Information Retrieval

M. Narasimha Murty
Professor, Dept. of CSA
Indian Institute of Science
Bangalore-560 012
mnm@csa.iisc.ernet.in

September 14, 2015

Ranked Retrieval

- Boolean Retrieval is Good for expert users with precise understanding of their needs and of the collections
- Not good for the majority of users
- Most users are not capable of writing Boolean queries . . .
- . . . or they are, but they think it’s too much work
- Most users don’t want to wade through 1000s of results
- This is particularly true of web search

Problem with Boolean Search: Feast or Famine

- Boolean queries often result in either too few (=0) or too many (1000s) results.
- Query 1 (boolean conjunction): [standard user dlink 650]
    - → 200,000 hits – feast
- Query 2 (boolean conjunction): [standard user dlink 650 no card found]
    - → 0 hits – famine
- In Boolean retrieval, it takes a lot of skill to come up with a query that produces a manageable number of hits.

Feast or Famine: No Problem in Ranked Retrieval

- With ranking, large result sets are not an issue
- Just show the top 10 results
- Doesn’t overwhelm the user
- Premise: the ranking algorithm works: More relevant results are ranked higher than less relevant results

Scoring as the Basis of Ranked Retrieval

- We wish to rank documents that are more relevant higher than documents that are less relevant.
- How can we accomplish such a ranking of the documents in the collection with respect to a query?
- Assign a score to each query-document pair, say in [0, 1].
- This score measures how well document and query "match".

Query-Document Matching Scores

- How do we compute the score of a query-document pair?
- Let’s start with a one-term query
- 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
- We will look at a number of alternatives for doing this

Jaccard Coefficient

- A commonly used measure of overlap of two sets
- Let *A* and *B* be two sets
- Jaccard coefficient:

$$\text{JACCARD}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$
$$(A \neq \emptyset \text{ or } B \neq \emptyset)$$

- JACCARD (*A*, *A*) = 1
- JACCARD (*A*, *B*) = 0 if *A* ∩ *B* = 0
- A and B don’t have to be the same size.
- Always assigns a number between 0 and 1.

What's Wrong with Jaccard?

- It doesn't consider term frequency (how many occurrences a term has).
- Rare terms are more informative than frequent terms. Jaccard does not consider this information.
- We need a more sophisticated way of normalizing for the length of a document.
- Later in this lecture, we'll use $|A \cap B| / \sqrt{|A \cup B|}$ (cosine) . . .
- . . . instead of $|A \cap B| / |A \cup B|$ (Jaccard) for length normalization.

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 because:
- A document with tf = 10 occurrences of the term is more relevant than a document with tf = 1 occurrence of the term
- But not 10 times more relevant
- Relevance does not increase proportionally with term frequency

Instead of Raw Frequency: Log Frequency

- The log frequency weight of term t in d is defined as follows

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

- tf*t,d* → w*t,d* :
  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*:
  tf-matching-score(*q*, *d*) = *t*∈*q*∩*d* (1 + log tf*t,d*)
- The score is 0 if none of the query terms is present in the document.

Frequency in Document vs in Collection

- In addition, to term frequency (the frequency of the term in the document) . . .
- . . .we also want to use the frequency of the term in the collection for weighting and ranking

Desired Weight for Rare Terms

- Rare terms are more informative than frequent terms.
- 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.
- → We want high weights for rare terms like ARACHNOCENTRIC.

Desired Weight for Frequent Terms

- Frequent terms are less informative than rare terms
- Consider a term in the query that is frequent in the collection (e.g., GOOD, INCREASE, LINE)
- A document containing this term is more likely to be relevant than a document that doesn't . . .
- . . . but words like GOOD, INCREASE and LINE are not sure indicators of relevance
- → For frequent terms like GOOD, INCREASE and LINE, we want positive weights . . .
- . . . but lower weights than for rare terms

Document Frequency

- We want high weights for rare terms like ARACHNOCENTRIC
- We want low (positive) weights for frequent words like GOOD, INCREASE and LINE
- We will use document frequency to factor this into computing the matching score
- The document frequency is the number of documents in the collection that the term occurs in

idf Weight

- df*t* is the document frequency, the number of documents that *t* occurs in.
- df*t* is an inverse measure of the informativeness of term *t*.
- We define the idf weight of term t as follows:

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

- (*N* is the number of documents in the collection.)
- idf*t* is a measure of the informativeness of the term.
- [log *N*/df*t*] instead of [*N*/df*t*] to "dampen" the effect of idf
- Note that we use the log transformation for both term frequency and document frequency.

Examples for idf

- Compute idf*t* using the formula: $\mathrm{idf}_t = \log_{10} \frac{1{,}000{,}000}{\mathrm{df}_t}$

term	df*t*	idf*t*
calpurnia	1	6
animal	100	4
sunday	1000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

Effect of idf on Ranking

- idf affects the ranking of documents for queries with at least two terms
- For example, in the query “arachnocentric line”, idf weighting increases the relative weight of ARACHNOCENTRIC and decreases the relative weight of LINE
- idf has little effect on ranking for one-term queries

Collection Frequency vs. Document Frequency

word	collection frequency	document frequency
INSURANCE	10440	3997
TRY	10422	8760

- Collection frequency of *t*: number of tokens of *t* in the collection
- Document frequency of *t*: number of documents *t* occurs in
- Why these numbers?
- Which word is a better search term (and should get a higher weight)?
- This example suggests that df (and idf) is better for weighting than cf (and "icf")

tf-idf Weighting

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

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$$

- tf-weight
- idf-weight
- 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

Summary: tf-idf

- Assign a tf-idf weight for each term t in each document *d*: $w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$

- The tf-idf weight . . .
- . . . increases with the number of occurrences within a document (term frequency)
- . . . increases with the rarity of the term in the collection (inverse document frequency)

Documents as Vectors

- Each document is now represented as a real-valued vector of tf-idf weights ∈ R|*V*|
- So we have a |*V*|-dimensional real-valued vector space
- Terms are axes of the space
- Documents are points or vectors in this space
- Very high-dimensional: tens of millions of dimensions when you apply this to web search engines
- Each vector is very sparse - most entries are zero

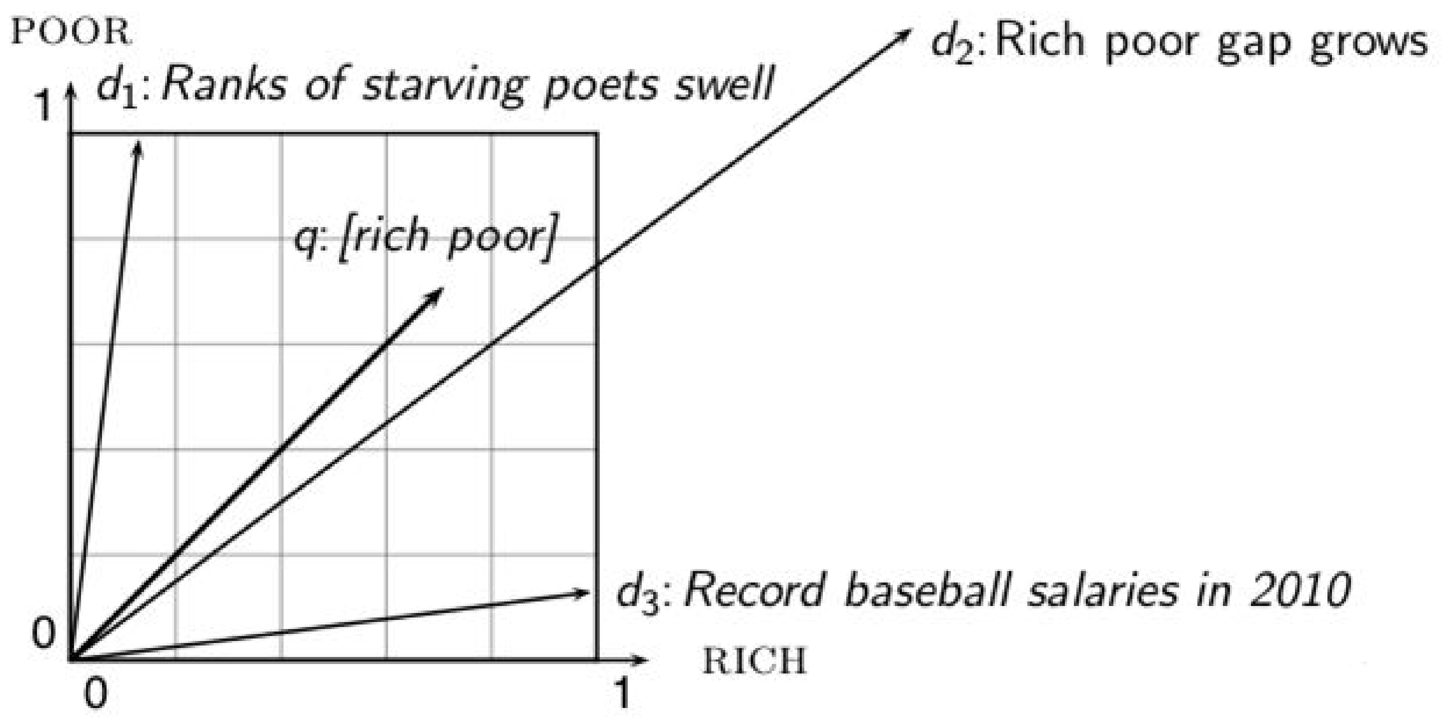
Queries as Vectors

- Key idea 1: do the same for queries: represent them as vectors in the high-dimensional space
- Key idea 2: Rank documents according to their proximity to the query
- proximity = similarity
- proximity ≈ negative distance
- Recall: We're doing this because we want to get away from the you're-either-in-or-out, feast-or-famine Boolean model
- Instead: rank relevant documents higher than nonrelevant documents

How do We Formalize Vector Space Similarity?

- First cut: (negative) 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


POOR
RICH
1
0
0
1
d_1: Ranks of starving poets swell
d_2: Rich poor gap grows
q: [rich poor]
d_3: Record baseball salaries in 2010

- The Euclidean distance of $\vec{q}$ and $\vec{d_2}$ is large although the distribution of terms in the query *q* and the distribution of terms in the document *d*2 are very similar
- Questions about basic vector space setup?

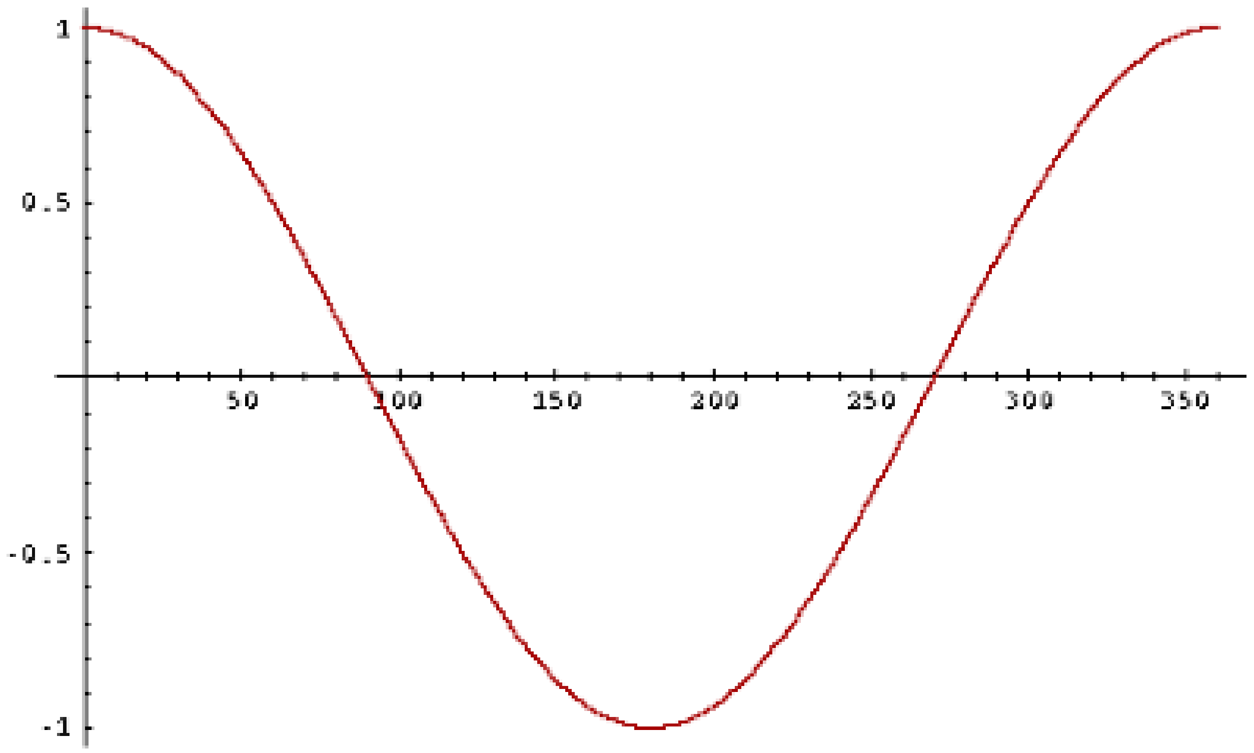
Use Angle Instead of Distance

- Rank documents according to angle with query
- Thought experiment: take a document d and append it to itself. Call this document *d′*. *d′* is twice as long as *d*.
- "Semantically" *d* and *d′* have the same content.
- The angle between the two documents is 0, corresponding to maximal similarity . . .
- . . . even though the Euclidean distance between the two documents can be quite large.

