Information Retrieval

- Ranking & Score
- Vector similarity / distance
- Evaluation
- Implicit user feedback

Development:
Moshe Friedman

Credits:
Yoav Goldberg, Ido Dagan, Reut Tsarfaty , Moshe Koppel, Wei Song, David Bamman, Ed Grefenstette, Chris Manning, Tsvi Kuflik, Hinrich Schütze, Christina Lioma and more

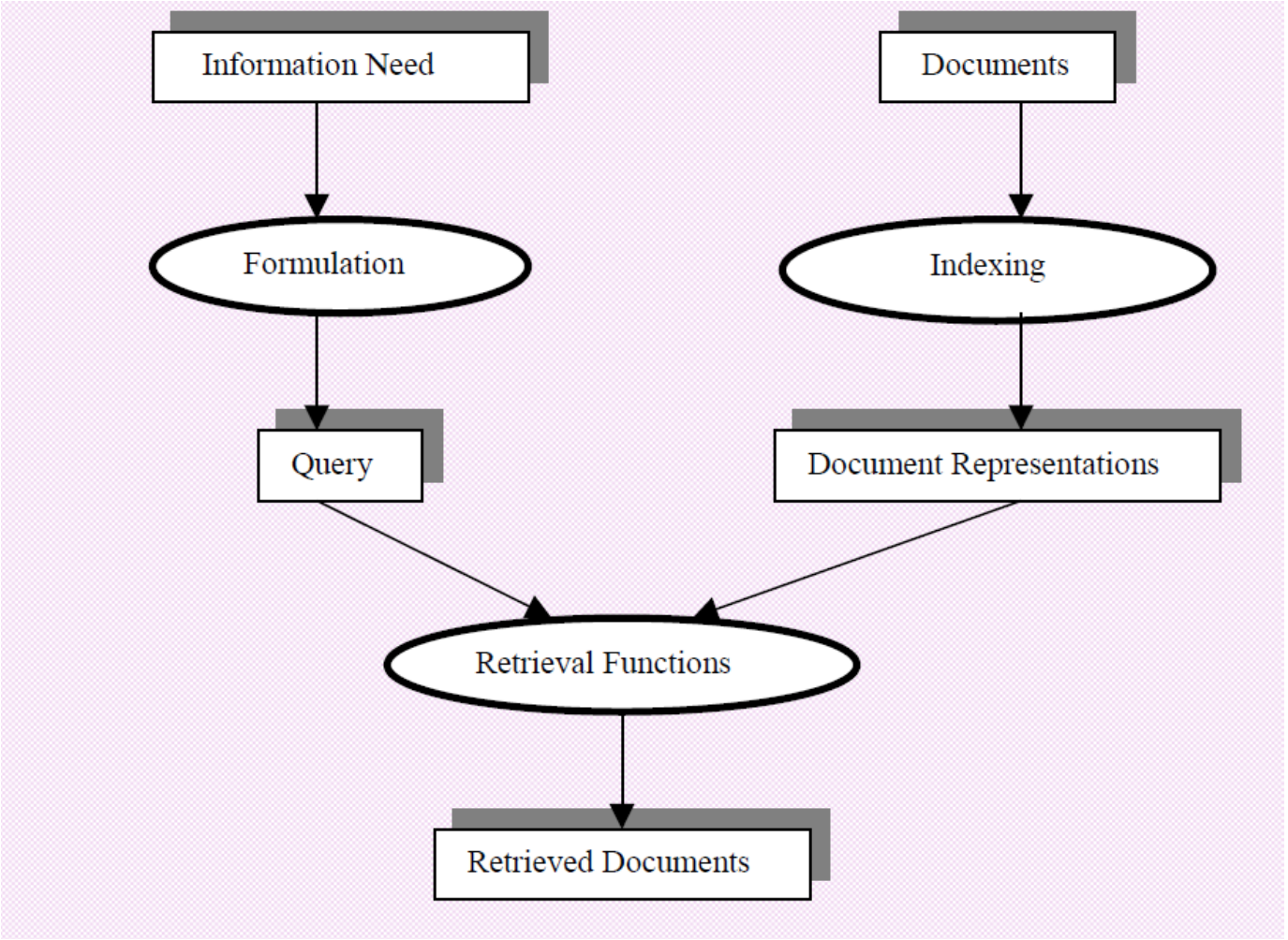
Information Retrieval - administration

Moshe Friedman

Email: moshefr.teach@gmail.com

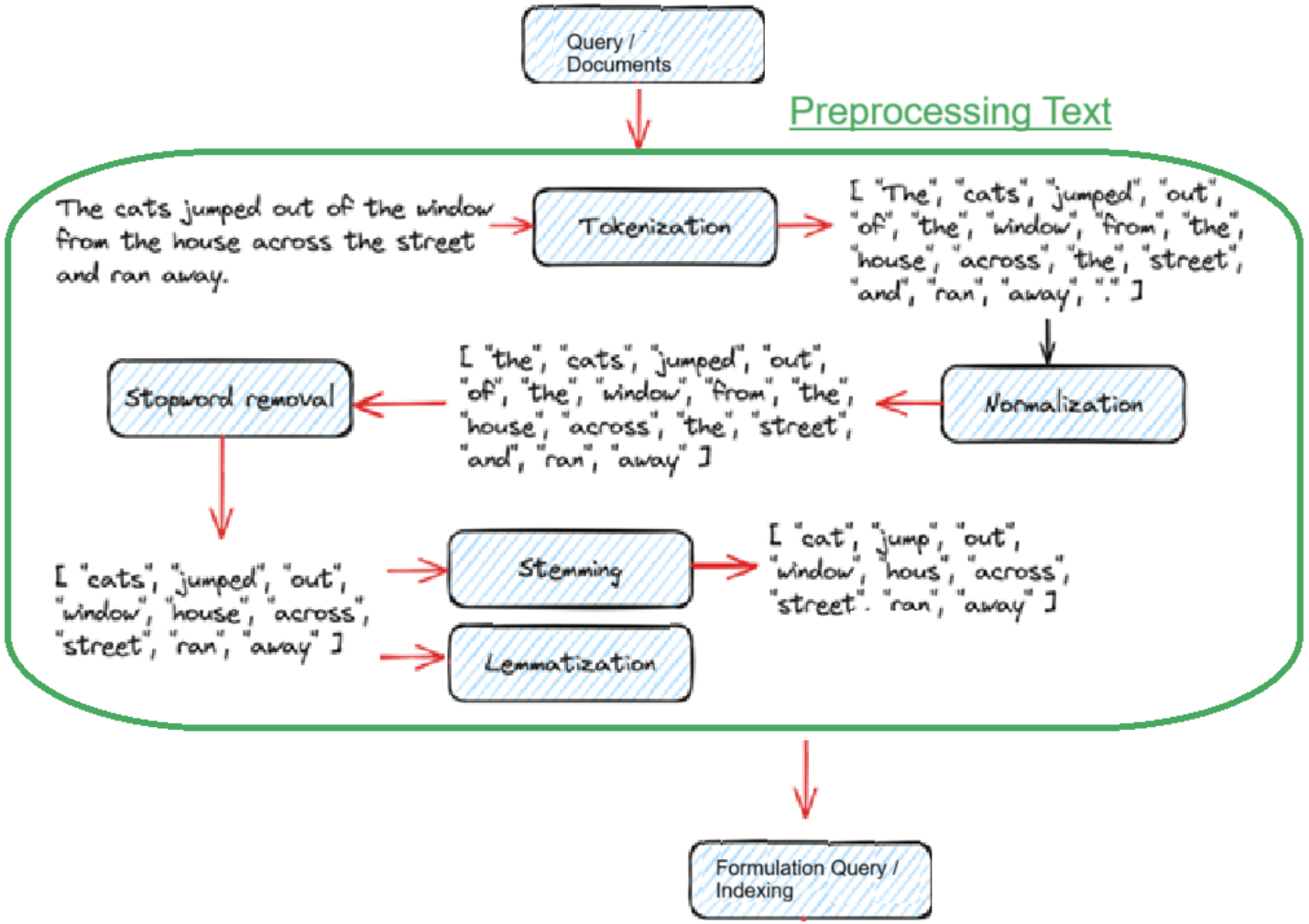
Reception time: before/after lesson/zoom with coordination

