terms in the documents
- In query expansion, users give additional input (good/bad search term) on words or phrases

Query Assist



sarah palin
sarah palin saturday night live
sarah polley
sarah paulson
snl sarah palin

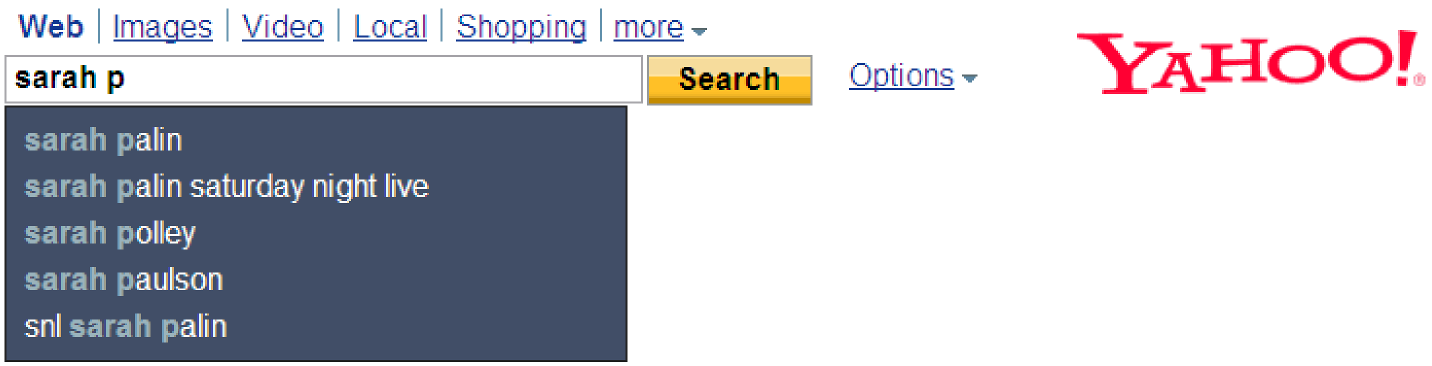
Would you expect such a feature to increase the query volume at a search engine?

How do we augment the user query?

- Manual thesaurus
  - E.g. MedLine: physician, syn: doc, doctor, MD, medico
  - Can be query rather than just synonyms
- Global Analysis: (static; of all documents in collection)
  - Automatically derived thesaurus
    - (co-occurrence statistics)
  - Refinements based on query log mining
    - Common on the web
- Local Analysis: (dynamic)
  - Analysis of documents in result set

Query assist

- Generally done by query log mining
- Recommend frequent recent queries that contain partial string typed by user
- A ranking problem! View each prior query as a doc – Rank-order those matching partial string …