From Angles to Cosines

- The following two notions are equivalent
- Rank documents according to the angle between query and document in decreasing order
- Rank documents according to cosine(query,document) in increasing order
- Cosine is a monotonically decreasing function of the angle for the interval [0°, 180°]

Cosine



Length Normalization

- How do we compute the cosine?
- A vector can be (length-) normalized by dividing each of its components by its length – here we use the *L*2 norm:

$$||x||_2 = \sqrt{\sum_i x_i^2}$$

- This maps vectors onto the unit sphere . . .

. . . since after normalization: $||x||_2 = \sqrt{\sum_i x_i^2} = 1.0$

- As a result, longer documents and shorter documents have weights of the same order of magnitude
- Effect on the two documents *d* and *d′* (*d* appended to itself) from earlier slide: they have identical vectors after length-normalization

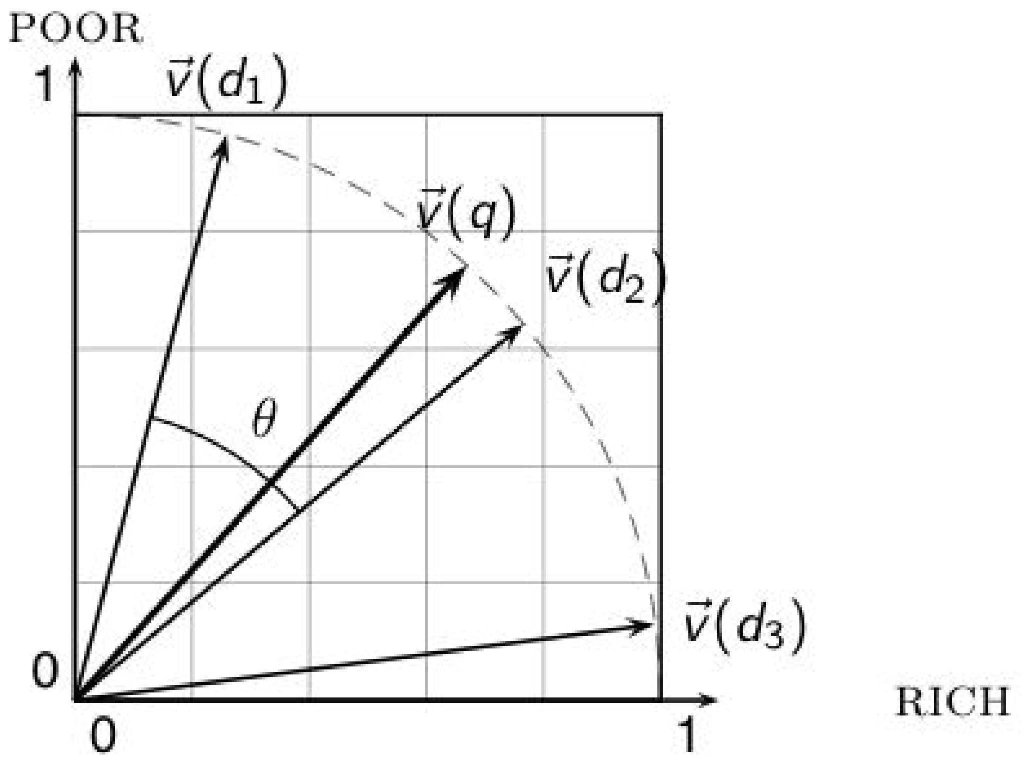
Cosine for Normalized Vectors

- For normalized vectors, the cosine is equivalent to the dot product or scalar product

$$\cos(\vec{q}, \vec{d}) = \vec{q} \cdot \vec{d} = \sum_i q_i \cdot d_i$$

- (i $\vec{q}$ an $\vec{d}$ are length-normalized)

Cosine Similarity Illustrated



Cosine: Example

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

term frequencies (counts)

term	SaS	PaP	WH
AFFECTION	115	58	20
JEALOUS	10	7	11
GOSSIP	2	0	6
WUTHERING	0	0	38

Cosine: Example

log frequency weighting

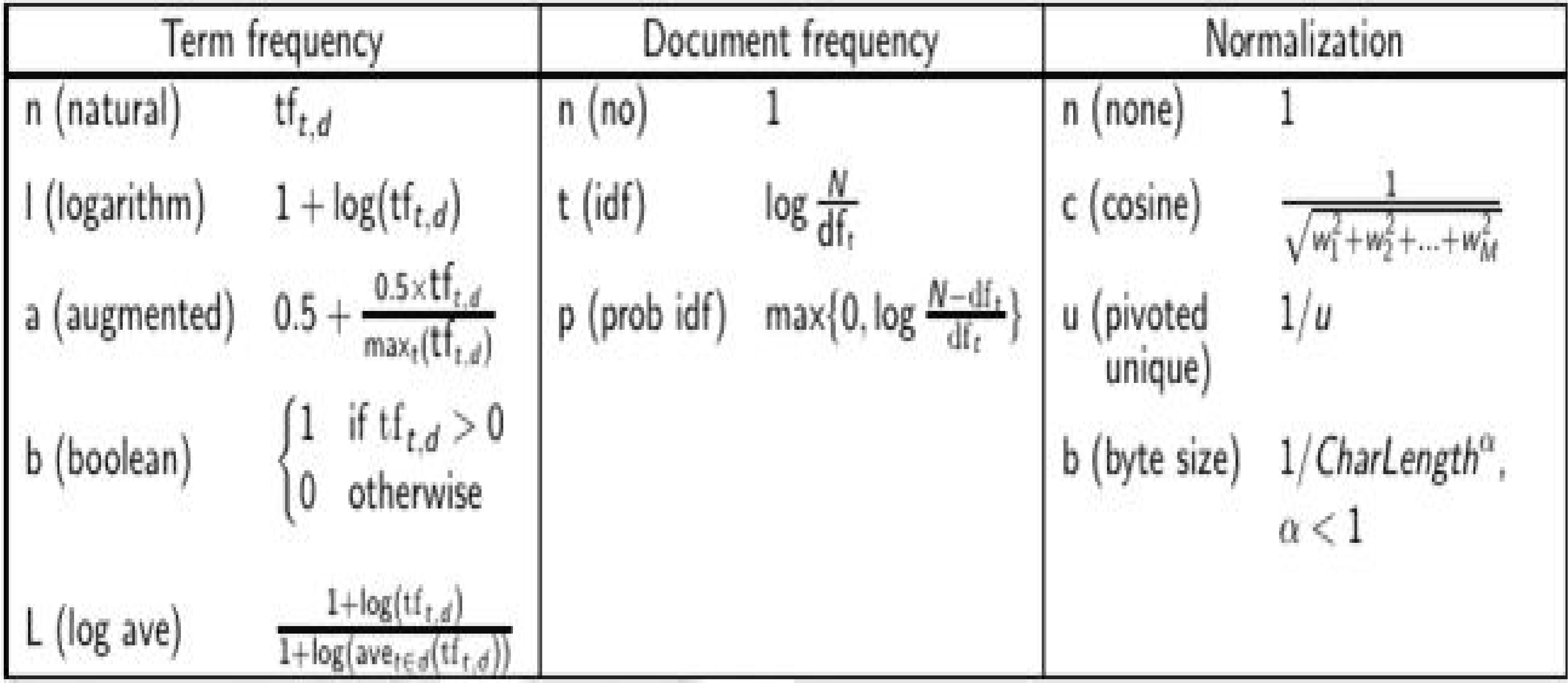
term	SaS	PaP	WH
AFFECTION	3.06	2.76	2.30
JEALOUS	2.0	1.85	2.04
GOSSIP	1.30	0	1.78
WUTHERING	0	0	2.58

log frequency weighting & cosine normalization

term	SaS	PaP	WH
AFFECTION	0.789	0.832	0.524
JEALOUS	0.515	0.555	0.465
GOSSIP	0.335	0.0	0.405
WUTHERING	0.0	0.0	0.588

- cos(SaS,PaP) $\approx 0.789 * 0.832 + 0.515 * 0.555 + 0.335 * 0.0 + 0.0 * 0.0 \approx 0.94$.
- cos(SaS,WH) ≈ 0.79
- cos(PaP,WH) ≈ 0.69
- Why do we have cos(SaS,PaP) > cos(SAS,WH)?

Components of tf-idf weighting

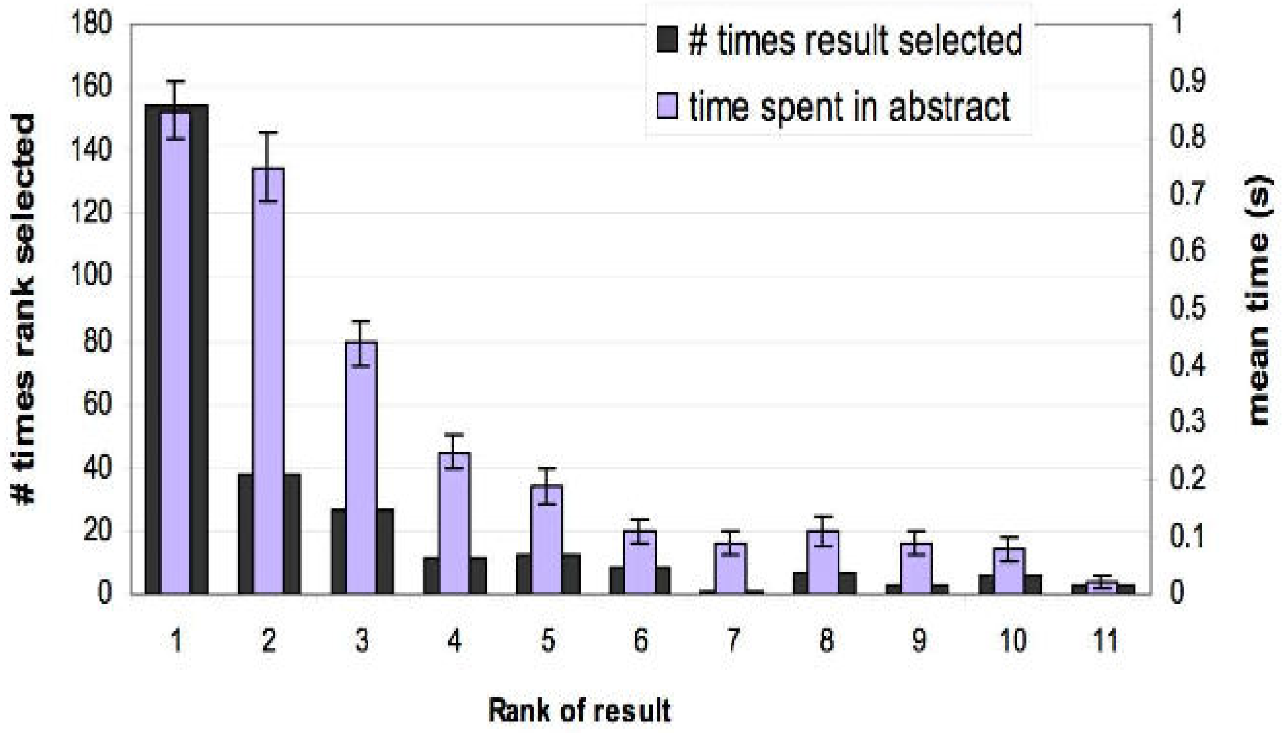


Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1+\log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5+\frac{0.5\times\mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	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)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$				