Information Retrieval Flow Process



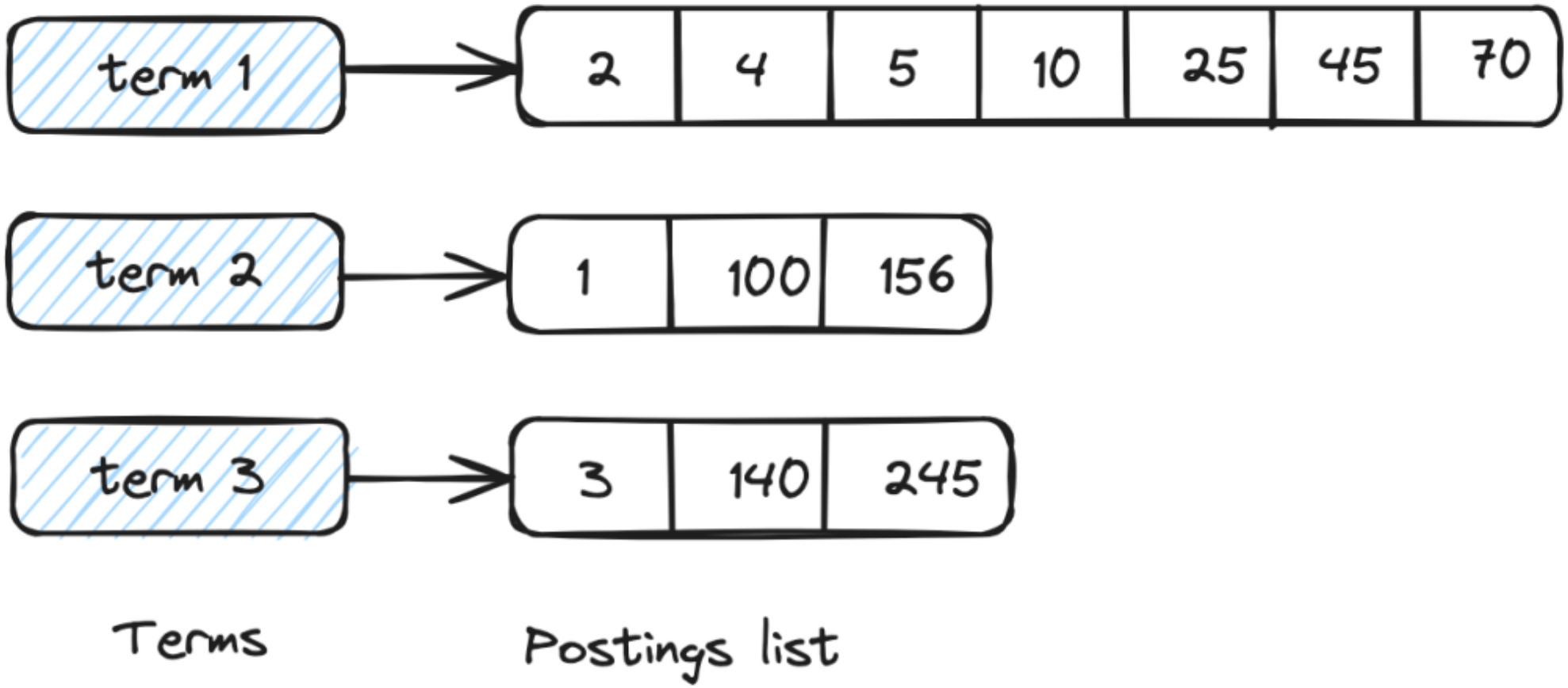
Credit: devopedia.org

Text Preprocessing (for Documents & Query)



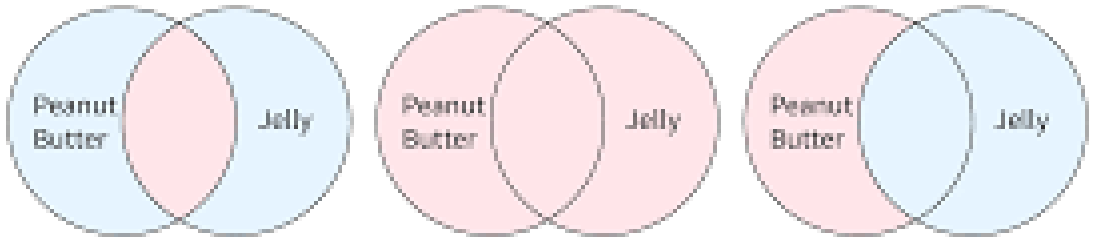
Credit: thatgirlcoder.com

Inverted Index (for Boolean Retrieval Model)



Credit: thatgirlcoder.com

Inverted Index (for Boolean Retrieval Model)



$\overline{q_{dnf}} = (\text{Cat} \wedge \text{Nice}) \vee (\text{Cat} \wedge \text{Afraid})$

q_{cc} q_{cc}

SCALER Topics

D1, D3, D4 and D5 retrieved

$q_{cc} = (\text{Cat} \wedge \text{Nice})$	K2	K3
D1	**1**	**1**
D2	0	0
D3	1	0
D4	1	0
D5	1	0

$q_{cc} = (\text{Cat} \wedge \text{Afraid})$	K2	K8
D1	1	0
D2	0	0
D3	**1**	**1**
D4	**1**	**1**
D5	**1**	**1**

Credit: scaler.com

Some of last week's topics

- What's text, Text as data – properties and challenges,
- Morphological analysis, Lemmatization, Stemming (Porter Stemmer)
- Tokenization (Segmentation)
- token normalization – Acronyms, Case, Accents, Punctuation marks, Sentence breaking, Part-of-speech tagging, Stop word removal
- Bag of Word (BOW) model
  - terms: original tokens, normalized words, partial word, token n-grams, character n-grams
  - Term frequency – Boolean, Term Count
- Positional Index, Proximity queries
- Soundex
- Spelling Edit Distance

Introduction to

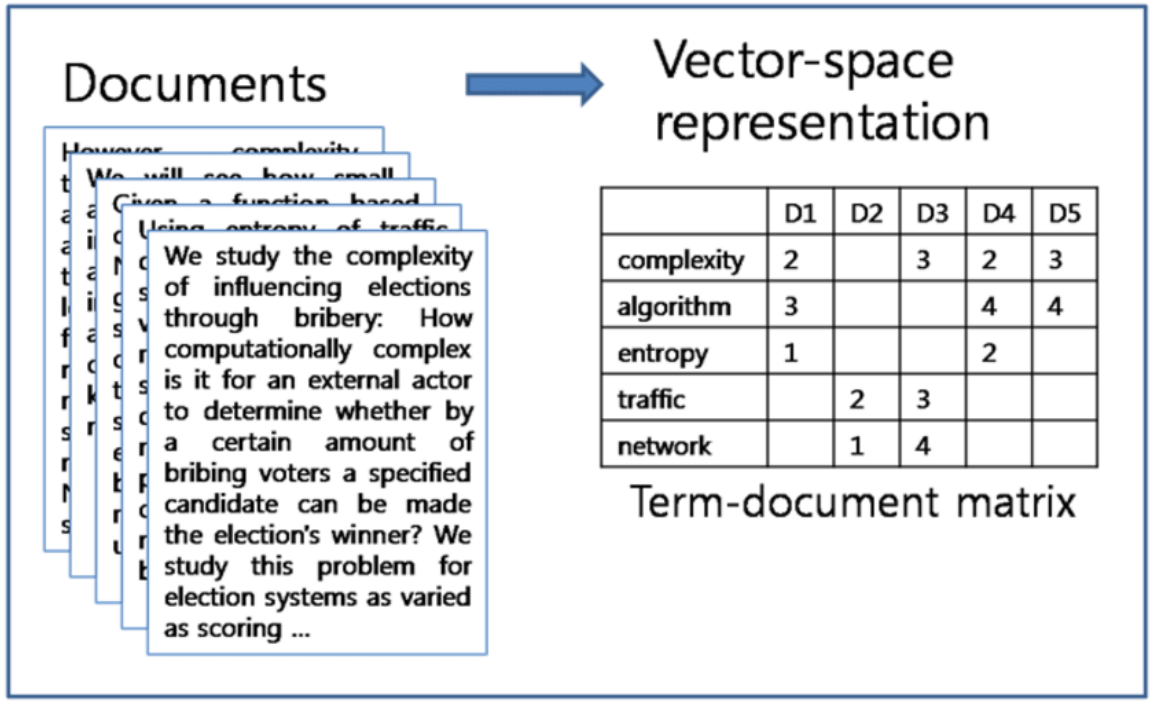
Information Retrieval

CS276: Information Retrieval and Web Search

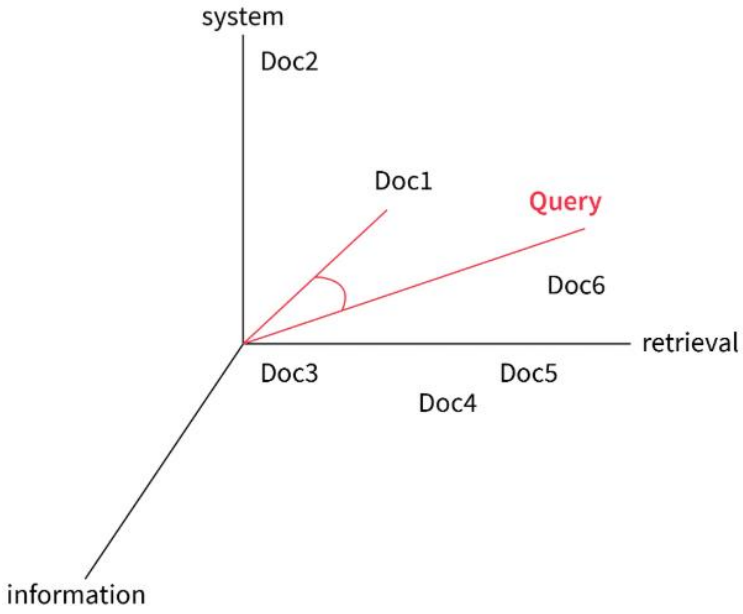
Pandu Nayak and Prabhakar Raghavan

Lecture 6: Scoring, Term Weighting and the Vector Space Model

Vector Space



	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		



Credit: devopedia.org

This lecture; IIR Sections 6.2-6.4.3

- Ranked retrieval
- Scoring documents
- Term frequency
- Collection statistics
- Weighting schemes
- Vector space scoring

Ranked retrieval

- Thus far, our queries have all been Boolean.
  - Documents either match or don't.
- Good for expert users with precise understanding of their needs and the collection.
  - Also good for applications: Applications can easily consume 1000s of results.
- Not good for most users.
  - Most users are incapable 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: "*standard user dlink 650*" → 200,000 hits
- Query 2: "*standard user dlink 650 no card found*": 0 hits
- It takes a lot of skill to come up with a query that produces a manageable number of hits.
  - AND gives too few; OR gives too many

Ranked retrieval models

- Rather than a set of documents satisfying a query expression, in ranked retrieval, the system returns an ordering over the (top) documents in the collection for a query
- Free text queries: Rather than a query language of operators and expressions, the user's query is just one or more words in a human language
- In principle, there are two separate choices here, but in practice, ranked retrieval has normally been associated with free text queries and vice versa

Recall (Lecture 2): Binary term-document incidence matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Each document is represented by a binary vector $\in \{0,1\}^{|V|}$

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*.
  - Note: Frequency means count in IR
- 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 1 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

$$w_{t,d} = \begin{cases} 1 + \log_2 \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

- 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.