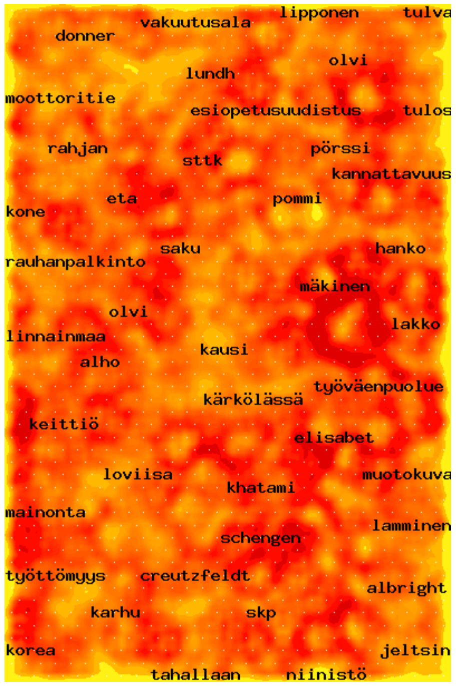


sarah p

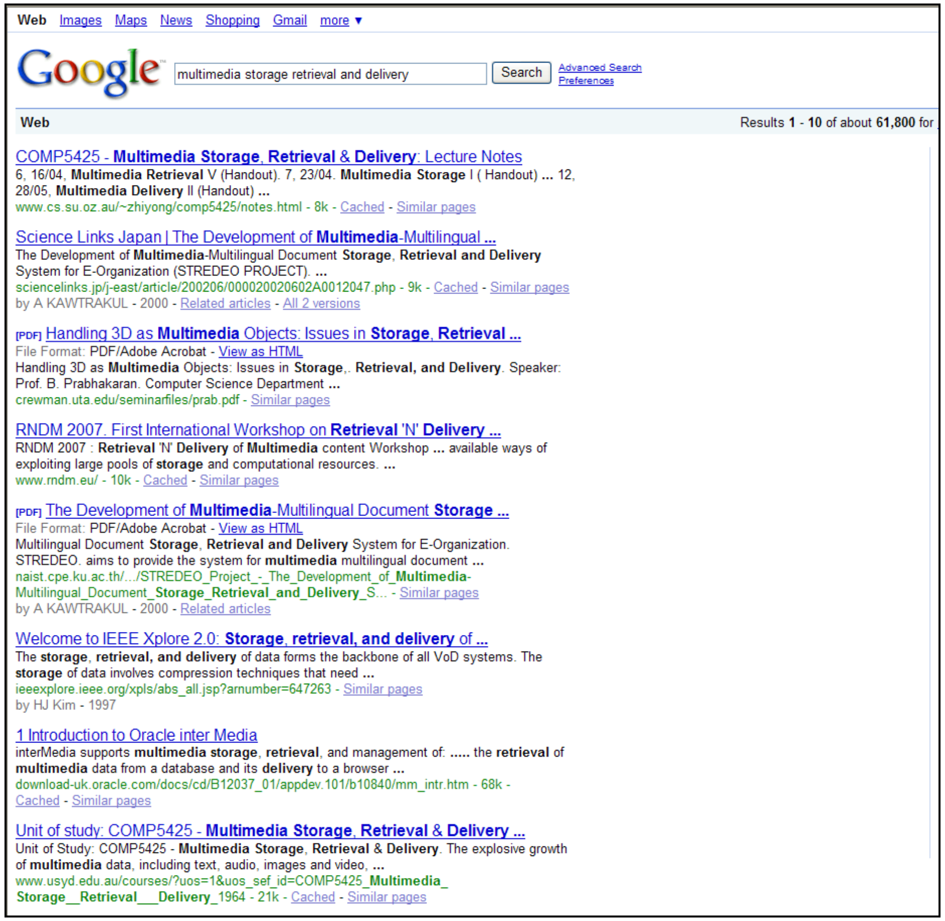
sarah palin
sarah palin saturday night live
sarah polley
sarah paulson
snl sarah palin

Results Presentation

- List
  - Title and Summary
- Visual



WEBSOM



Web Images Maps News Shopping Gmail more ▾

Google [multimedia storage retrieval and delivery] Search Advanced Search Preferences

Web Results **1** - **10** of about **61,800** for

COMP5425 - **Multimedia Storage**, **Retrieval** & **Delivery**: Lecture Notes
6, 16/04, **Multimedia Retrieval** V (Handout). 7, 23/04. **Multimedia Storage** I (Handout) ... 12, 28/05, **Multimedia Delivery** II (Handout) ...
www.cs.su.oz.au/~zhiyong/comp5425/notes.html - 8k - Cached - Similar pages

Science Links Japan | The Development of **Multimedia**-Multilingual ...
The Development of **Multimedia**-Multilingual Document **Storage**, **Retrieval and Delivery** System for E-Organization (STREDEO PROJECT). ...
sciencelinks.jp/j-east/article/200206/000020020602A0012047.php - 9k - Cached - Similar pages
by A KAWTRAKUL - 2000 - Related articles - All 2 versions

[PDF] Handling 3D as **Multimedia** Objects: Issues in **Storage**, **Retrieval** ...
File Format: PDF/Adobe Acrobat - View as HTML
Handling 3D as **Multimedia** Objects: Issues in **Storage**,. **Retrieval, and Delivery**. Speaker: Prof. B. Prabhakaran. Computer Science Department ...
crewman.uta.edu/seminarfiles/prab.pdf - Similar pages

RNDM 2007. First International Workshop on **Retrieval** 'N' **Delivery** ...
RNDM 2007 : **Retrieval** 'N' **Delivery** of **Multimedia** content Workshop ... available ways of exploiting large pools of **storage** and computational resources. ...
www.rndm.eu/ - 10k - Cached - Similar pages