Ranked Retrieval in Vector Space Model

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

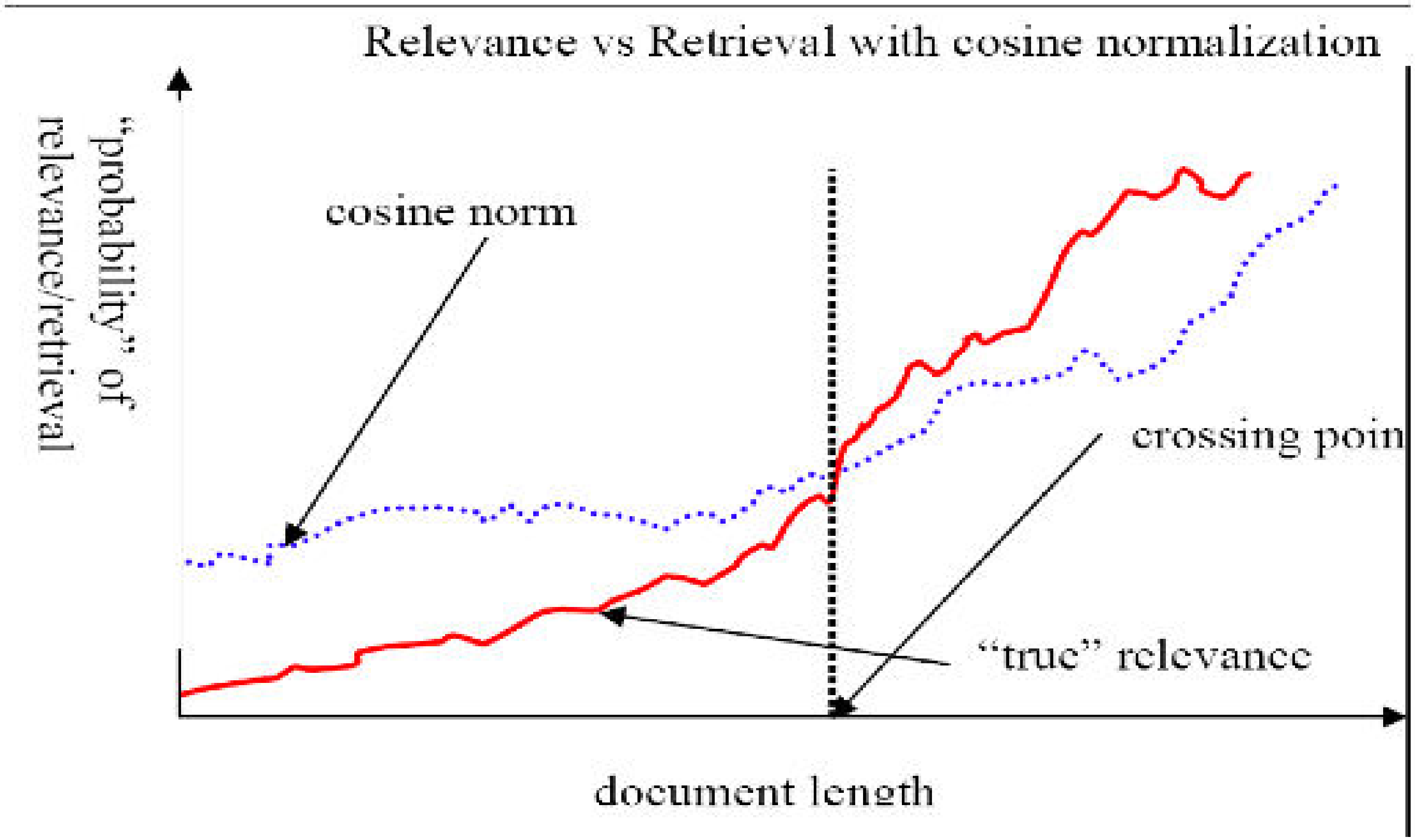
Looking Vs Clicking



- Users view results one and two more often / thoroughly
- Users click most frequently on result one

Google

Pivot Normalization



source:
Lilian Lee

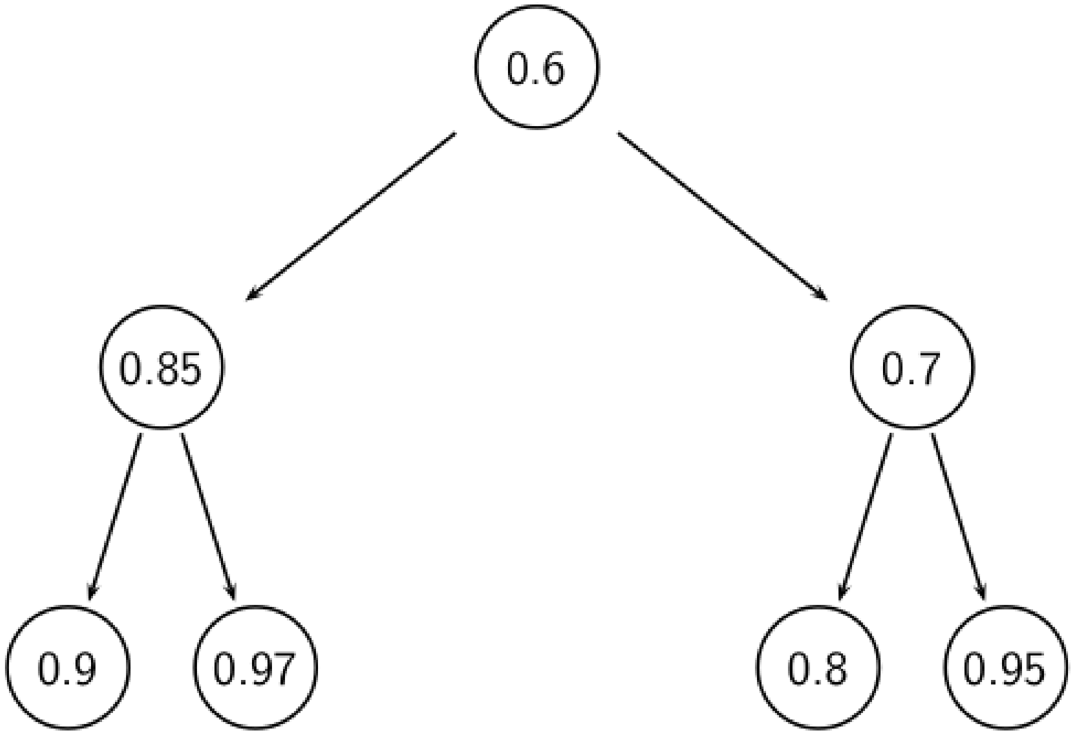
Heuristics for Finding the Top *k* Even Faster

- Document-at-a-time processing
- We complete computation of the query-document similarity score of document *di* before starting to compute the query-document similarity score of *di*+1.
- Requires a consistent ordering of documents in the postings lists
- Term-at-a-time processing
- We complete processing the postings list of query term *ti* before starting to process the postings list of *ti*+1
- Requires an accumulator for each document "still in the running"
- The most effective heuristics switch back and forth between term-at-a-time and document-at-a-time processing

Use Min Heap for Selecting Top *k* out of *N*

- Use a binary min heap
- A binary min heap is a binary tree in which each node's value is less than the values of its children
- It takes $O(N \log k)$ operations to construct the *k*-heap containing the *k* largest values (where *N* is the number of documents)
- Essentially linear in *N* for small k and large *N*

Binary Min Heap



Selecting *k* top scoring documents in *O*(*N* log *k*)

- Goal: Keep the *k* top documents seen so far
- Use a binary min heap
- To process a new document *d*′ with score *s*′:
- Get current minimum *hm* of heap (in *O*(1))
- If s′ ≤ *hm* skip to next document
- If s′ > *hm* heap-delete-root (in *O*(log *k*))
- Heap-add *d*′/*s*′ (in *O*(1))
- Reheapify (in *O*(log *k*))

Measures for a Search Engine

- How fast does it index?
  e.g., number of bytes per hour
- How fast does it search?
  e.g., latency as a function of queries per second
- What is the cost per query?
  in dollars

Measures for a Search Engine

- All of the preceding criteria are measurable: we can quantify speed / size / money
- However, the key measure for a search engine is user happiness
- What is user happiness?
  - Factors include:
    - Speed of response
    - Size of index
    - Uncluttered UI
    - Most important: relevance
    - (actually, maybe even more important: it’s free)
  - Note that none of these is sufficient: blindingly fast, but useless answers won’t make a user happy.
- How can we quantify user happiness?

Most common definition of user happiness: Relevance

- User happiness is equated with the relevance of search results to the query
- But how do you measure relevance?
- Standard methodology in information retrieval consists of three elements
  - A benchmark document collection
  - A benchmark suite of queries
  - An assessment of the relevance of each query-document pair

Relevance: Query vs. Information Need

- Relevance to what?
- First take: relevance to the query
- "Relevance to the query" is very problematic.
- Information need *i* : "I am looking for information on whether drinking red wine is more effective at reducing your risk of heart attacks than white wine."
- This is an information need, not a query.
- Query *q*: [red wine white wine heart attack]
- Consider document *d*′: *At heart of his speech was an attack on the wine industry lobby for downplaying the role of red and white wine in drunk driving*.
- *d*′ is an excellent match for query *q* . . .
- *d*′ is not relevant to the information need *i* .

Relevance: Query vs. Information need

- User happiness can only be measured by relevance to an information need, not by relevance to queries
- Our terminology is sloppy in these slides and in IIR: we talk about query-document relevance judgments even though we mean information-need-document relevance judgments

Precision and Recall

- Precision (*P*) is the fraction of retrieved documents that are relevant

$$\text{Precision} = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})} = P(\text{relevant}|\text{retrieved})$$

- Recall (*R*) is the fraction of relevant documents that are retrieved

$$\text{Recall} = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})} = P(\text{retrieved}|\text{relevant})$$

Precision and recall

	Relevant	Nonrelevant
Retrieved	true positives (TP)	false positives (FP)
Not retrieved	false negatives (FN)	true negatives (TN)

$P = TP \,/\, (\, TP + FP \,)$

$R = TP \,/\, (\, TP + FN \,)$

Precision/Recall Trade-off

- You can increase recall by returning more docs
- Recall is a non-decreasing function of the number of docs retrieved
- A system that returns all docs has 100% recall!
- The converse is also true (usually): It’s easy to get high precision for very low recall
- Suppose the document with the largest score is relevant. How can we maximize precision?

A combined measure: *F*

- *F* allows us to trade off precision against recall.

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

$\alpha \in [0, 1]$ and thus $\beta 2 \in [0,\infty]$

- Most frequently used: balanced *F* with $\beta = 1$ or $\alpha = 0.5$
- This is the harmonic mean of *P* and *R*: $\frac{1}{F} = \frac{1}{2}(\frac{1}{P} + \frac{1}{R})$
- What value range of *β* weights recall higher than precision?