Rare terms are more informative

- Rare terms are more informative than frequent terms
  - Recall stop words
- 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*.

Collection vs. Document frequency

- Collection frequency of *t* is the number of occurrences of *t* in the collection
- Document frequency of t is the number of documents in which t occurs
- Example:

Word	Collection frequency	Document frequency
insurance	10440	3997
try	10422	8760

- Which word is for better search (gets higher weight)

idf weight

- df_t is the document frequency of *t*: the number of documents that contain *t*
  - df_t is an inverse measure of the informativeness of *t*
  - $\mathrm{df}_t \leq N$
- We define the idf (inverse document frequency) of *t* by

$$\mathrm{idf}_t = \log_2(N/\mathrm{df}_t)$$

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

idf example, suppose $N = 256$

term	df_t	idf_t
calpurnia	1	8
animal	16	4
sunday	32	3
fly	64	2
under	128	1
the	256	0

$$\text{idf}_t = \log_2 (N/\text{df}_t)$$

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

Effect of idf on ranking

- Does idf have an effect on ranking for one-term queries, like
  - iPhone
- idf has no effect on ranking one term queries
  - idf affects the ranking of documents for queries with at least two terms
- For the query capricious person, idf weighting makes occurrences of capricious count for much more in the final document ranking than occurrences of person.

tf-idf weighting

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

$$w_{t,d} = \log(1 + \mathrm{tf}_{t,d}) \times \log_2(N/\mathrm{df}_t)$$

- 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

Score for a document given a query

$$\text{Score}(q,d) = \sum_{t \in q \cap d} \text{tf.idf}_{t,d}$$

- There are many variants
  - How “tf” is computed (with/without logs)
  - Whether the terms in the query are also weighted
  - …

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 $\in \mathbb{R}^{|V|}$

Summary - Ranking

$$match\_score(doc, query) = \sum_{term \in query \cap doc} score(term)$$

$$match\_rank(doc, query) = index\ in\ reverse(sorted([s\ for\ d\ in\ corpus\ for\ s\ in\ match\_score(d, q)]))$$

score term:

boolean_score$(doc, term) = 1\ if\ term\ in\ doc\ else\ 0$

tf_score$(doc, term) = count(term\ in\ sore)\ if\ term\ in\ doc\ else\ 0$

log_tf_score$(doc, term) = \log\big(1 + count(term\ in\ sore)\big)\ if\ term\ in\ doc\ else\ 0$

tf−idf_score$(doc, term) = tf\ *_{score(doc,term)} \cdot\ idf(doc, term)\ if\ term\ in\ doc\ else\ 0$

idf$(term) = \log\left(\frac{N}{df(term)}\right)$ $df(term) = count(term\ in\ doc\ for\ doc\ in\ corpus)$

* Could be either $tf_{score(doc,term)}$ *or* log_tf_score$(doc, term)$

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: tens of millions of dimensions when you apply this to a web search engine
- These are very sparse vectors - most entries are zero.

Queries as vectors

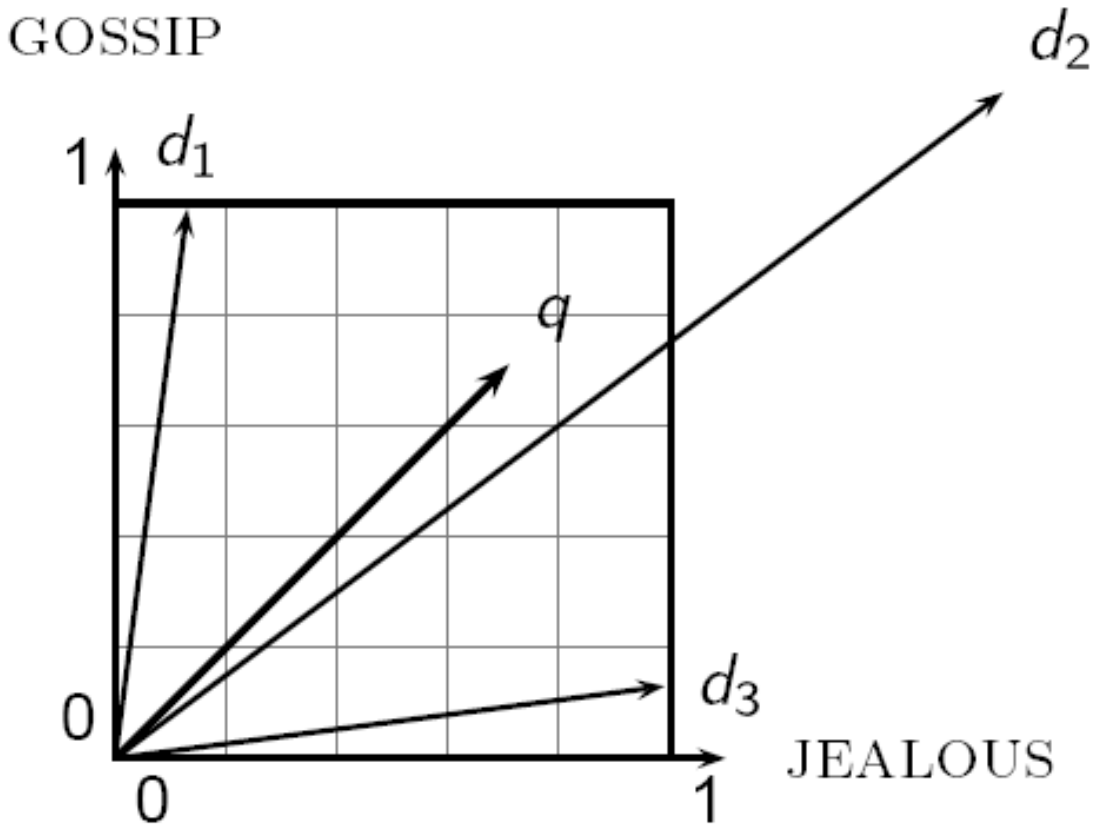
- Key idea 1: Do the same for queries: represent them as vectors in the space
- Key idea 2: Rank documents according to their proximity to the query in this space
- proximity = similarity of vectors
- proximity ≈ inverse of distance

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.



Euclidean Distance example

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

p = 2, Euclidean Distance

- Observation_1: [1, 7, 9]
- Observation_2: [11, 21, 4]

Euclidean Distance example

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

p = 2, Euclidean Distance

- Observation_1: [1, 7, 9]
- Observation_2: [11, 21, 4]
- $(1–11)^2 + (7–21)^2 + (9–4)^2 = 100 + 196 + 25 = 321$
- The square root of 321 ~ 17.91
- The distance between these two observations is 17.91!

Manhattan Distance example

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p} = \sum_{i=1}^{n} |x_i - y_i|$$

p = 1, Manhattan Distance

- Observation_1: [1, 7, 9]
- Observation_2: [11, 21, 4]

Manhattan Distance example

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p} = \sum_{i=1}^{n} |x_i - y_i|$$

p = 1, Manhattan Distance

- Observation_1: [1, 7, 9]
- Observation_2: [11, 21, 4]
- |1–11| + |7–21| + |9–4| = 10 + 14 + 5 = 29
- The distance between the two observations is 29 now!

Chebyshev Distance example

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p} = \max(|x_i - y_i|)$$

p = ∞, Chebyshev Distance

- Observation_1: [1, 7, 9]
- Observation_2: [11, 21, 4]

Chebyshev Distance example

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p} = \max(|x_i - y_i|)$$

p = ∞, Chebyshev Distance

- Observation_1: [1, 7, 9]
- Observation_2: [11, 21, 4]
- max(|1–11|,|7–21|,|9–4|)= max(10, 14, 5) =14
- The distance between the two observations is 14 now!