[PDF] The Development of **Multimedia**-Multilingual Document **Storage** ...
File Format: PDF/Adobe Acrobat - View as HTML
Multilingual Document **Storage**, **Retrieval and Delivery** System for E-Organization. STREDEO. aims to provide the system for **multimedia** multilingual document ...
naist.cpe.ku.ac.th/.../STREDEO_Project_-_The_Development_of_**Multimedia**-Multilingual_Document_**Storage**_**Retrieval**_and_**Delivery**_S... - Similar pages
by A KAWTRAKUL - 2000 - Related articles

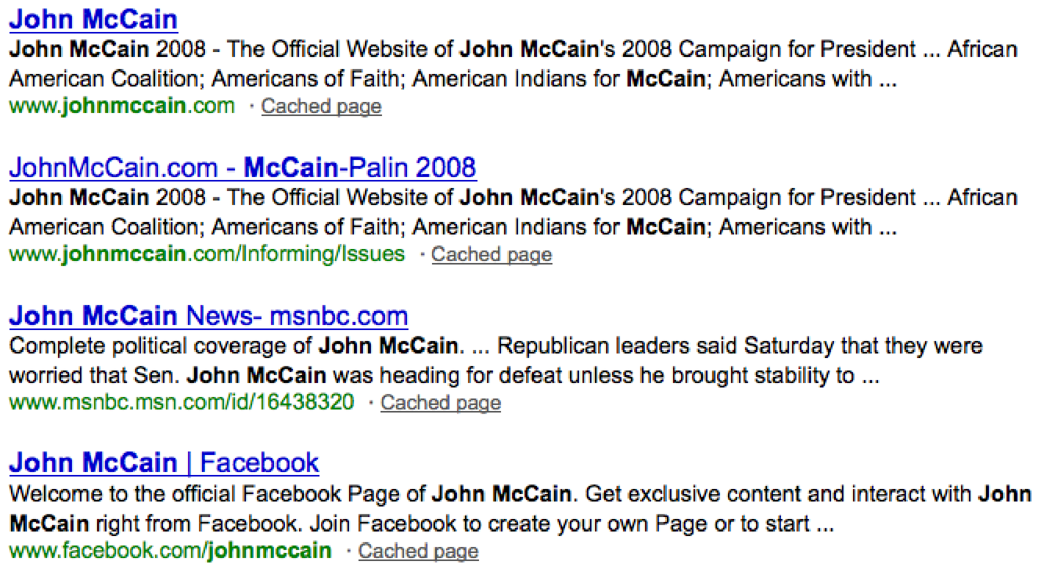
Welcome to IEEE Xplore 2.0: **Storage**, **retrieval, and delivery** of ...
The **storage**, **retrieval, and delivery** of data forms the backbone of all VoD systems. The **storage** of data involves compression techniques that need ...
ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=647263 - Similar pages
by HJ Kim - 1997

1 Introduction to Oracle inter Media
interMedia supports **multimedia storage**, **retrieval**, and management of: ..... the **retrieval** of **multimedia** data from a database and its **delivery** to a browser ...
download-uk.oracle.com/docs/cd/B12037_01/appdev.101/b10840/mm_intr.htm - 68k - Cached - Similar pages

Unit of study: COMP5425 - **Multimedia Storage**, **Retrieval** & **Delivery** ...
Unit of Study: COMP5425 - **Multimedia Storage**, **Retrieval** & **Delivery**. The explosive growth of **multimedia** data, including text, audio, images and video, ...
www.usyd.edu.au/courses/?uos=1&uos_sef_id=COMP5425_**Multimedia**_**Storage**__**Retrieval**___**Delivery**_1964 - 21k - Cached - Similar pages

Result Summaries

- Having ranked the documents matching a query, we wish to present a results list
- Most commonly, a list of the document titles plus a short summary, aka “10 blue links”


John McCain
John McCain 2008 - The Official Website of **John McCain**'s 2008 Campaign for President ... African American Coalition; Americans of Faith; American Indians for **McCain**; Americans with ...
www.**johnmccain**.com · Cached page