F: Example

	relevant	**not relevant**	
retrieved	20	40	60
not retrieved	60	1,000,000	1,000,060
	80	1,000,040	1,000,120

- $P = 20/(20 + 40) = 1/3$
- $R = 20/(20 + 60) = 1/4$
- $$F_1 = 2\frac{1}{\frac{1}{\frac{1}{3}}+\frac{1}{\frac{1}{4}}} = 2/7$$

Accuracy

- Why do we use complex measures like precision, recall, and *F*?
- Why not something simple like accuracy?
- Accuracy is the fraction of decisions (relevant/nonrelevant) that are correct
- In terms of the contingency table above,
  - accuracy = $(TP + TN)/(TP + FP + FN + TN)$.
- Why is accuracy not a useful measure for web information retrieval?

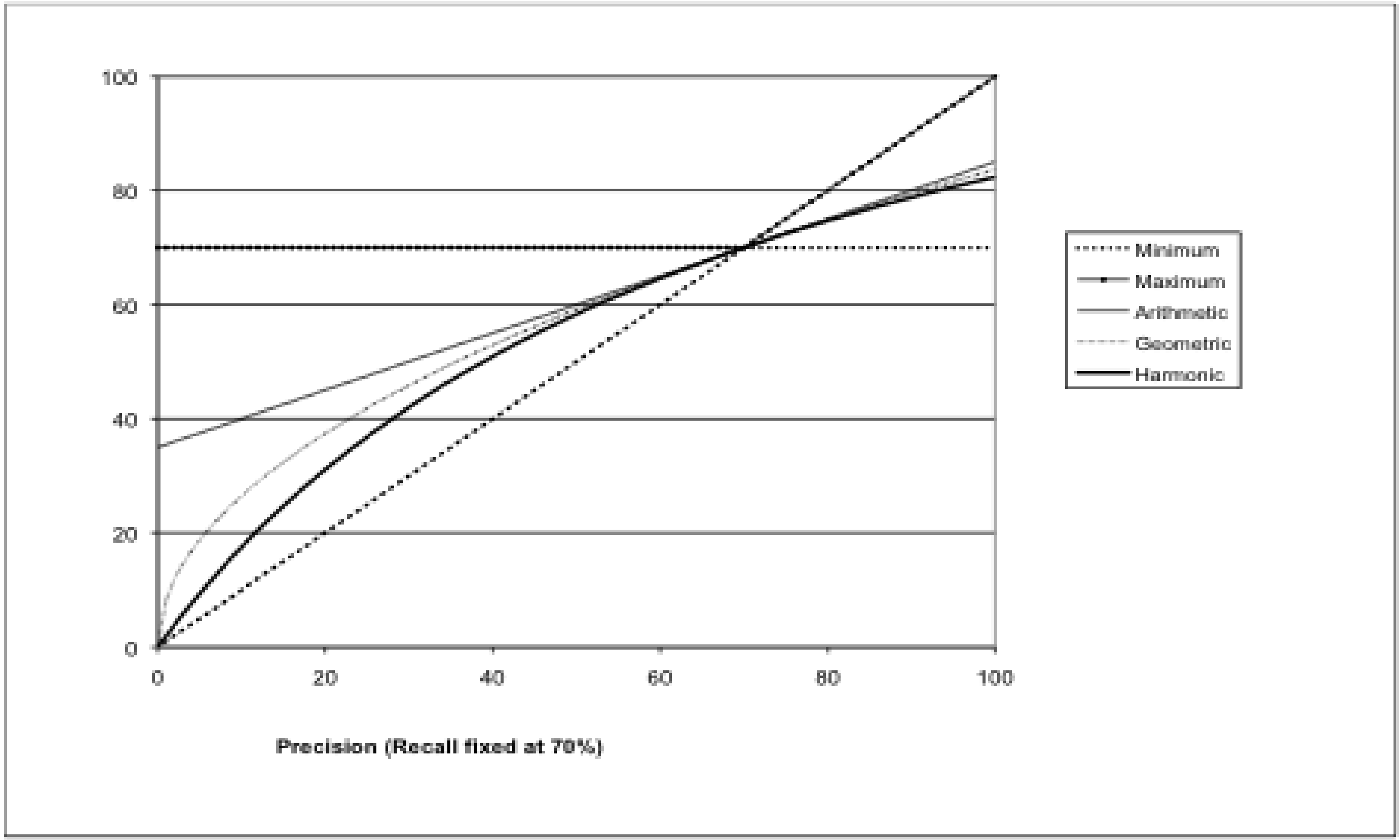
Why Accuracy is a Useless Measure in IR

- Simple trick to maximize accuracy in IR: always say no and return nothing
- You then get 99.99% accuracy on most queries.
- Searchers on the web (and in IR in general) want to find something and have a certain tolerance for junk.
- It's better to return some bad hits as long as you return something.
- →We use precision, recall, and *F* for evaluation, not accuracy.

F: Why Harmonic Mean?

- Why don’t we use a different mean of *P* and *R* as a measure?
  - e.g., the arithmetic mean
- The simple (arithmetic) mean is 50% for “return-everything” search engine, which is too high
- Desideratum: Punish really bad performance on either precision or recall
  - Taking the minimum achieves this
  - But minimum is not smooth and hard to weight
  - *F* (harmonic mean) is a kind of smooth minimum

*F*1 and Other Averages



- We can view the harmonic mean as a kind of soft minimum

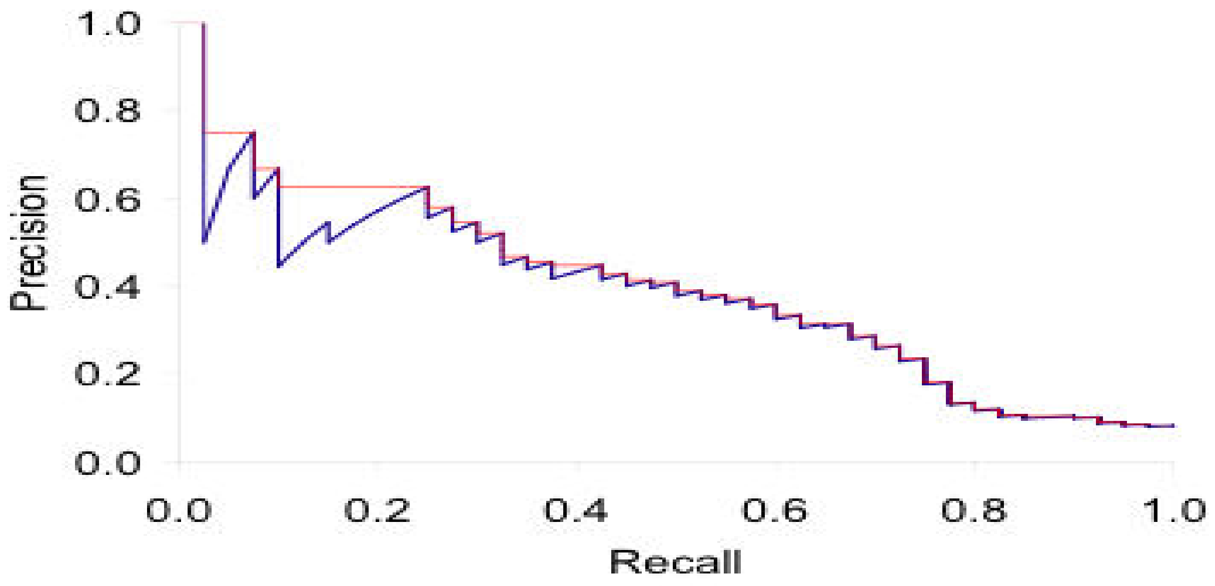
Difficulties in using Precision, Recall and *F*

- We need relevance judgments for information-need-document pairs – but they are expensive to produce
- For alternatives to using precision/recall and having to produce relevance judgments – see end of this lecture

Precision-Recall Curve

- Precision/recall/F are measures for unranked sets
- We can easily turn set measures into measures of ranked lists
- Just compute the set measure for each “prefix”: the top 1, top 2, top 3, top 4 etc results
- Doing this for precision and recall gives you a precision-recall curve

A Precision-Recall Curve



- Each point corresponds to a result for the top k ranked hits (*k* = 1, 2, 3, 4, . . .)
- Interpolation (in red): Take maximum of all future points
- Rationale for interpolation: The user is willing to look at more stuff if both precision and recall get better

What we Need for a Benchmark

- A collection of documents
- Documents must be representative of the documents we expect to see in reality
- A collection of information needs
- . . .which we will often incorrectly refer to as queries
- Information needs must be representative of the information needs we expect to see in reality
- Human relevance assessments
- We need to hire/pay “judges” or assessors to do this.
- Expensive, time-consuming
- Judges must be representative of the users we expect to see in reality

Standard Relevance Benchmark: TREC

- TREC = Text Retrieval Conference (TREC)
- Organized by the U.S. National Institute of Standards and Technology (NIST)
- TREC is actually a set of several different relevance benchmarks
- Best known: TREC Ad Hoc, used for first 8 TREC evaluations between 1992 and 1999
- 1.89 million documents, mainly newswire articles, 450 information needs
- No exhaustive relevance judgments – too expensive
- Rather, NIST assessors' relevance judgments are available only for the documents that were among the top k returned for some system which was entered in the TREC evaluation for which the information need was developed

Validity of Relevance Assessments

- Relevance assessments are only usable if they are consistent
- If they are not consistent, then there is no “truth” and experiments are not repeatable
- How can we measure this consistency or agreement among judges?
- → Kappa measure

Kappa Measure

- Kappa is measure of how much judges agree or disagree
- Designed for categorical judgments
- Corrects for chance agreement
- *P*(*A*) = proportion of time judges agree
- *P*(*E*) = what agreement would we get by chance

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

- *k* =? for (i) chance agreement (ii) total agreement
- Values of *k* in the interval [2/3, 1.0] are seen as acceptable.
- With smaller values: need to redesign relevance assessment methodology used etc.

Calculating the Kappa Statistic

Judge 1 Relevance	Judge 2 Relevance			
	Yes	No	Total	
Yes	300	20	320	
No	10	70	80	
Total	310	90	400	

P(*A*) = (300 + 70)/400 = 370/400 = 0.925

P(*nonrelevant*) = (80 + 90)/(400 + 400) = 170/800 = 0.2125
P(*relevant*) = (320 + 310)/(400 + 400) = 630/800 = 0.7878
Probability that the two judges agreed by chance *P*(*E*) =
P(*nonrelevant*)^2 + *P*(relevant)^2 = 0.2125^2 + 0.7878^2 = 0.665
Kappa statistic *κ* = (*P*(*A*) − *P*(*E*))/(1 − *P*(*E*)) =
(0.925 − 0.665)/(1 − 0.665) = 0.776 (still in acceptable range)

Evaluation at Large Search Engines

- Recall is difficult to measure on the web
- Search engines often use precision at top *k*, e.g., *k* = 10 . . .
- . . . or use measures that reward you more for getting rank 1 right than for getting rank 10 right
- Search engines also use non-relevance-based measures
- Example 1: clickthrough on first result
- Not very reliable if you look at a single clickthrough (you may realize after clicking that the summary was misleading and the document is nonrelevant) . . .
- . . . but pretty reliable in the aggregate
- Example 2: Ongoing studies of user behavior in the lab
- Example 3: A/B testing

A/B Testing

- Purpose: Test a single innovation
- Prerequisite: You have a large search engine up and running
- Have most users use old system
- Divert a small proportion of traffic (e.g., 1%) to the new system that includes the innovation
- Evaluate with an “automatic” measure like clickthrough on first result
- Now we can directly see if the innovation does improve user happiness
- Probably the evaluation methodology that large search engines trust most

How do we Present Results to the User?