Use angle instead of distance

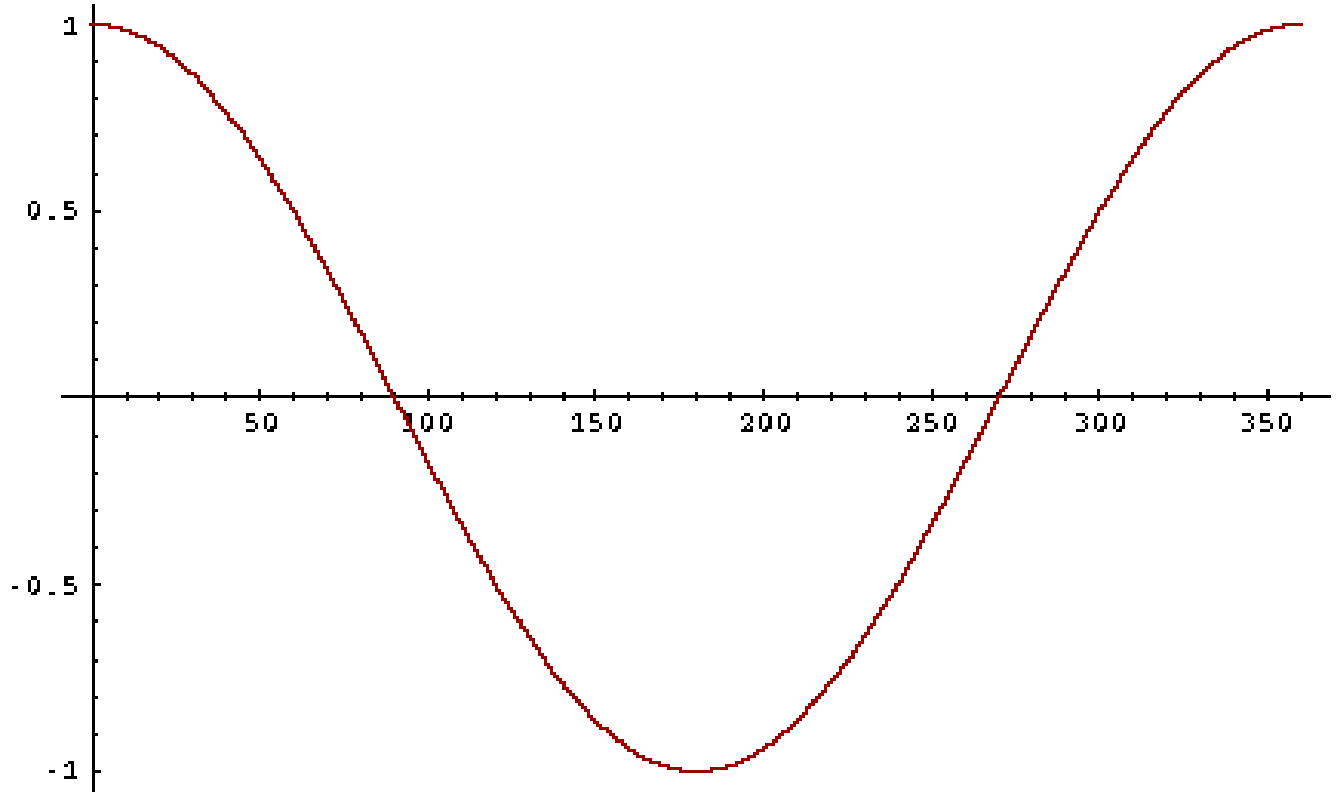
- Thought experiment: take a document *d* and append it to itself. Call this document *d*'.
- "Semantically" d and d' have the same content
- The Euclidean distance between the two documents can be quite large
- The angle between the two documents is 0, corresponding to maximal similarity.

- Key idea: Rank documents according to angle with query.

From angles to cosines

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

From angles to cosines



- But how should we be computing cosines?

Length normalization

- A vector can be (length-) normalized by dividing each of its components by its length – for this we use the L_2 norm:

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

- Dividing a vector by its L_2 norm makes it a unit (length) vector (on surface of unit hypersphere)
- Effect on the two documents d and d' (d appended to itself) from earlier slide: they have identical vectors after length-normalization.
  - Long and short documents now have comparable weights

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 weight of term i in the query
d_i is the 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 $\vec{d}$.

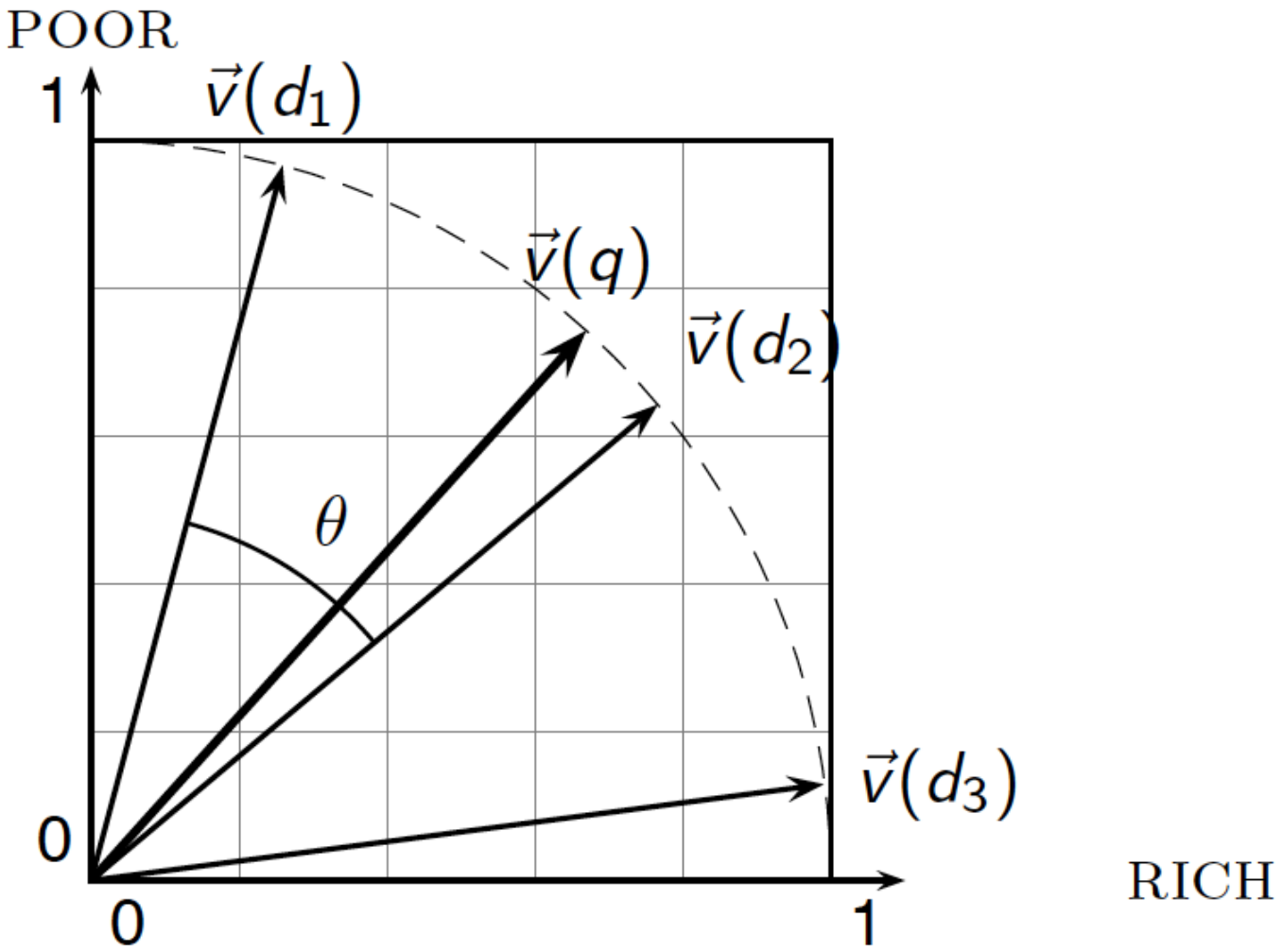
Cosine for length-normalized vectors

- For length-normalized vectors, cosine similarity is simply the dot product (or scalar product):

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

for q, d length-normalized.

Cosine similarity illustrated



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)

Note: To simplify this example, we don't do idf weighting.

3 documents example contd.

Log frequency weighting

term	SaS	PaP	WH
affection	3.06	2.76	2.30
jealous	2.00	1.85	2.04
gossip	1.30	0	1.78
wuthering	0	0	2.58

After length normalization

term	SaS	PaP	WH
affection	0.789	0.832	0.524
jealous	0.515	0.555	0.465
gossip	0.335	0	0.405
wuthering	0	0	0.588

dot(SaS,PaP) ≈ 12.1
dot(SaS,WH) ≈ 13.4
dot(PaP,WH) ≈ 10.1

cos(SaS,PaP) ≈ 0.94
cos(SaS,WH) ≈ 0.79
cos(PaP,WH) ≈ 0.69

Cosine Similarity Example

Document Term Frequency – for each term we count the number of occurrences of the term in the document

Document	team	coach	hockey	baseball	soccer	penalty	score	win	loss	season
Doc1	5	0	3	0	2	0	0	2	0	0
Doc2	3	0	2	0	1	1	0	1	0	1
Doc3	0	7	0	2	1	0	0	3	0	0
Doc4	0	1	0	0	1	2	2	0	3	0

Cosine Similarity Example - Solution

- Denote the first two term-frequency vectors as $\vec{x}, \vec{y}$
- $\vec{x} = (5,0,3,0,2,0,0,2,0,0)$
- $\vec{y} = (3,0,2,0,1,1,0,1,0,1)$

$$\text{Sim}(\vec{x}, \vec{y}) = \frac{\vec{x}^T \cdot \vec{y}}{\|\vec{x}\| \cdot \|\vec{y}\|}$$

Exercise: Calculate the cosine similarity.

- Assume normalization with L_2
- $\|\vec{x}\| = \sqrt{5^2 + 0^2 + 3^2 + 0^2 + 2^2 + 0^2 + 0^2 + 2^2 + 0^2 + 0^2} = 6.48$
- $\|\vec{y}\| = \sqrt{3^2 + 0^2 + 2^2 + 0^2 + 1^2 + 1^2 + 0^2 + 1^2 + 0^2 + 1^2} = 4.12$
- $\vec{x}^T \cdot \vec{y} = 5 \cdot 3 + 0 \cdot 0 + 3 \cdot 2 + 0 \cdot 0 + 2 \cdot 1 + 0 \cdot 1 + 0 \cdot 0 + 2 \cdot 1 + 0 \cdot 0 + 0 \cdot 1 = 25$

$$\text{Sim}(\vec{x}, \vec{y}) = \frac{\vec{x}^T \cdot \vec{y}}{\|\vec{x}\| \cdot \|\vec{y}\|} = \frac{25}{6.48 \cdot 4.12} = 0.94$$