JohnMcCain.com - **McCain**-Palin 2008
John McCain 2008 - The Official Website of **John McCain**'s 2008 Campaign for President ... African American Coalition; Americans of Faith; American Indians for **McCain**; Americans with ...
www.**johnmccain**.com/Informing/Issues · Cached page

John McCain News- msnbc.com
Complete political coverage of **John McCain**. ... Republican leaders said Saturday that they were worried that Sen. **John McCain** was heading for defeat unless he brought stability to ...
www.msnbc.msn.com/id/16438320 · Cached page

John McCain | Facebook
Welcome to the official Facebook Page of **John McCain**. Get exclusive content and interact with **John McCain** right from Facebook. Join Facebook to create your own Page or to start ...
www.facebook.com/**johnmccain** · Cached page

Summaries

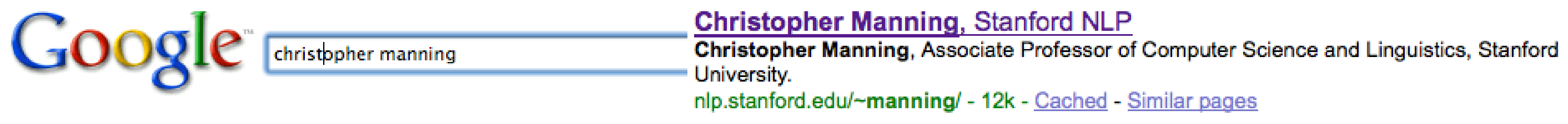
- The title is typically automatically extracted from document metadata. What about the summaries?
  - This description is crucial.
  - User can identify good/relevant hits based on description.
- Two basic kinds:
  - Static
  - Dynamic
- A **static summary** of a document is always the same, regardless of the query that hit the doc
- A **dynamic summary** is a *query-dependent* attempt to explain why the document was retrieved for the query at hand

Static summaries

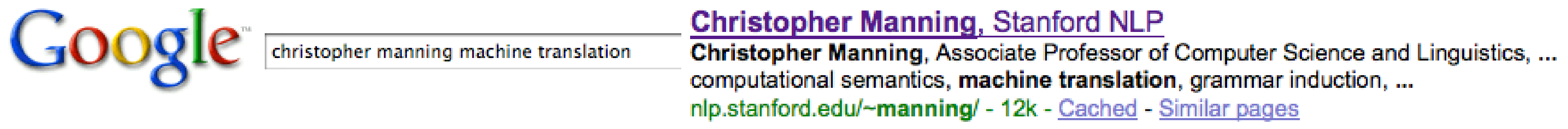
- In typical systems, the static summary is a subset of the document
- Simplest heuristic: the first 50 (or so – this can be varied) words of the document
  - Summary cached at indexing time
- More sophisticated: extract from each document a set of “key” sentences
  - Simple NLP heuristics to score each sentence
  - Summary is made up of top-scoring sentences.
- Most sophisticated: NLP used to synthesize a summary
  - Seldom used in IR; cf. text summarization work

Dynamic summaries

- Present one or more "windows" within the document that contain several of the query terms
  - "KWIC" snippets: Keyword in Context presentation
- Generated in conjunction with scoring
  - If query found as a phrase, all or some occurrences of the phrase in the doc
  - If not, document windows that contain multiple query terms
- The summary itself gives the entire content of the window – all terms, not only the query terms – how?



Christopher Manning, Stanford NLP
Christopher Manning, Associate Professor of Computer Science and Linguistics, Stanford University.
nlp.stanford.edu/~**manning**/ - 12k - Cached - Similar pages