- Most often: as a list – aka “10 blue links”
- How should each document in the list be described?
- This description is crucial
- The user often can identify good hits (= relevant hits) based on the description
- No need to “click” on all documents sequentially

Summaries

- Two basic kinds: (i) static (ii) dynamic
- A static summary of a document is always the same, regardless of the query that was issued by the user.
- Dynamic summaries are query-dependent. They 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 words of the document
- 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: complex NLP to synthesize/generate a summary
- For most IR applications: not quite ready for prime time yet

Dynamic Summaries

- Present one or more “windows” or snippets within the document that contain several of the query terms.
- Prefer snippets in which query terms occurred as a phrase
- Prefer snippets in which query terms occurred jointly in a small window
- The summary that is computed this way gives the entire content of the window – all terms, not just the query terms.

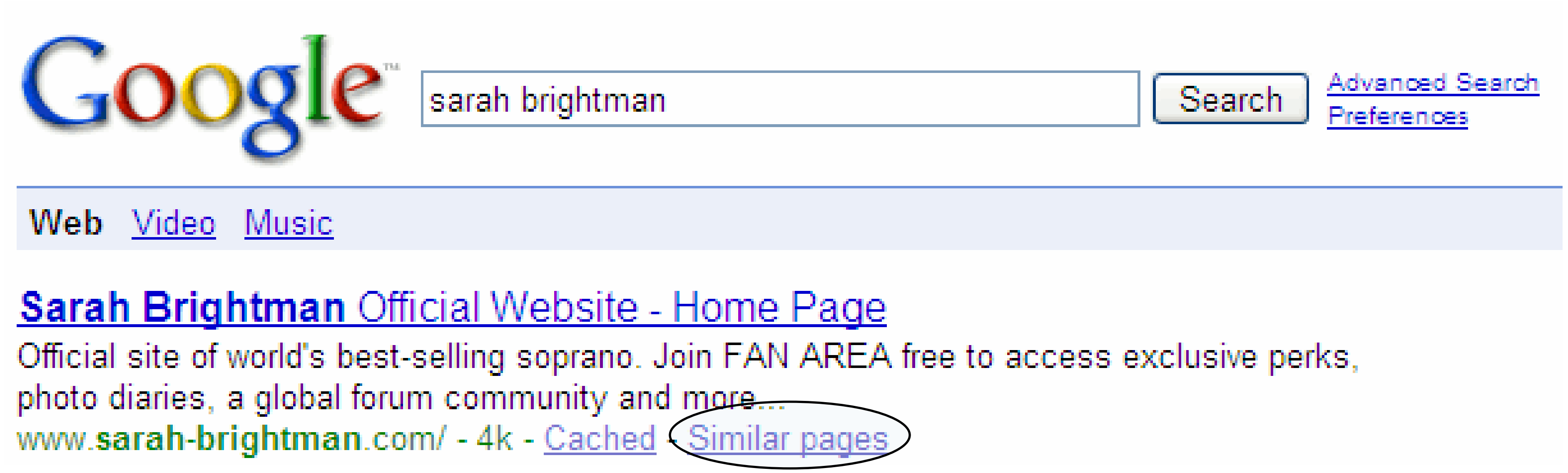
Relevance Feedback

- 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

Relevance Feedback

- We will use *ad hoc retrieval* to refer to regular retrieval without relevance feedback
- We now look at examples of relevance feedback that highlight different aspects

Similar Pages



sarah brightman

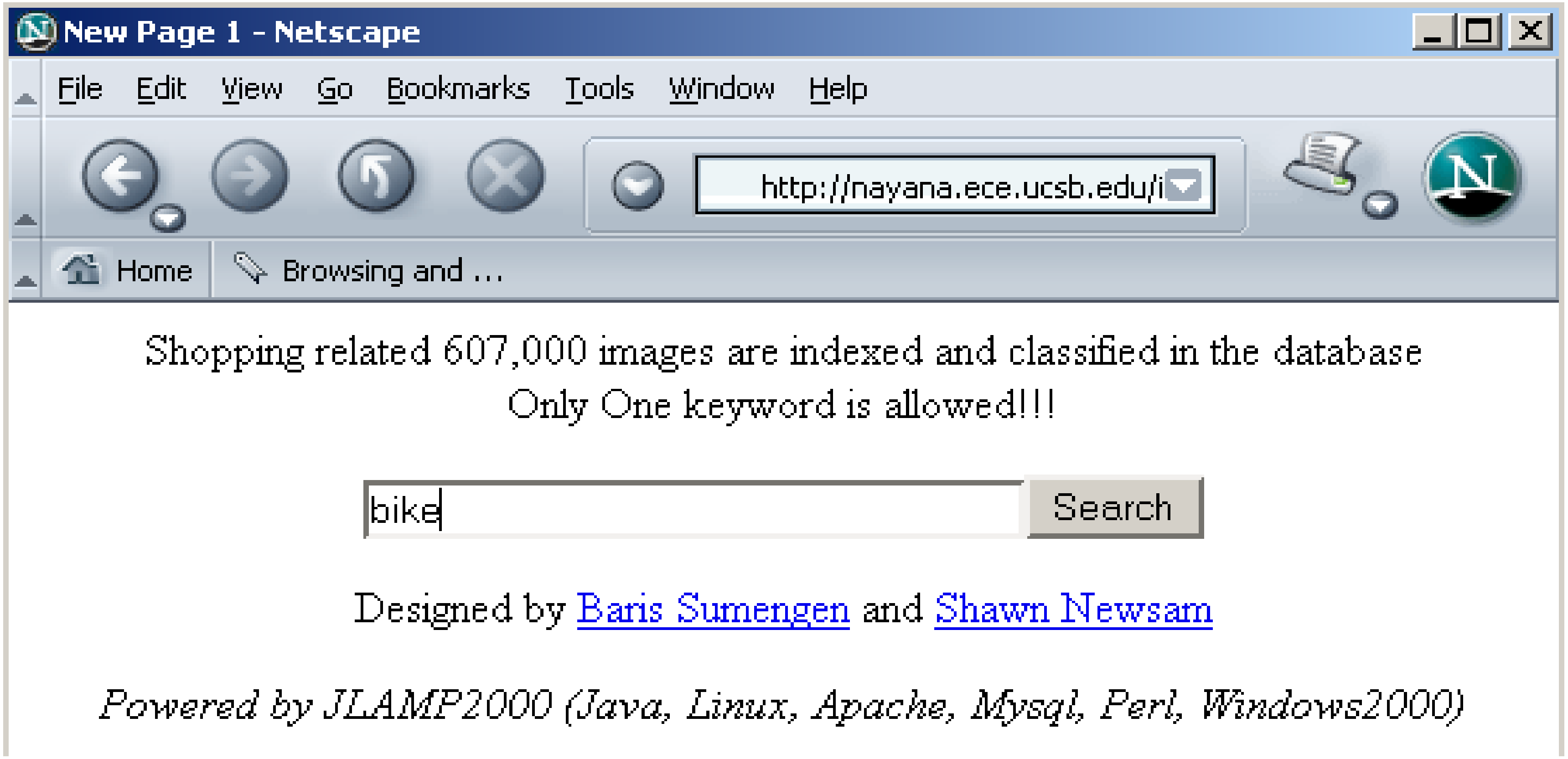
Sarah Brightman Official Website - Home Page

Official site of world's best-selling soprano. Join FAN AREA free to access exclusive perks, photo diaries, a global forum community and more...

www.**sarah-brightman**.com/ - 4k - Cached - Similar pages

Relevance Feedback: Example

- Image search engine http://nayana.ece.ucsb.edu/imsearch/imsearch.html


New Page 1 - Netscape
File Edit View Go Bookmarks Tools Window Help
http://nayana.ece.ucsb.edu/i
Home
Browsing and ...
bike
Search

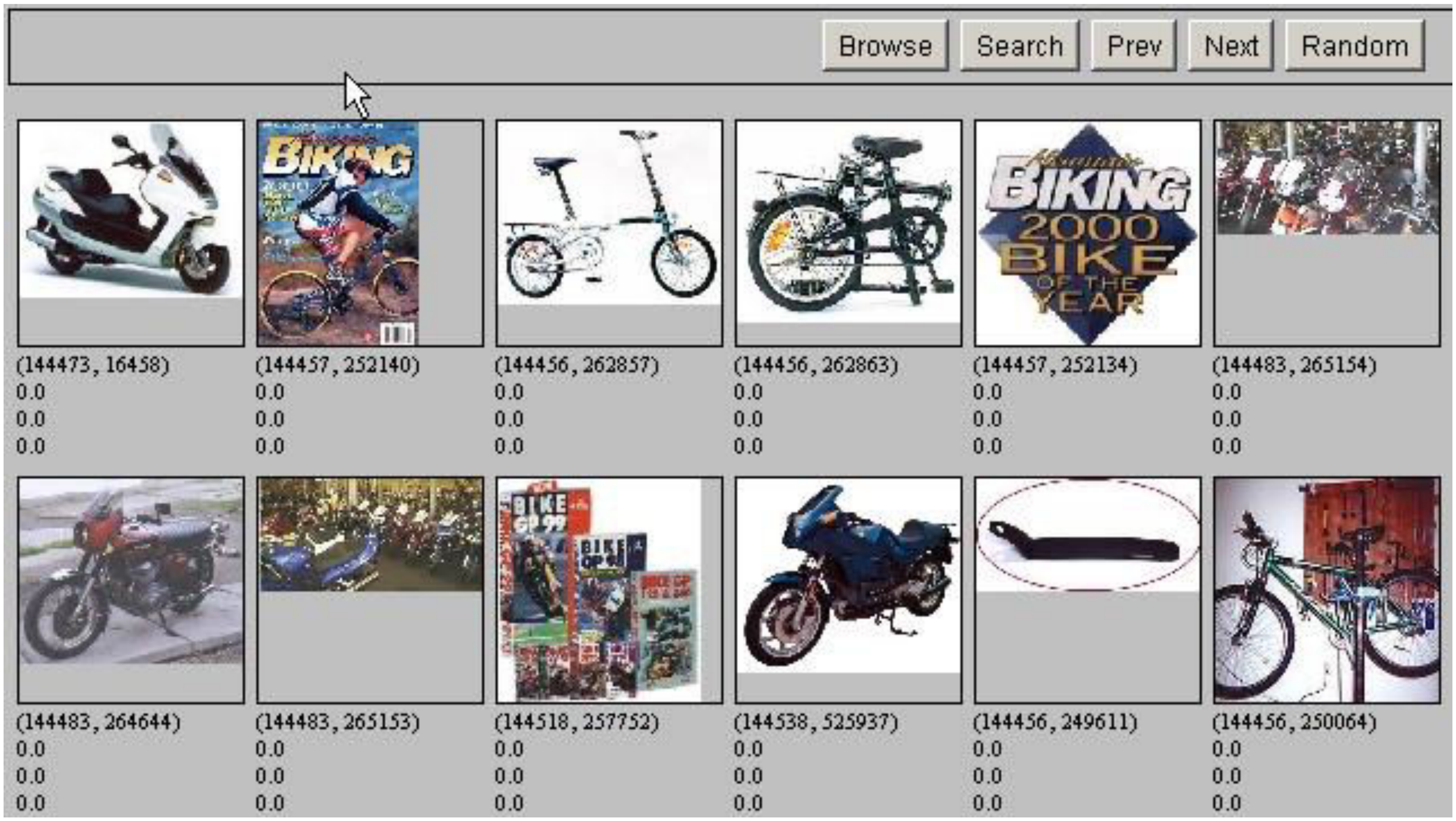
Shopping related 607,000 images are indexed and classified in the database

Only One keyword is allowed!!!