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} × 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[]
```

Computing cosine scores

- Previous algorithm scores term-at-a-time (TAAT)
- Algorithm can be adapted to scoring document-at-a-time (DAAT)
- Storing $w_{t,d}$ in each posting could be expensive
  - …because we'd have to store a floating point number
  - For tf-idf scoring, it suffices to store $tf_{t,d}$ in the posting and idf_t in the head of the postings list
- Extracting the top K items can be done with a priority queue (e.g., a heap)

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)	$1/CharLength^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$				

Weighting may differ in queries vs documents

- Many search engines allow for different weightings for queries vs. documents
- SMART Notation: denotes the combination in use in an engine, with the notation *ddd.qqq,* using the acronyms from the previous table
- A very standard weighting scheme is: lnc.ltc
- Document: logarithmic tf (l as first character), no idf and cosine normalization
- Query: logarithmic tf (l in leftmost column), idf (t in second column), cosine normalization …

tf-idf example: lnc.ltc

Document: *car insurance auto insurance*
Query: *best car insurance*

Term	Query						Document				Prod
	tf-raw	tf-wt	df	idf	wt	n'lize	tf-raw	tf-wt	wt	n'lize	
auto	0	0	5000	2.3	0	0	1	1	1	0.52	0
best	1	1	50000	1.3	1.3	0.34	0	0	0	0	0
car	1	1	10000	2.0	2.0	0.52	1	1	1	0.52	0.27
insurance	1	1	1000	3.0	3.0	0.78	2	1.3	1.3	0.68	0.53

Doc length $= \sqrt{1^2 + 0^2 + 1^2 + 1.3^2} \approx 1.92$

Score = 0+0+0.27+0.53 = 0.8

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

Resources for today's lecture

- IIR 6.2 – 6.4.3

- http://www.miislita.com/information-retrieval-tutorial/cosine-similarity-tutorial.html
  - Term weighting and cosine similarity tutorial for SEO folk!

Introduction to

Information Retrieval

Probabilistic Information Retrieval

Christopher Manning and Pandu Nayak

From Boolean to Ranked Retrieval

1. Why ranked retrieval?
2. Introduction to the classical probabilistic retrieval model and the probability ranking principle
3. The Binary Independence Model: BIM
4. Relevance feedback, briefly
5. The vector space model (VSM) (quick cameo)
6. BM25 model
7. Ranking with features: BM25F (if time allows …)

1. Ranked retrieval

- Thus far, our queries have all been Boolean
  - Documents either match or don't
- Can be good for expert users with precise understanding of their needs and the collection
  - Can also be good for applications: Applications can easily consume 1000s of results
- Not good for the majority of users
  - Most users incapable 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: "*standard user dlink 650*" → 200,000 hits
- Query 2: "*standard user dlink 650 no card found*": 0 hits
- It takes a lot of skill to come up with a query that produces a manageable number of hits
  - AND gives too few; OR gives too many
- Suggested solution:
  - Rank documents by goodness – a sort of clever "soft AND"

Term frequency and the VSM

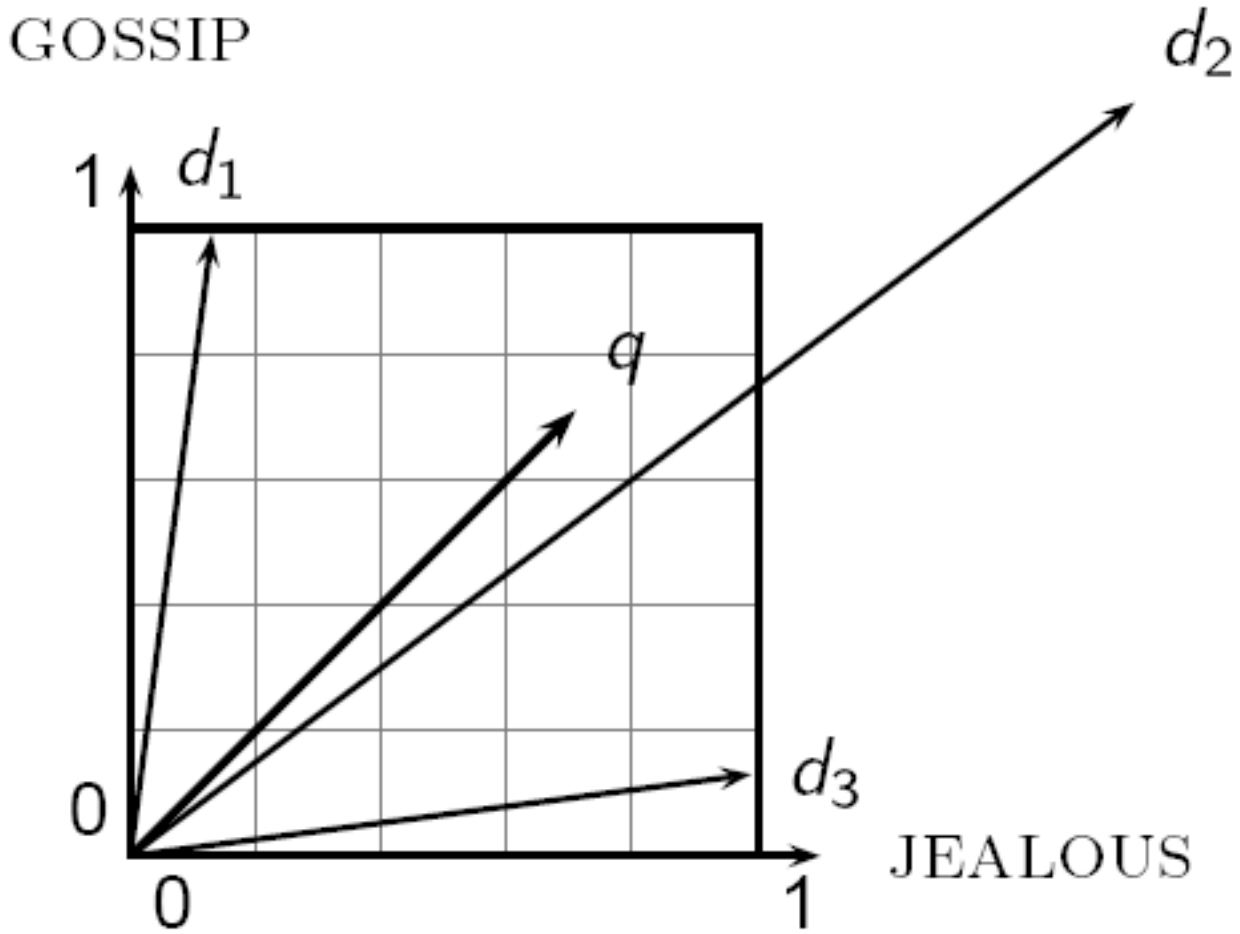
- Right in the first lecture, we said that a page should rank higher if it mentions a word more
  - Perhaps modulated by things like page length
- Why not in BIM? Much of early IR was designed for titles or abstracts, and not for modern full text search