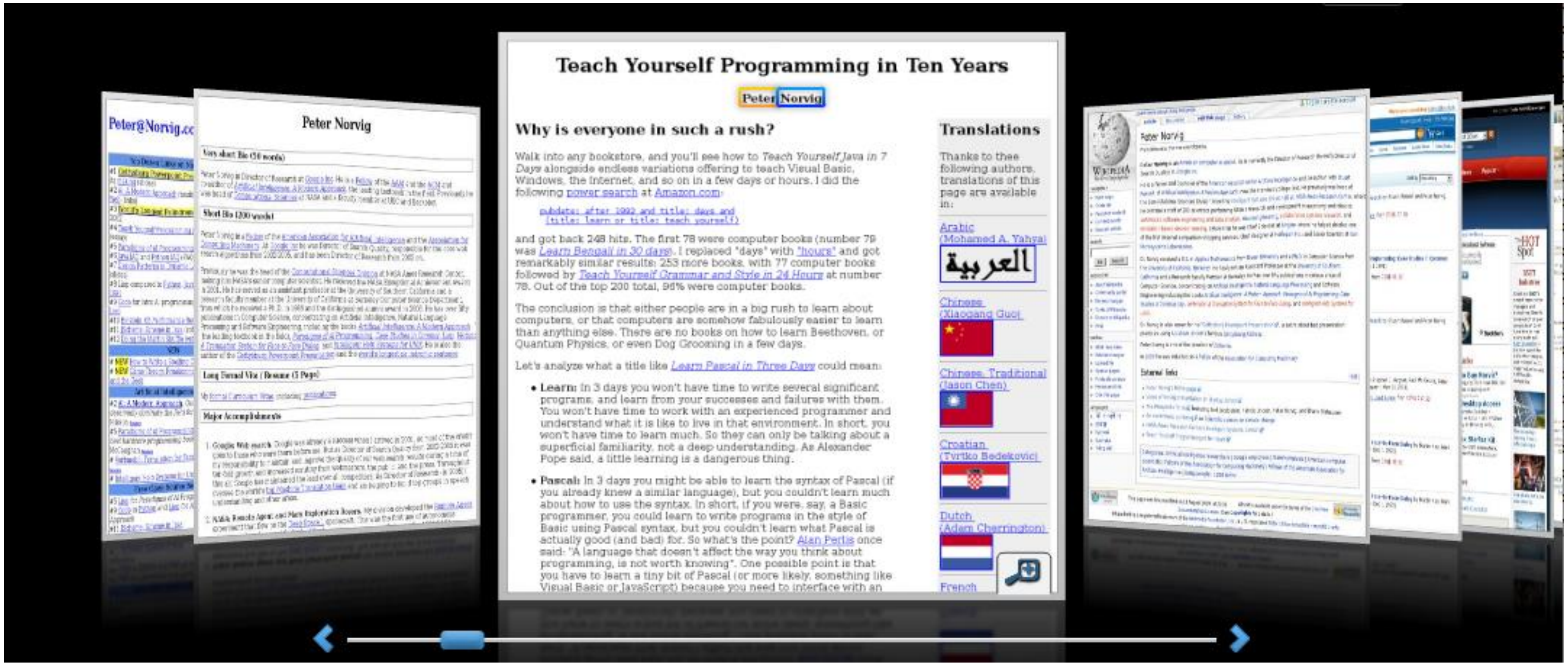
Christopher Manning, Stanford NLP
Christopher Manning, Associate Professor of Computer Science and Linguistics, ...
computational semantics, **machine translation**, grammar induction, ...
nlp.stanford.edu/~**manning**/ - 12k - Cached - Similar pages

Dynamic Summaries

- Producing good dynamic summaries is a tricky optimization problem
  - The real estate for the summary is normally small and fixed
  - Want short item, so show as many KWIC matches as possible, and perhaps other things like title
  - Want snippets to be long enough to be useful
  - Want linguistically well-formed snippets: users prefer snippets that contain complete phrases
  - Want snippets maximally informative about doc
- But users really like snippets, even if they complicate IR system design

Alternative results presentations?

- An active area of HCI research
- An alternative: http://www.searchme.com / copies the idea of Apple’s Cover Flow for search results



Teach Yourself Programming in Ten Years

Peter Norvig

Why is everyone in such a rush?

Walk into any bookstore, and you'll see how to *Teach Yourself Java in 7 Days* alongside endless variations offering to teach Visual Basic, Windows, the Internet, and so on in a few days or hours. I did the following power search at Amazon.com:

pubdate: after 1992 and title: days and (title: learn or title: teach yourself)

and got back 248 hits. The first 78 were computer books (number 79 was *Learn Bengali in 30 days*). I replaced "days" with "hours" and got remarkably similar results: 253 more books, with 77 computer books followed by *Teach Yourself Grammar and Style in 24 Hours* at number 78. Out of the top 200 total, 96% were computer books.