Designed by Baris Sumengen and Shawn Newsam

Powered by JLAMP2000 (Java, Linux, Apache, Mysql, Perl, Windows2000)

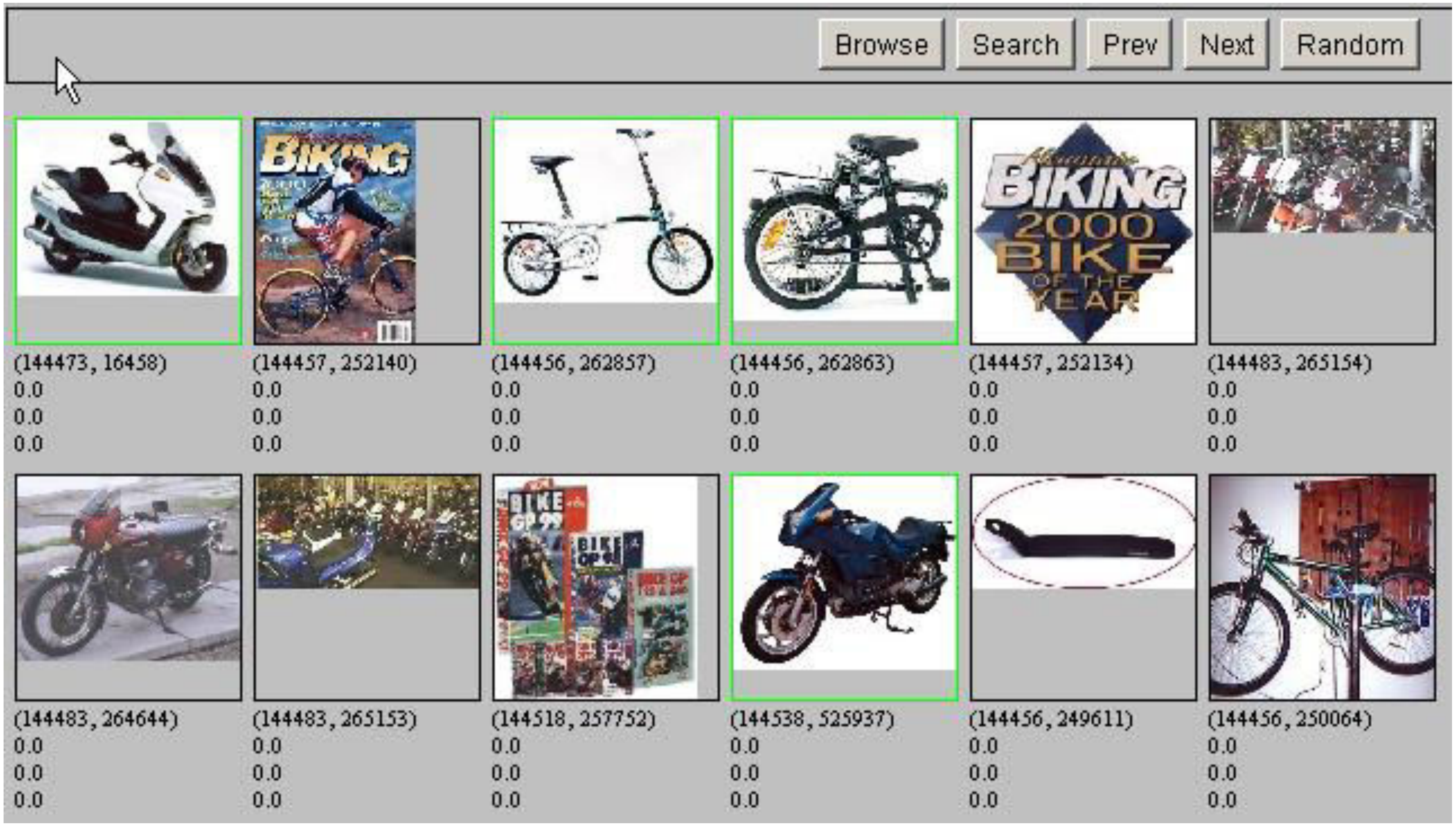
Results for Initial Query



```
(144473, 16458)   (144457, 252140)  (144456, 262857)  (144456, 262863)  (144457, 252134)  (144483, 265154)
0.0               0.0               0.0               0.0               0.0               0.0
0.0               0.0               0.0               0.0               0.0               0.0
0.0               0.0               0.0               0.0               0.0               0.0

(144483, 264644)  (144483, 265153)  (144518, 257752)  (144538, 525937)  (144456, 249611)  (144456, 250064)
0.0               0.0               0.0               0.0               0.0               0.0
0.0               0.0               0.0               0.0               0.0               0.0
0.0               0.0               0.0               0.0               0.0               0.0
```

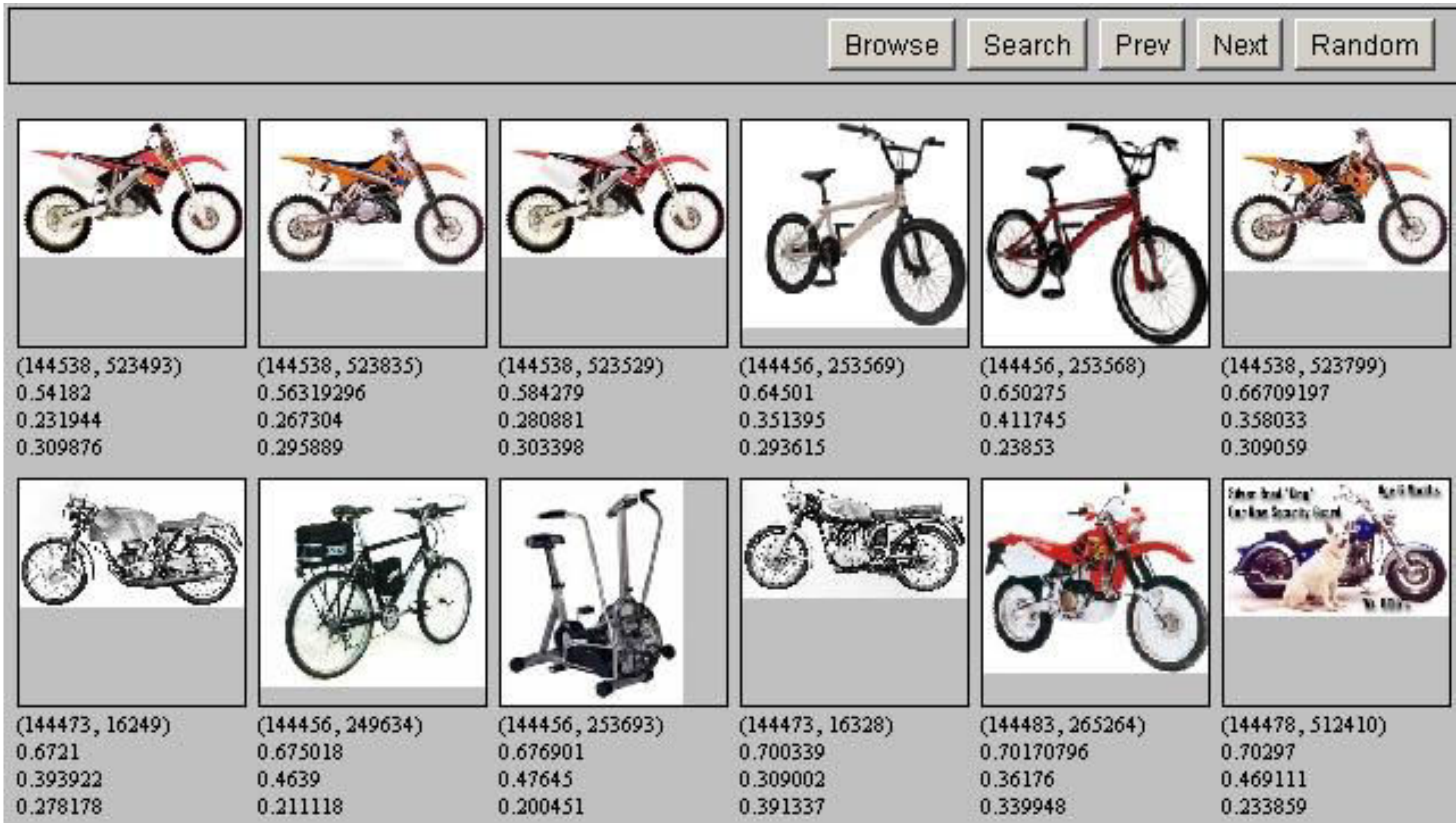
Relevance Feedback



```
(144473, 16458)   (144457, 252140)  (144456, 262857)  (144456, 262863)  (144457, 252134)  (144483, 265154)
0.0               0.0               0.0               0.0               0.0               0.0
0.0               0.0               0.0               0.0               0.0               0.0
0.0               0.0               0.0               0.0               0.0               0.0

(144483, 264644)  (144483, 265153)  (144518, 257752)  (144538, 525937)  (144456, 249611)  (144456, 250064)
0.0               0.0               0.0               0.0               0.0               0.0
0.0               0.0               0.0               0.0               0.0               0.0
0.0               0.0               0.0               0.0               0.0               0.0
```

Results after Relevance Feedback



(144538, 523493)	(144538, 523835)	(144538, 523529)	(144456, 253569)	(144456, 253568)	(144538, 523799)
0.54182	0.56319296	0.584279	0.64501	0.650275	0.66709197
0.231944	0.267304	0.280881	0.351395	0.411745	0.358033
0.309876	0.295889	0.303398	0.293615	0.23853	0.309059
(144473, 16249)	(144456, 249634)	(144456, 253693)	(144473, 16328)	(144483, 265264)	(144478, 512410)
0.6721	0.675018	0.676901	0.700339	0.70170796	0.70297
0.393922	0.4639	0.47645	0.309002	0.36176	0.469111
0.278178	0.211118	0.200451	0.391337	0.339948	0.233859

Initial Query/Results

- Initial query: *New space satellite applications*

  + 1. 0.539, 08/13/91, NASA Hasn't Scrapped Imaging Spectrometer

  + 2. 0.533, 07/09/91, NASA Scratches Environment Gear From Satellite Plan

  3. 0.528, 04/04/90, Science Panel Backs NASA Satellite Plan, But Urges Launches of Smaller Probes
  4. 0.526, 09/09/91, A NASA Satellite Project Accomplishes Incredible Feat: Staying Within Budget
  5. 0.525, 07/24/90, Scientist Who Exposed Global Warming Proposes Satellites for Climate Research
  6. 0.524, 08/22/90, Report Provides Support for the Critics Of Using Big Satellites to Study Climate
  7. 0.516, 04/13/87, Arianespace Receives Satellite Launch Pact From Telesat Canada
  8. 0.509, 12/02/87, Telecommunications Tale of Two Companies

- + User then marks relevant documents with "+".