- We now want a model with term frequency in it
- We'll mainly look at a probabilistic model (BM25)
- First, a quick summary of vector space model

Summary – vector space ranking (ch. 6)

- Represent the query as a weighted term frequency/inverse document frequency (tf-idf) vector
  - (0, 0, 0, 0, 2.3, 0, 0, 0, 1.78, 0, 0, 0, …, 0, 8.17, 0, 0)
- Represent each document as a weighted tf-idf vector
  - (1.2, 0, 3.7, 1.5, 2.0, 0, 1.3, 0, 3.7, 1.4, 0, 0, …, 3.5, 5.1, 0, 0)

- 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

Cosine similarity



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)	$1/CharLength^{\alpha}$, $\alpha<1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$				

Okapi BM25 [Robertson et al. 1994, TREC City U.]

- BM25 “Best Match 25” (they had a bunch of tries!)
  - Developed in the context of the Okapi system
  - Started to be increasingly adopted by other teams during the TREC competitions
  - It works well

- Goal: be sensitive to term frequency and document length while not adding too many parameters
  - (Robertson and Zaragoza 2009; Spärck Jones et al. 2000)

“Early” versions of BM25

- Version 1: using the saturation function

$$c_i^{BM25v1}(tf_i) = c_i^{BIM}\frac{tf_i}{k_1 + tf_i}$$

- Version 2: BIM simplification to IDF

$$c_i^{BM25v2}(tf_i) = \log\frac{N}{df_i}\square\frac{(k_1+1)tf_i}{k_1 + tf_i}$$

  - (k_1+1) factor doesn’t change ranking, but makes term score 1 when $tf_i = 1$
  - Similar to *tf-idf*, but term scores are bounded

Document length normalization

- Longer documents are likely to have larger tf_i values

- Why might documents be longer?
  - Verbosity: suggests observed tf_i too high
  - Larger scope: suggests observed tf_i may be right

- A real document collection probably has both effects
- … so should apply some kind of partial normalization

Document length normalization

- Document length:

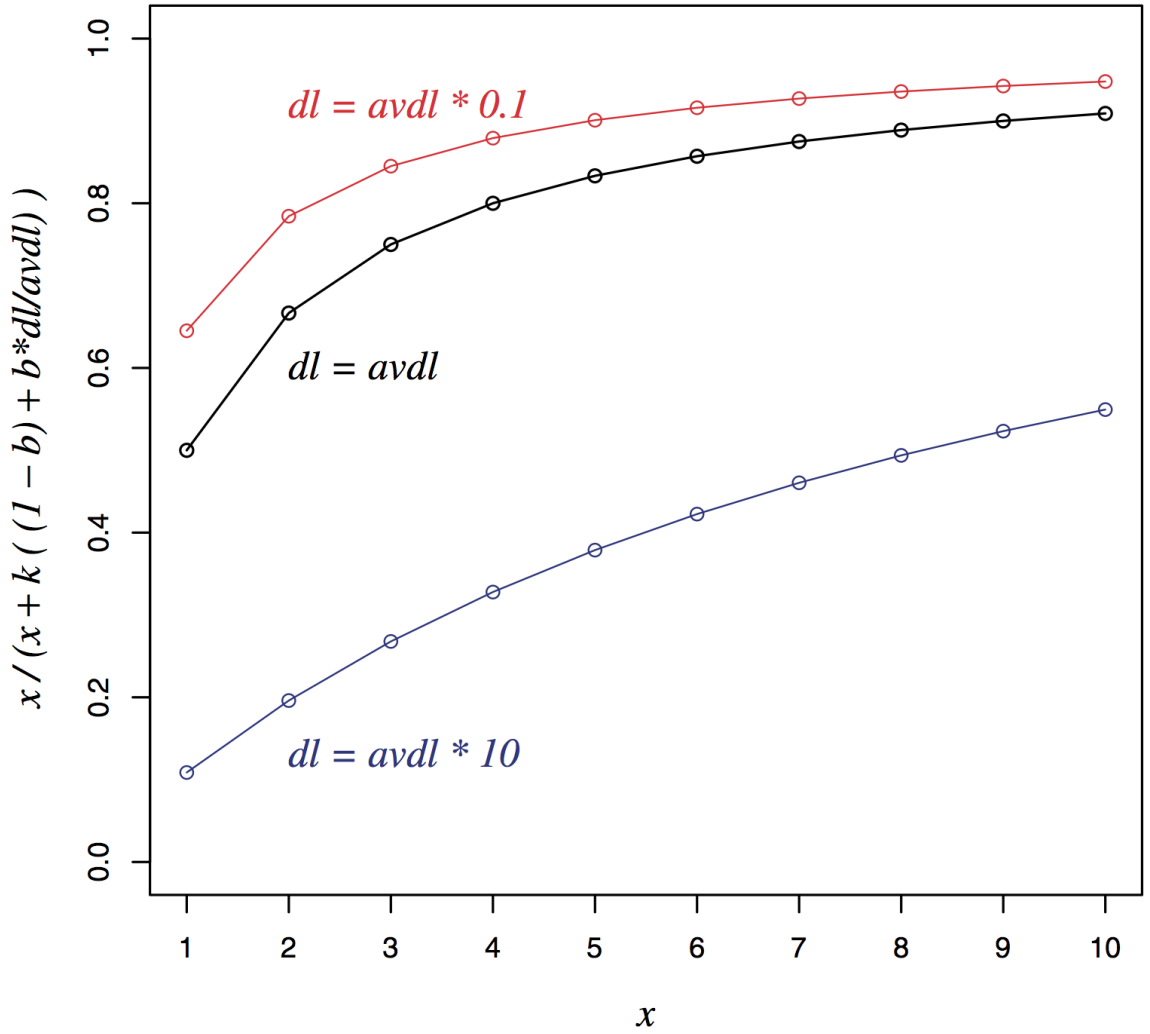
$$dl = \sum_{i \in V} tf_i$$

- $avdl$: Average document length over collection
- Length normalization component

$$B = \left((1-b) + b\frac{dl}{avdl}\right), \qquad 0 \le b \le 1$$

  - $b = 1$ full document length normalization
  - $b = 0$ no document length normalization

Document length normalization



Okapi BM25

- Normalize *tf* using document length

$$tf_i' = \frac{tf_i}{B}$$

$$c_i^{BM25}(tf_i) = \log \frac{N}{df_i} \square \frac{(k_1+1)tf_i'}{k_1 + tf_i'}$$

$$= \log \frac{N}{df_i} \square \frac{(k_1+1)tf_i}{k_1((1-b)+b\frac{dl}{avdl})+tf_i}$$

- BM25 ranking function

$$RSV^{BM25} = \sum_{i \square q} c_i^{BM25}(tf_i);$$

Okapi BM25

$$RSV^{BM25} = \sum_{i \in q} \log \frac{N}{df_i} \cdot \frac{(k_1 + 1) tf_i}{k_1((1-b) + b\frac{dl}{avdl}) + tf_i}$$

- k_1 controls term frequency scaling
  - $k_1 = 0$ is binary model; k_1 large is raw term frequency
- b controls document length normalization
  - $b = 0$ is no length normalization; $b = 1$ is relative frequency (fully scale by document length)
- Typically, k_1 is set around 1.2–2 and b around 0.75
- *IIR* sec. 11.4.3 discusses incorporating query term weighting and (pseudo) relevance feedback

Why is BM25 better than VSM tf-idf?

- Suppose your query is [machine learning]
- Suppose you have 2 documents with term counts:
  - doc1: learning 1024; machine 1
  - doc2: learning 16; machine 8

- tf-idf: $\log_2$ tf * $\log_2$ (N/df)
  - doc1: 11 * 7 + 1 * 10 = **87**