The conclusion is that either people are in a big rush to learn about computers, or that computers are somehow fabulously easier to learn than anything else. There are no books on how to learn Beethoven, or Quantum Physics, or even Dog Grooming in a few days.

Let's analyze what a title like *Learn Pascal in Three Days* could mean:

- **Learn:** In 3 days you won't have time to write several significant programs, and learn from your successes and failures with them. You won't have time to work with an experienced programmer and understand what it is like to live in that environment. In short, you won't have time to learn much. So they can only be talking about a superficial familiarity, not a deep understanding. As Alexander Pope said, a little learning is a dangerous thing.
- **Pascal:** In 3 days you might be able to learn the syntax of Pascal (if you already knew a similar language), but you couldn't learn much about how to use the syntax. In short, if you were, say, a Basic programmer, you could learn to write programs in the style of Basic using Pascal syntax, but you couldn't learn what Pascal is actually good (and bad) for. So what's the point? Alan Perlis once said: "A language that doesn't affect the way you think about programming, is not worth knowing". One possible point is that you have to learn a tiny bit of Pascal (or more likely, something like Visual Basic or JavaScript) because you need to interface with an

Translations

Thanks to the following authors, translations of this page are available in:

Arabic (Mohamed A. Yahya)

العربية

Chinese (Xiaogang Guo)

Chinese, Traditional (Jason Chen)

Croatian (Tvrtko Bedekovic)

Dutch (Adam Cherrington)

French

Evaluation: Measures for IR

- How fast does it index
  - Number of documents/hour
  - (Average document size)
- How fast does it search
  - Latency as a function of index size
- Expressiveness of query language
  - Ability to express complex information needs
  - Speed on complex queries
- Uncluttered UI
- Is it free?

Evaluating an IR system

- Note: the **information need** is translated into a **query**
- Relevance is assessed relative to the **information need** *not* the **query**
- E.g., Information need: *I'm looking for information on whether drinking red wine is more effective at reducing your risk of heart attacks than white wine.*
- Query: ***wine red white heart attack effective***
- You evaluate whether the doc addresses the information need, not whether it has these words

Standard relevance benchmarks

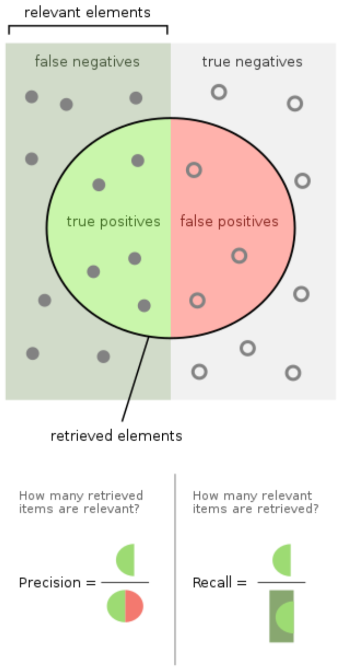
- TREC - National Institute of Standards and Technology (NIST) has run a large IR test bed for many years
- Reuters and other benchmark doc collections used
- "Retrieval tasks" specified
  - sometimes as queries
- Human experts mark, for each query and for each doc, <u>Relevant</u> or <u>Nonrelevant</u>
  - or at least for subset of docs that some system returned for that query

Unranked retrieval evaluation: Precision and Recall

- **Precision**: fraction of retrieved docs that are relevant = P(relevant|retrieved)
- **Recall**: fraction of relevant docs that are retrieved = P(retrieved|relevant)

	Relevant	Nonrelevant
Retrieved	tp	fp
Not Retrieved	fn	tn

- Precision P = tp/(tp + fp)
- Recall R = tp/(tp + fn)



Precision/Recall

- You can get high recall (but low precision) by retrieving all docs for all queries!
- Recall is a non-decreasing function of the number of docs retrieved

- In a good system, precision decreases as either the number of docs retrieved or recall increases
  - This is not a theorem, but a result with strong empirical confirmation

Difficulties in using precision/recall

- Should average over large document collection/query ensembles
- Need human relevance assessments
  - People aren't reliable assessors
- Assessments have to be binary
  - Nuanced assessments?
- Heavily skewed by collection/authorship
  - Results may not translate from one domain to another