Expanded Query after Relevance Feedback

2.074	new	15.106	space
30.816	satellite	5.660	application
5.991	nasa	5.196	eos
4.196	launch	3.972	aster
3.516	instrument	3.446	arianespace
3.004	bundespost	2.806	ss
2.790	rocket	2.053	scientist
2.003	broadcast	1.172	earth
0.836	oil	0.646	measure

Results for Expanded Query

2

1 1. 0.513, 07/09/91, NASA Scratches Environment Gear From Satellite Plan

2. 0.500, 08/13/91, NASA Hasn't Scrapped Imaging Spectrometer

3. 0.493, 08/07/89, When the Pentagon Launches a Secret Satellite, Space Sleuths Do Some Spy Work of Their Own

8 4. 0.493, 07/31/89, NASA Uses 'Warm' Superconductors For Fast Circuit

5. 0.492, 12/02/87, Telecommunications Tale of Two Companies

6. 0.491, 07/09/91, Soviets May Adapt Parts of SS-20 Missile For Commercial Use

7. 0.490, 07/12/88, Gaping Gap: Pentagon Lags in Race To Match the Soviets In Rocket Launchers

8. 0.490, 06/14/90, Rescue of Satellite By Space Agency To Cost $90 Million

Key Concept: Centroid

- The <u>centroid</u> 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

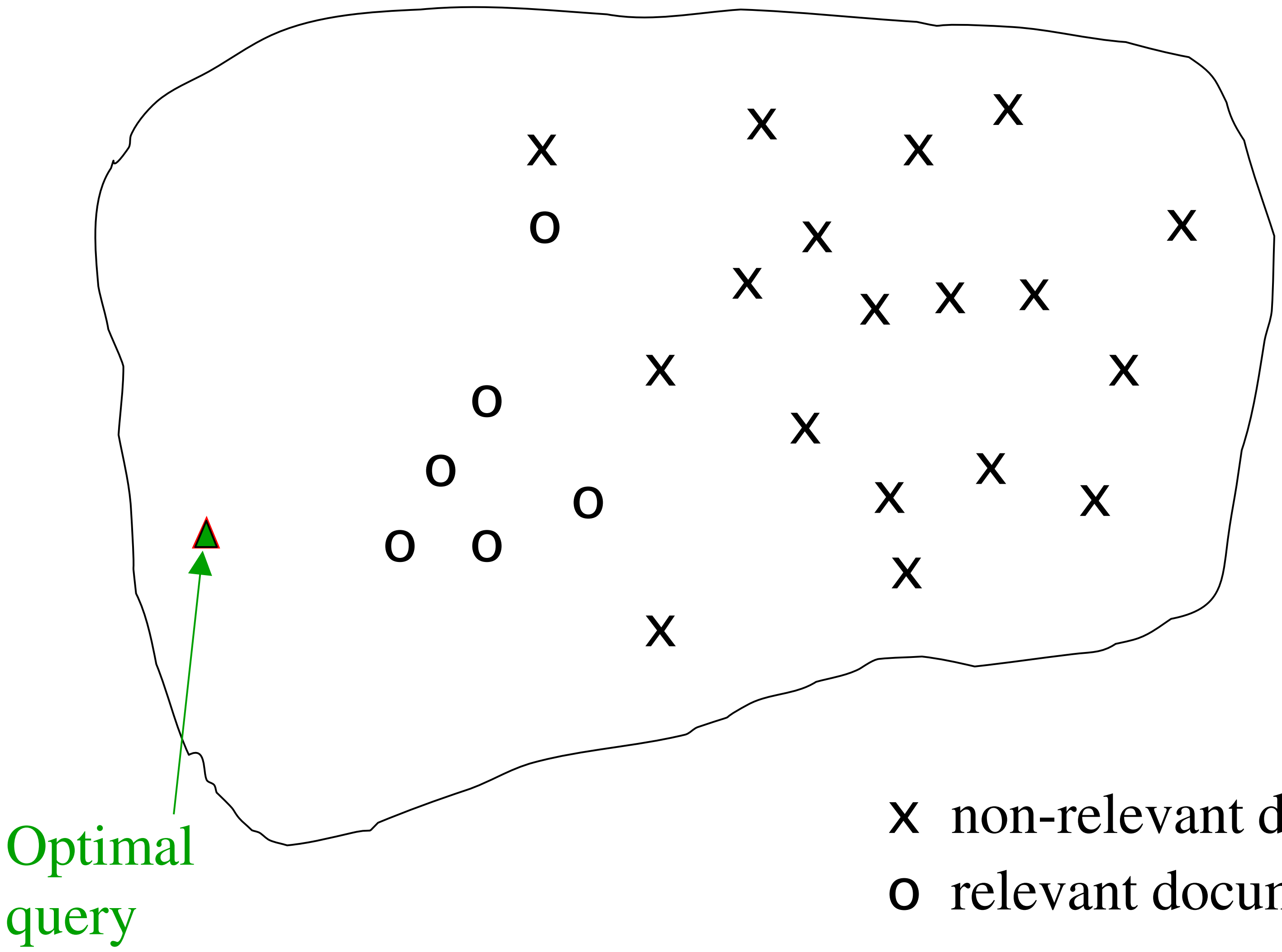
$$\vec{q}_{opt} = \arg\max_{\vec{q}} [\cos(\vec{q}, \vec{\mu}(C_r)) - \cos(\vec{q}, \vec{\mu}(C_{nr}))]$$

- 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:

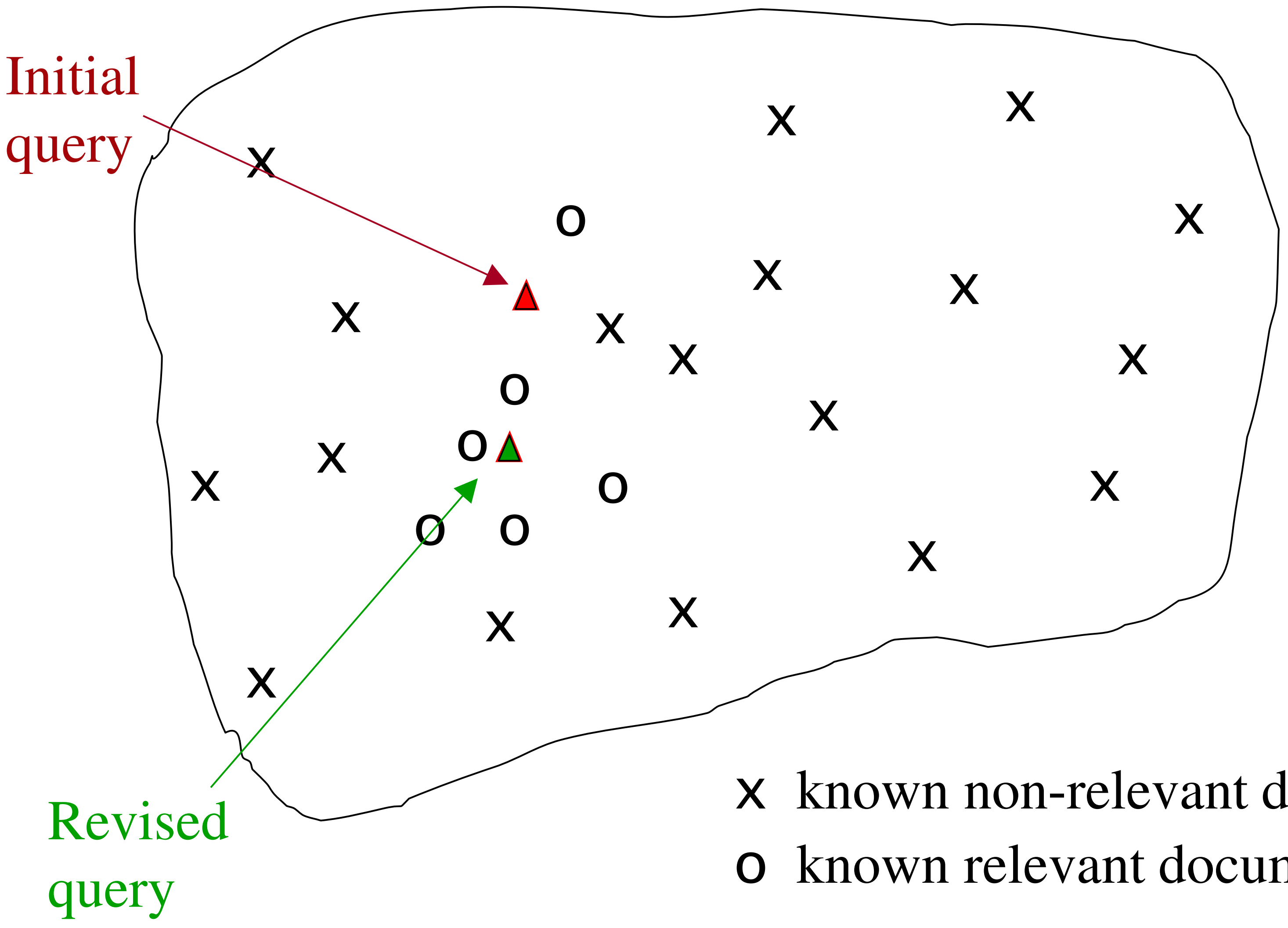
$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j$$

- 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

Subtleties to Note

- Tradeoff α vs. β/γ : If we have a lot of judged documents, we want a higher β/γ.
- Some weights in query vector can go negative
  - Negative term weights are ignored (set to 0)

Relevance Feedback on Initial Query



x known non-relevant documents

o known relevant documents

Relevance Feedback in Vector Spaces

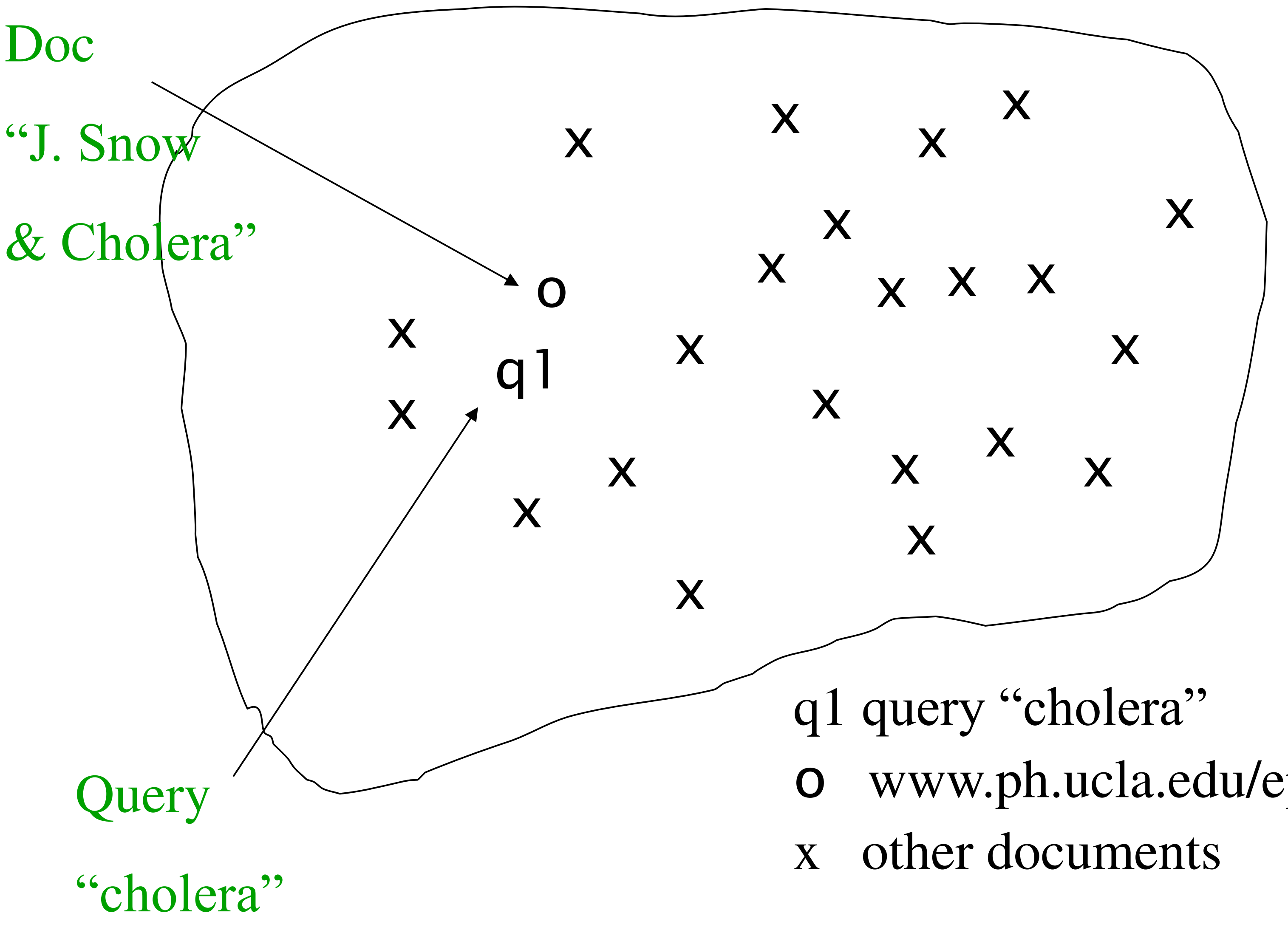
- We can modify the query based on relevance feedback and apply standard vector space model
- Use only the docs that were marked
- Relevance feedback can improve recall and precision
- Relevance feedback is most useful for increasing *recall* in situations where recall is important
  - Users can be expected to review results and to take time to iterate

Positive vs Negative Feedback

- Positive feedback is more valuable than negative feedback (so, set $\gamma < \beta$; e.g. $\gamma = 0.25$, $\beta = 0.75$)
- Many systems only allow positive feedback ($\gamma=0$)

Why?