  - doc2: 5 * 7 + 4 * 10 = **75**
- BM25: $k_1 = 2$
  - doc1: 7 * 3 + 10 * 1 = **31**
  - doc2: 7 * 2.67 + 10 * 2.4 = **42.7**

Introduction to

Information Retrieval

Evaluation

Chris Manning and Pandu Nayak

CS276 – Information Retrieval and Web Search

Early public test Collections (20th C)

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

Typical TREC

Recent datasets: 100s of million web pages (GOV, ClueWeb, ...)

Now we have the basics of a benchmark

- Let’s review some evaluation measures
  - *Precision*
  - *Recall*
  - DCG
  - …

Evaluating an IR system

- Note: **user need** is translated into a **query**
- Relevance is assessed relative to the **user need,** *not* the **query**
- E.g., Information need: *My swimming pool bottom is becoming black and needs to be cleaned*.
- Query: ***pool cleaner***
- Assess whether the doc addresses the underlying need, not whether it has these words

Unranked retrieval evaluation: Precision and Recall – recap from IIR 8/video

- **Binary assessments**

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)

Rank-Based Measures

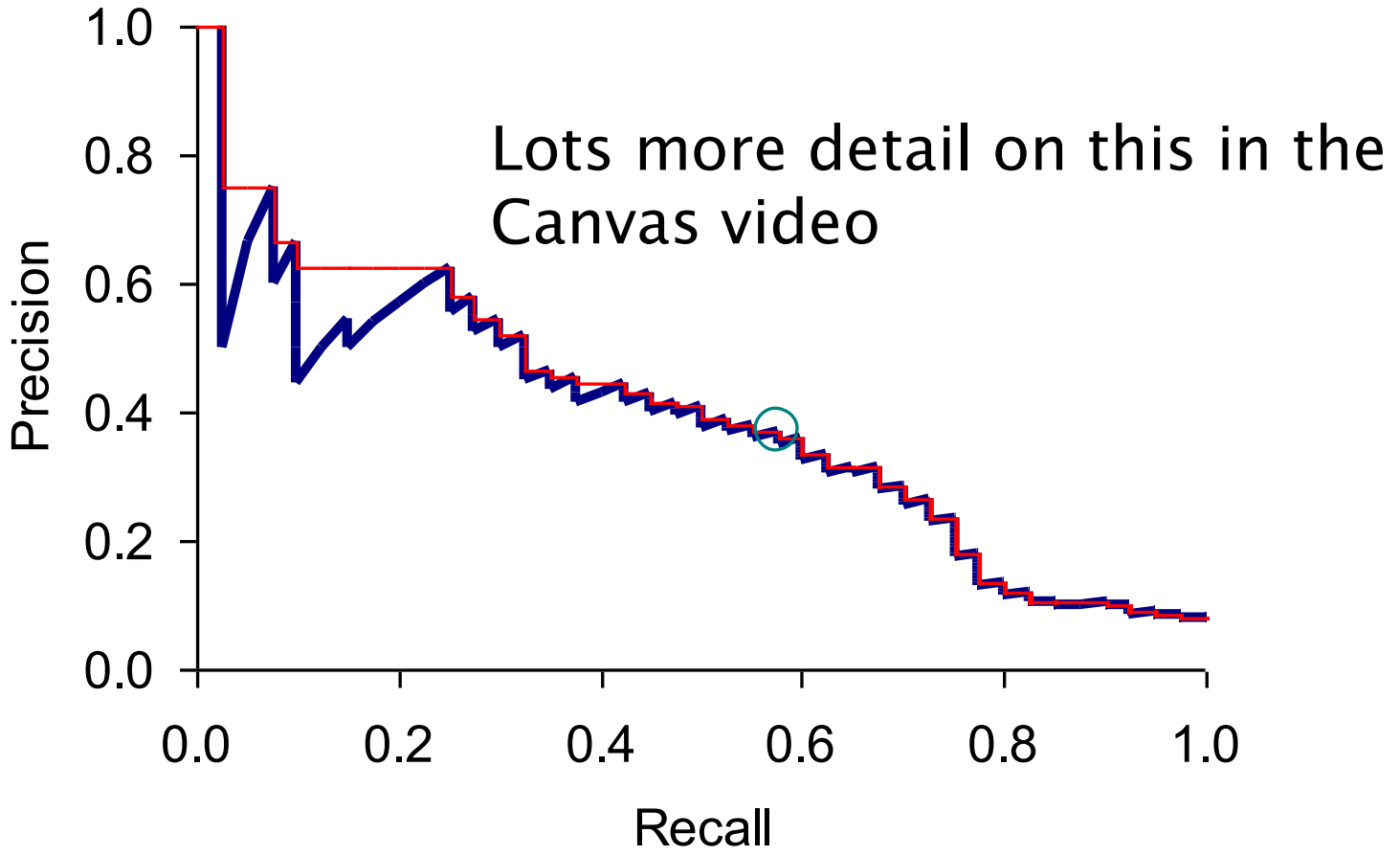
- Binary relevance
  - Precision@K (P@K)
  - Mean Average Precision (MAP)
  - Mean Reciprocal Rank (MRR)

Precision@K

- Set a rank threshold K
- Compute % relevant in top K
- Ignores documents ranked lower than K
- Ex:
  - Prec@3 of 2/3
  - Prec@4 of 2/4
  - Prec@5 of 3/5

- In similar fashion we have Recall@K

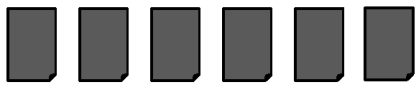
A precision-recall curve



Mean Average Precision

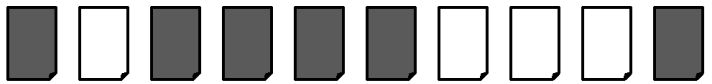
- Consider rank position of each ***relevant*** doc
  - K_1, K_2, … K_R
- Compute Precision@K for each K_1, K_2, … K_R
- Average precision = average of P@K
- Ex:  has AvgPrec of $\frac{1}{3} \cdot \left(\frac{1}{1} + \frac{2}{3} + \frac{3}{5} \right) \approx 0.76$
- MAP is Average Precision across multiple queries/rankings

Average Precision



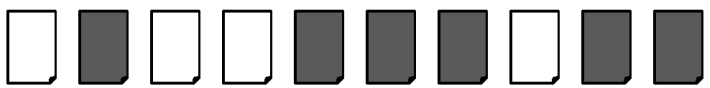
= the relevant documents

Ranking #1



Recall	0.17	0.17	0.33	0.5	0.67	0.83	0.83	0.83	0.83	1.0
Precision	1.0	0.5	0.67	0.75	0.8	0.83	0.71	0.63	0.56	0.6

Ranking #2

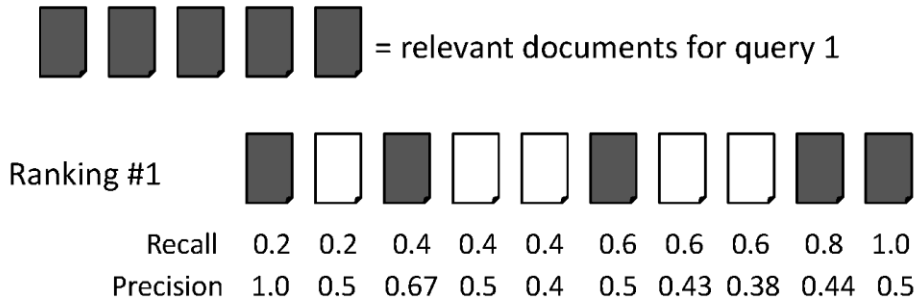


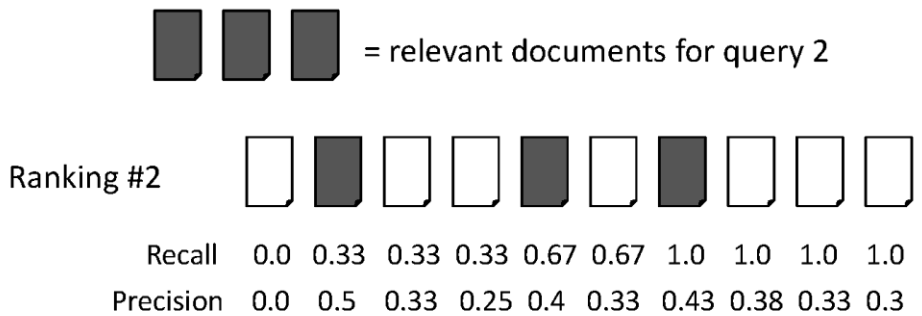
Recall	0.0	0.17	0.17	0.17	0.33	0.5	0.67	0.67	0.83	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.5	0.57	0.5	0.56	0.6

$$\text{Ranking } \#1\text{: } (1.0 + 0.67 + 0.75 + 0.8 + 0.83 + 0.6)/6 = 0.78$$

$$\text{Ranking } \#2\text{: } (0.5 + 0.4 + 0.5 + 0.57 + 0.56 + 0.6)/6 = 0.52$$

MAP



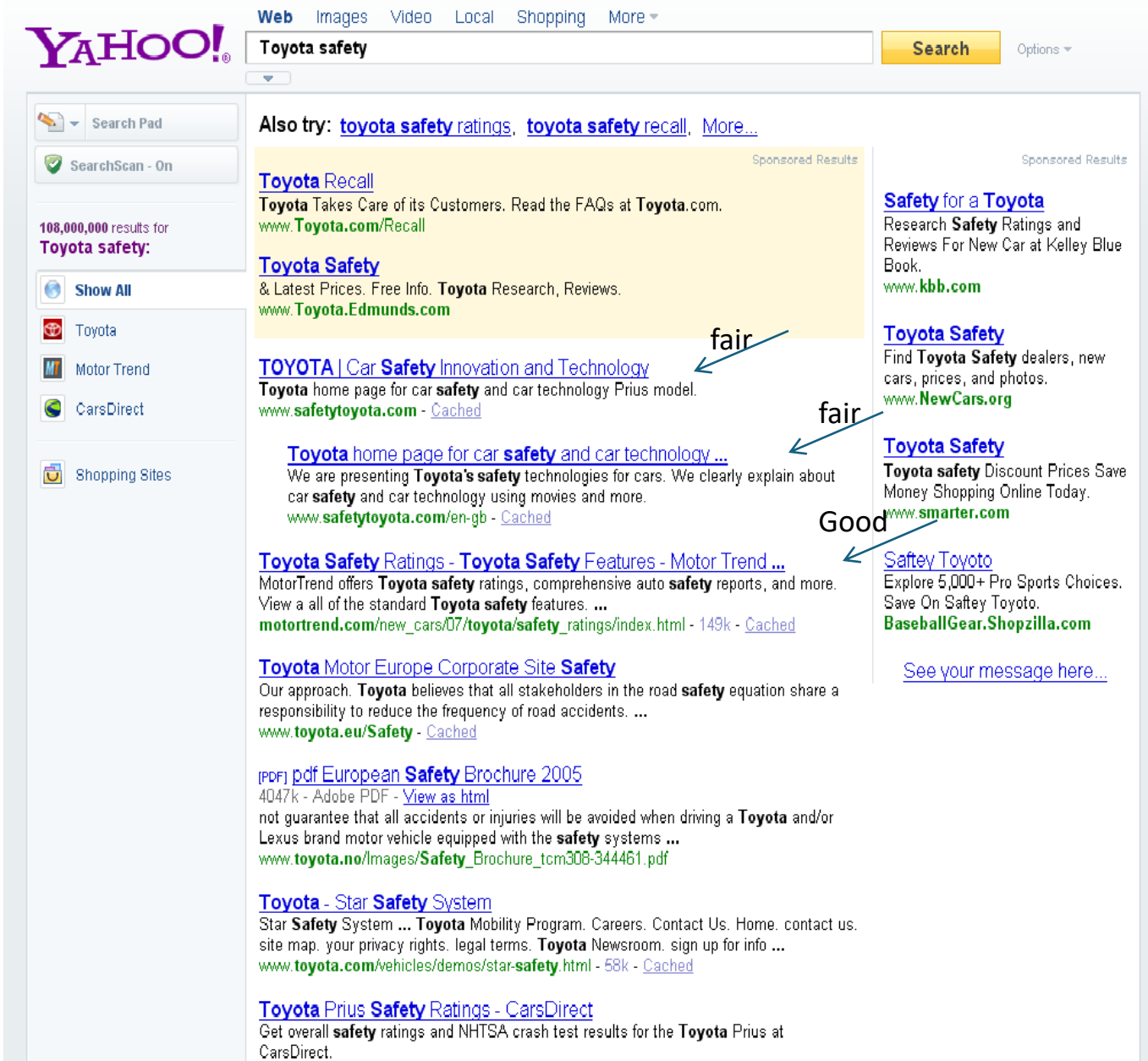


$$average\ precision\ query\ 1 = (1.0 + 0.67 + 0.5 + 0.44 + 0.5)/5 = 0.62$$
$$average\ precision\ query\ 2 = (0.5 + 0.4 + 0.43)/3 = 0.44$$

$$mean\ average\ precision = (0.62 + 0.44)/2 = 0.53$$

Mean average precision

- If a relevant document never gets retrieved, we assume the precision corresponding to that relevant doc to be zero
- MAP is macro-averaging: each query counts equally
- Now perhaps most commonly used measure in research papers
- Good for web search?
- MAP assumes user is interested in finding many relevant documents for each query
- MAP requires many relevance judgments in text collection


YAHOO!
Web Images Video Local Shopping More
Toyota safety
Search
Options
Search Pad
SearchScan - On
108,000,000 results for
Toyota safety:
Show All
Toyota
Motor Trend
CarsDirect