A combined measure: *F*

- Combined measure that assesses precision/recall tradeoff is **F measure** (weighted harmonic mean):

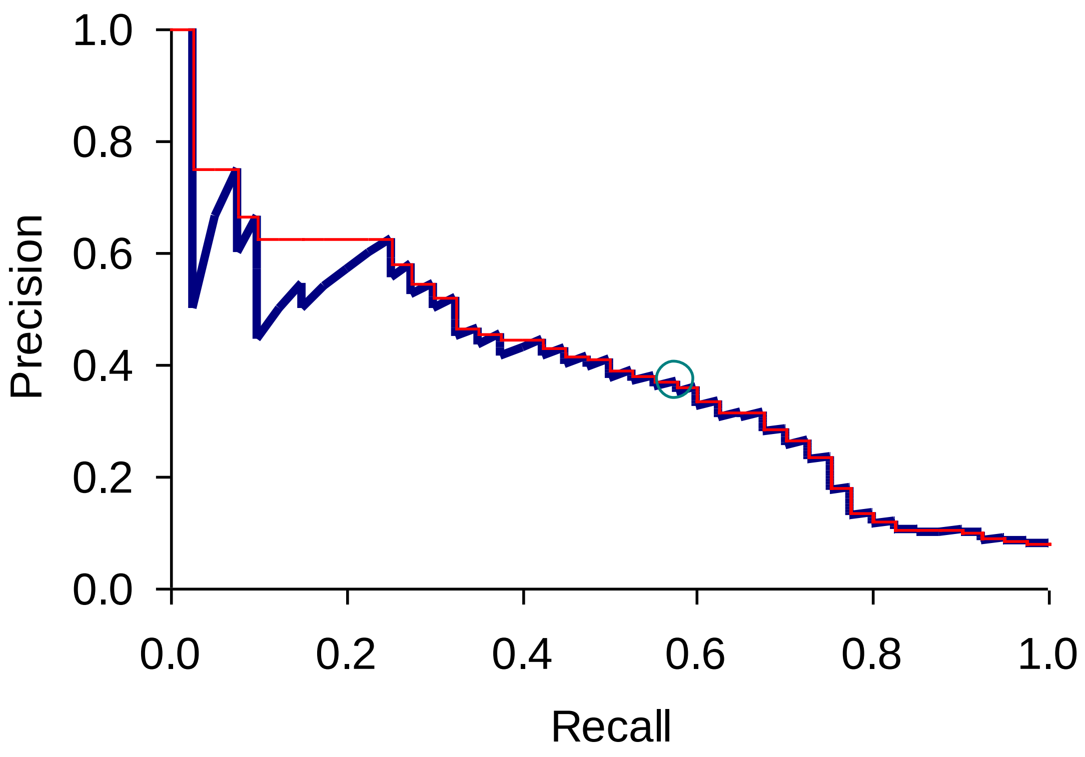
$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

- People usually use balanced F_1 measure
  - i.e., with $\beta = 1$ or $\alpha = ½$
- Harmonic mean is a conservative average
  - See CJ van Rijsbergen, *Information Retrieval*

Evaluating ranked results

- Evaluation of ranked results:
  - The system can return any number of results
  - By taking various numbers of the top returned documents (levels of recall), the evaluator can produce a *precision-recall curve*