Aside: Vector Space can be Counterintuitive



q1 query “cholera”

o www.ph.ucla.edu/epi/snow.html

x other documents

High-dimensional Vector Spaces

- The queries “cholera” and “john snow” are far from each other in vector space.
- How can the document “John Snow and Cholera” be close to both of them?
- Our intuitions for 2- and 3-dimensional space don't work in >10,000 dimensions
- 3 dimensions: If a document is close to many queries, then some of these queries must be close to each other
- Doesn't hold for a high-dimensional space

Relevance Feedback: Assumptions

- A1: User has sufficient knowledge for initial query.
- A2: Relevance prototypes are “well-behaved”.
  - Term distribution in relevant documents will be similar
  - Term distribution in non-relevant documents will be different from those in relevant documents
    - Either: All relevant documents are tightly clustered around a single prototype.
    - Or: There are different prototypes, but they have significant vocabulary overlap.
    - Similarities between relevant and irrelevant documents are small

Violation of A1

- User does not have sufficient initial knowledge.
- Examples:
  - Misspellings (Brittany Speers).
  - Cross-language information retrieval (hígado).
  - Mismatch of searcher’s vocabulary vs. collection vocabulary
    - Cosmonaut/astronaut

Violation of A2

- There are several relevance prototypes.
- Examples:
  - Burma/Myanmar
  - Contradictory government policies
  - Pop stars that worked at Burger King
- Often: instances of a general concept
- Good editorial content can address problem
  - Report on contradictory government policies

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

Why?

- Users are often reluctant to provide explicit feedback
- It's often harder to understand why a particular document was retrieved after applying relevance feedback

Evaluation of Relevance Feedback Strategies

- Use q_0 and compute precision and recall graph
- Use q_m and compute precision recall graph
  - Assess on all documents in the collection
    - Spectacular improvements, but … it's cheating!
    - Partly due to known relevant documents ranked higher
    - Must evaluate with respect to documents not seen by user
  - Use documents in residual collection (set of documents minus those assessed relevant)
    - Measures usually then lower than for original query
    - But a more realistic evaluation
    - Relative performance can be validly compared
- Empirically, one round of relevance feedback is often very useful. Two rounds is sometimes marginally useful.

Evaluation of Relevance Feedback

- Second method – assess only the docs *not* rated by the user in the first round
  - Could make relevance feedback look worse than it really is
  - Can still assess relative performance of algorithms
- Most satisfactory – use two collections each with their own relevance assessments
  - q_0 and user feedback from first collection
  - q_m run on second collection and measured

Evaluation: Caveat

- True evaluation of usefulness must compare to other methods taking the same amount of time.
- Alternative to relevance feedback: User revises and resubmits query.
- Users may prefer revision/resubmission to having to judge relevance of documents.
- There is no clear evidence that relevance feedback is the “best use” of the user’s time.

Relevance Feedback on the Web

- Some search engines offer a similar/related pages feature (this is a trivial form of relevance feedback)
  - Google (link-based)
  - Altavista
  - Stanford WebBase

α/β/γ ??

- But some don’t because it’s hard to explain to average user:
  - Alltheweb
  - bing
  - Yahoo
- Excite initially had true relevance feedback, but abandoned it due to lack of use.

Excite Relevance Feedback

Spink et al. 2000

- Only about 4% of query sessions from a user used relevance feedback option
  - Expressed as "More like this" link next to each result
- But about 70% of users only looked at first page of results and didn't pursue things further
  - So 4% is about 1/8 of people extending search
- Relevance feedback improved results about 2/3 of the time

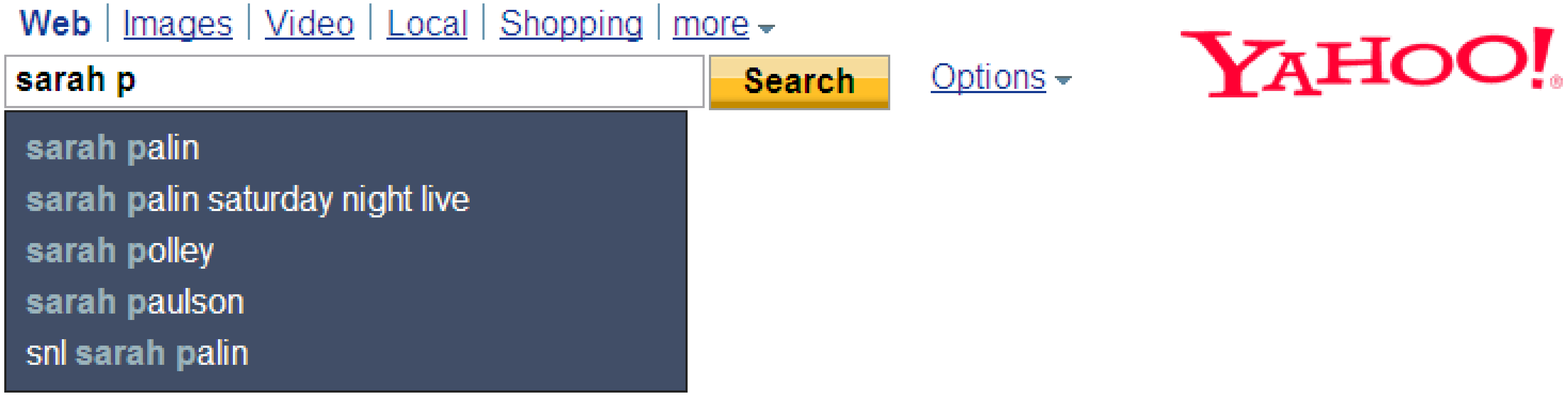
Pseudo Relevance Feedback

- 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

Thesaurus-Based Query Expansion

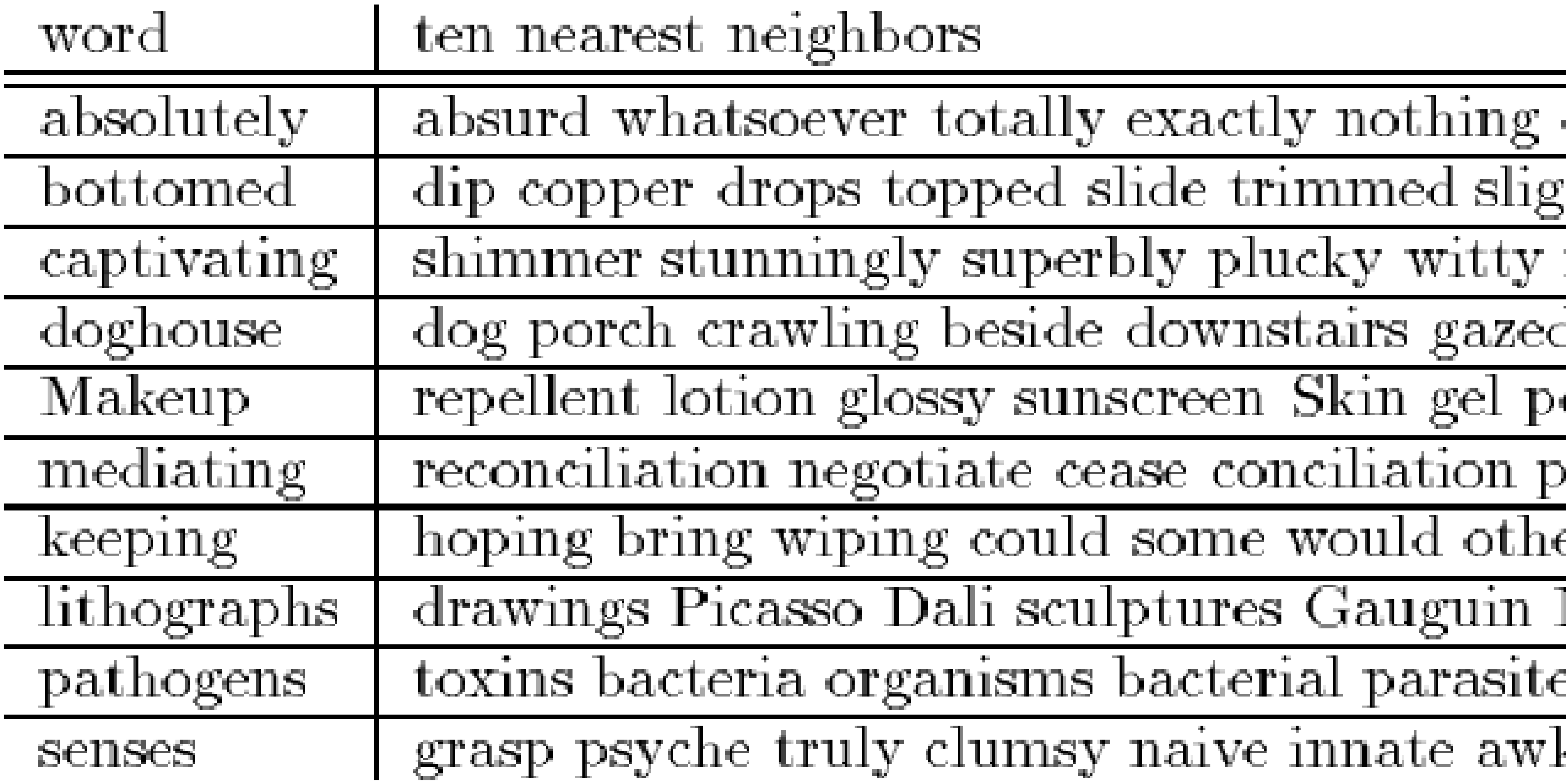
- For each term, *t*, in a query, expand the query with synonyms and related words of *t* from the thesaurus
  - feline → feline cat
- May weight added terms less than original query terms.
- Generally increases recall
- Widely used in many science/engineering fields
- May significantly decrease precision, particularly with ambiguous terms.
  - “interest rate” → “interest rate fascinate evaluate”
- There is a high cost of manually producing a thesaurus
  - And for updating it for scientific changes

Automatic Thesaurus Generation

- Attempt to generate a thesaurus automatically by analyzing the collection of documents
- Fundamental notion: similarity between two words
- Definition 1: Two words are similar if they co-occur with similar words.
- Definition 2: Two words are similar if they occur in a given grammatical relation with the same words.
- You can harvest, peel, eat, prepare, etc. apples and pears, so apples and pears must be similar.
- Co-occurrence based is more robust, grammatical relations are more accurate.

Why?

Automatic Thesaurus Generation Example



word	ten nearest neighbors
absolutely	absurd whatsoever totally exactly nothing
bottomed	dip copper drops topped slide trimmed slig
captivating	shimmer stunningly superbly plucky witty
doghouse	dog porch crawling beside downstairs gaze
Makeup	repellent lotion glossy sunscreen Skin gel p
mediating	reconciliation negotiate cease conciliation p
keeping	hoping bring wiping could some would othe
lithographs	drawings Picasso Dali sculptures Gauguin
pathogens	toxins bacteria organisms bacterial parasite
senses	grasp psyche truly clumsy naive innate awk

Automatic Thesaurus Generation Discussion

- Quality of associations is usually a problem
- Term ambiguity may introduce irrelevant statistically correlated terms
  - "Apple computer" $\rightarrow$ "Apple red fruit computer"
- Problems:
  - False positives: Words deemed similar that are not
  - False negatives: Words deemed dissimilar that are similar
- Since terms are highly correlated anyway, expansion may not retrieve many additional documents

Indirect Relevance Feedback

- On the web, DirectHit introduced a form of indirect relevance feedback
- DirectHit ranked documents higher that users look at more often
  - Clicked on links are assumed likely to be relevant
    - Assuming the displayed summaries are good, etc.
- Globally: Not necessarily user or query specific
  - This is the general area of clickstream mining
- Today – handled as part of machine-learned ranking