A precision-recall curve

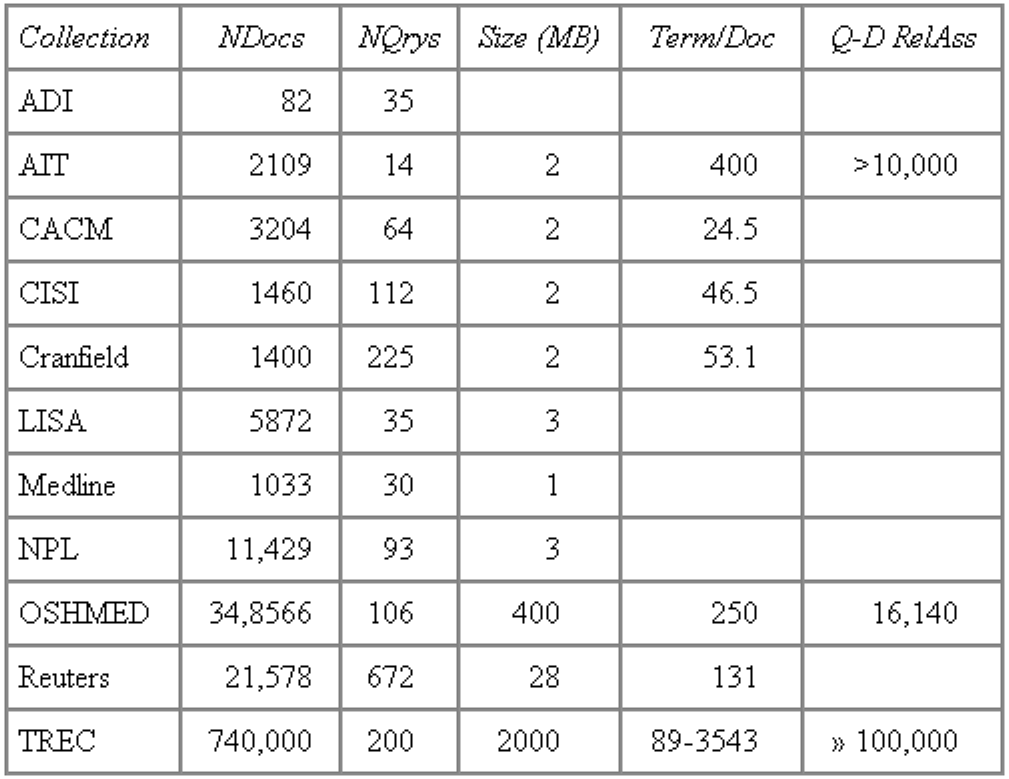


Averaging over queries

- A precision-recall graph for one query isn't a very sensible thing to look at
- You need to average performance over a whole bunch of queries.
- But there's a technical issue:
  - Precision-recall calculations place some points on the graph
  - How do you determine a value (interpolate) between the points?

Test Collections

TABLE 4.3 Common Test Corpora



Collection	*NDocs*	*NQrys*	*Size (MB)*	*Term/Doc*	*Q-D RelAss*
ADI	82	35			
AIT	2109	14	2	400	>10,000
CACM	3204	64	2	24.5	
CISI	1460	112	2	46.5	
Cranfield	1400	225	2	53.1	
LISA	5872	35	3		
Medline	1033	30	1		
NPL	11,429	93	3		
OSHMED	34,8566	106	400	250	16,140
Reuters	21,578	672	28	131	
TREC	740,000	200	2000	89-3543	» 100,000

TREC

- TREC Ad Hoc task from first 8 TRECs is standard IR task
  - 50 detailed information needs a year
  - Human evaluation of pooled results returned
  - More recently other related things: Web track, HARD
- A TREC query (TREC 5)

```
<top>
<num> Number:  225
<desc> Description:
What is the main function of the Federal Emergency Management Agency
    (FEMA) and the funding level provided to meet emergencies?  Also,
    what resources are available to FEMA such as people, equipment,
    facilities?
</top>
```

Q&A FOR ASSIGNMENT

Need To Know

- Overall of retreival system
- (Inverted) Indexing
- Boolean query
- Vector space query
- Similarity measurement
- User interaction through relevance feedback
- Evaluation

References

- Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze, Introduction to information retrieval, Cambridge University Press, 2008. [Library]
  - http://www-csli.stanford.edu/~hinrich/information-retrieval-book.html
- Ricardo Baeza-Yates, Berthier Ribeiro-Neto, Modern Information Retrieval, Addison Wesley, 1999. [Library]
  - http://people.ischool.berkeley.edu/~hearst/irbook/
- I. Witten, A. Moffat, T. Bell, Managing Gigabytes (2nd edition), Morgan Kaufmann, 1999. [Library]
- C. J. Van Rijsbergen, Information Retrieval, 1979.
  - Access online http://www.dcs.gla.ac.uk/Keith/Preface.html
- Wolfgang Hürst, Web Search, Summer Term 2006, Albert-Ludwigs-University