Shopping Sites
Also try: toyota safety ratings, toyota safety recall, More...
Sponsored Results
Toyota Recall
Toyota Takes Care of its Customers. Read the FAQs at Toyota.com.
www.Toyota.com/Recall
Toyota Safety
& Latest Prices. Free Info. Toyota Research, Reviews.
www.Toyota.Edmunds.com
fair
TOYOTA | Car Safety Innovation and Technology
Toyota home page for car safety technology.
www.safetytoyota.com - Cached
fair
Toyota home page for car safety and car technology ...
We are presenting Toyota's safety technologies for cars. We clearly explain about car safety and car technology using movies and more.
www.safetytoyota.com/en-gb - Cached
Good
Toyota Safety Ratings - Toyota Safety Features - Motor Trend ...
MotorTrend offers Toyota safety ratings, comprehensive auto safety reports, and more. View a all of the standard Toyota safety features. ...
motortrend.com/new_cars/07/toyota/safety_ratings/index.html - 149k - Cached
Toyota Motor Europe Corporate Site Safety
Our approach. Toyota believes that all stakeholders in the road safety equation share a responsibility to reduce the frequency of road accidents. ...
www.toyota.eu/Safety - Cached
[PDF] pdf European Safety Brochure 2005
4047k - Adobe PDF - View as html
not guarantee that all accidents or injuries will be avoided when driving a Toyota and/or Lexus brand motor vehicle equipped with the safety systems ...
www.toyota.no/Images/Safety_Brochure_tcm308-344461.pdf
Toyota - Star Safety System
Star Safety System ... Toyota Mobility Program. Careers. Contact Us. Home. contact us. site map. your privacy rights. legal terms. Toyota Newsroom. sign up for info ...
www.toyota.com/vehicles/demos/star-safety.html - 58k - Cached
Toyota Prius Safety Ratings - CarsDirect
Get overall safety ratings and NHTSA crash test results for the Toyota Prius at CarsDirect.
Sponsored Results
Safety for a Toyota
Research Safety Ratings and Reviews For New Car at Kelley Blue Book.
www.kbb.com
Toyota Safety
Find Toyota Safety dealers, new cars, prices, and photos.
www.NewCars.org
Toyota Safety
Toyota safety Discount Prices Save Money Shopping Online Today.
www.smarter.com
Saftey Toyoto
Explore 5,000+ Pro Sports Choices. Save On Saftey Toyoto.
BaseballGear.Shopzilla.com
See your message here...

Discounted Cumulative Gain

- Popular measure for evaluating web search and related tasks

- Two assumptions:
  - Highly relevant documents are more useful than marginally relevant documents
  - the lower the ranked position of a relevant document, the less useful it is for the user, since it is less likely to be examined

Mean Reciprocal Rank

- Consider rank position, K, of first relevant doc
  - Could be – only clicked doc
- Reciprocal Rank score = $\frac{1}{K}$
- MRR is the mean RR across multiple queries

Human judgments are

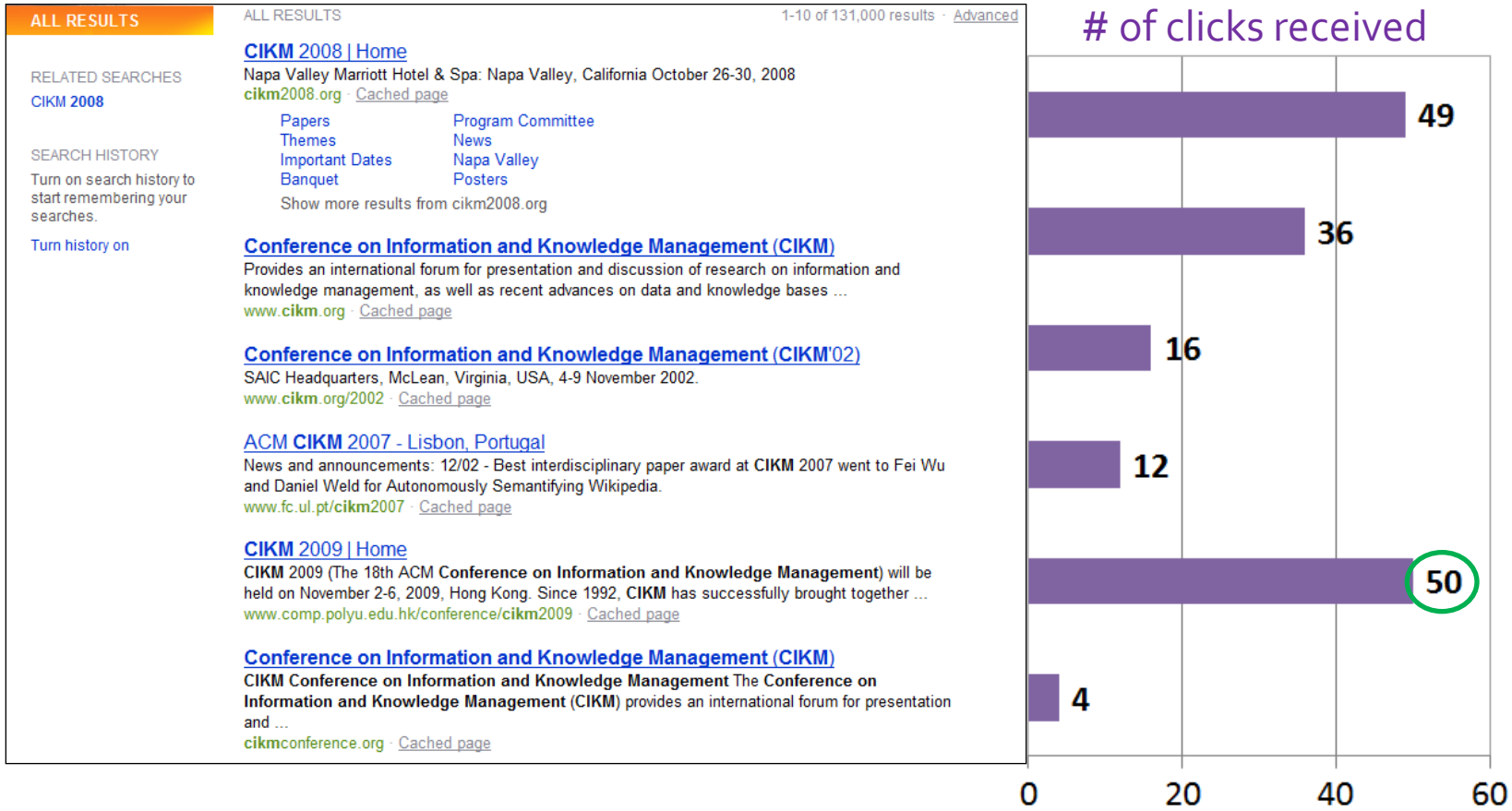
- Expensive
- Inconsistent
  - Between raters
  - Over time
- Decay in value as documents/query mix evolves
- Not always representative of “real users”
  - Rating vis-à-vis query, don’t know underlying need
  - May not understand meaning of terms, etc.
- So – what alternatives do we have?

Using user Clicks

User Behavior

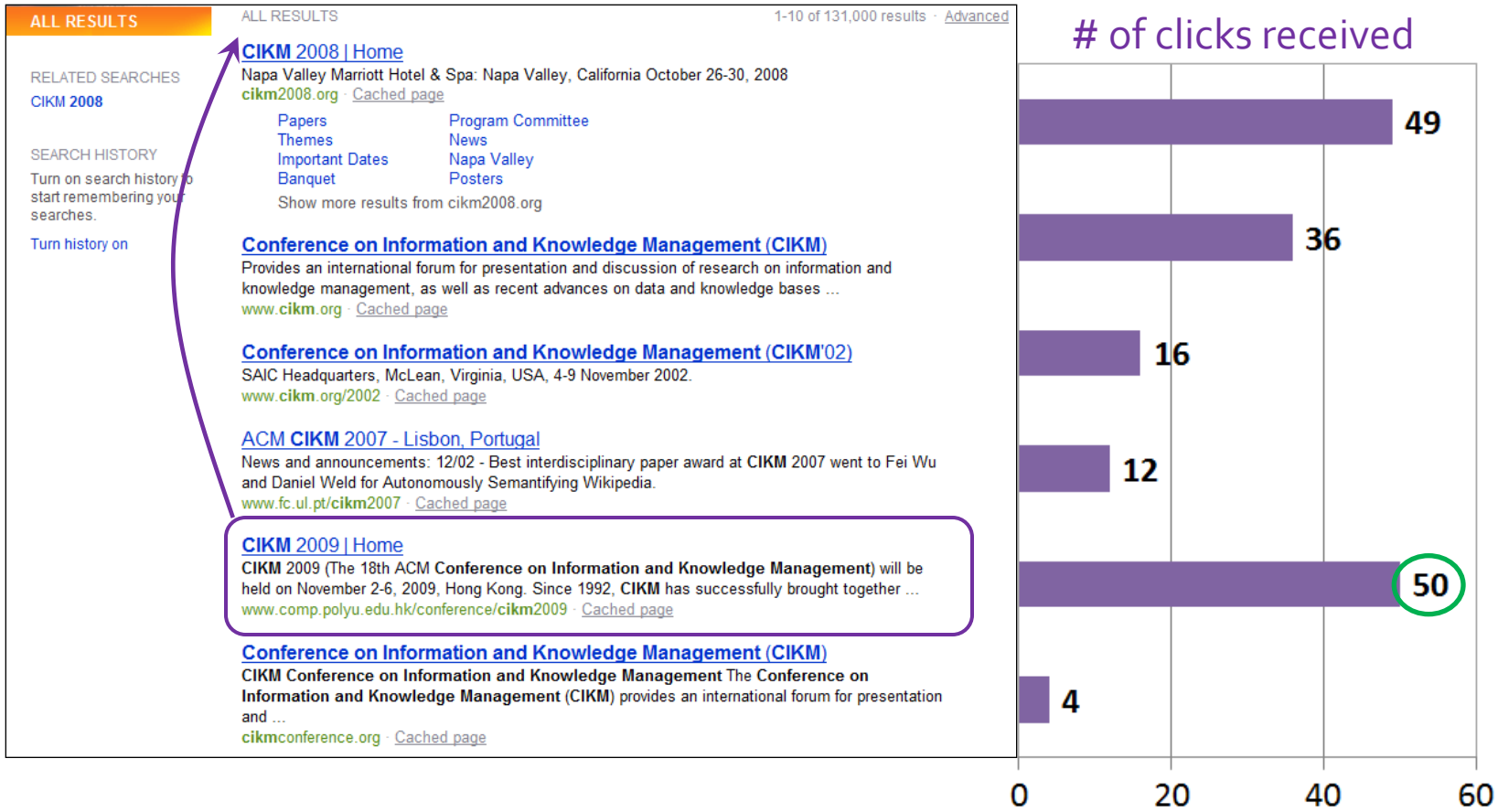
Taken with slight adaptation from Fan Guo and Chao Liu's 2009/2010 CIKM tutorial: Statistical Models for Web Search: Click Log Analysis

- Search Results for "CIKM" (in 2009!)



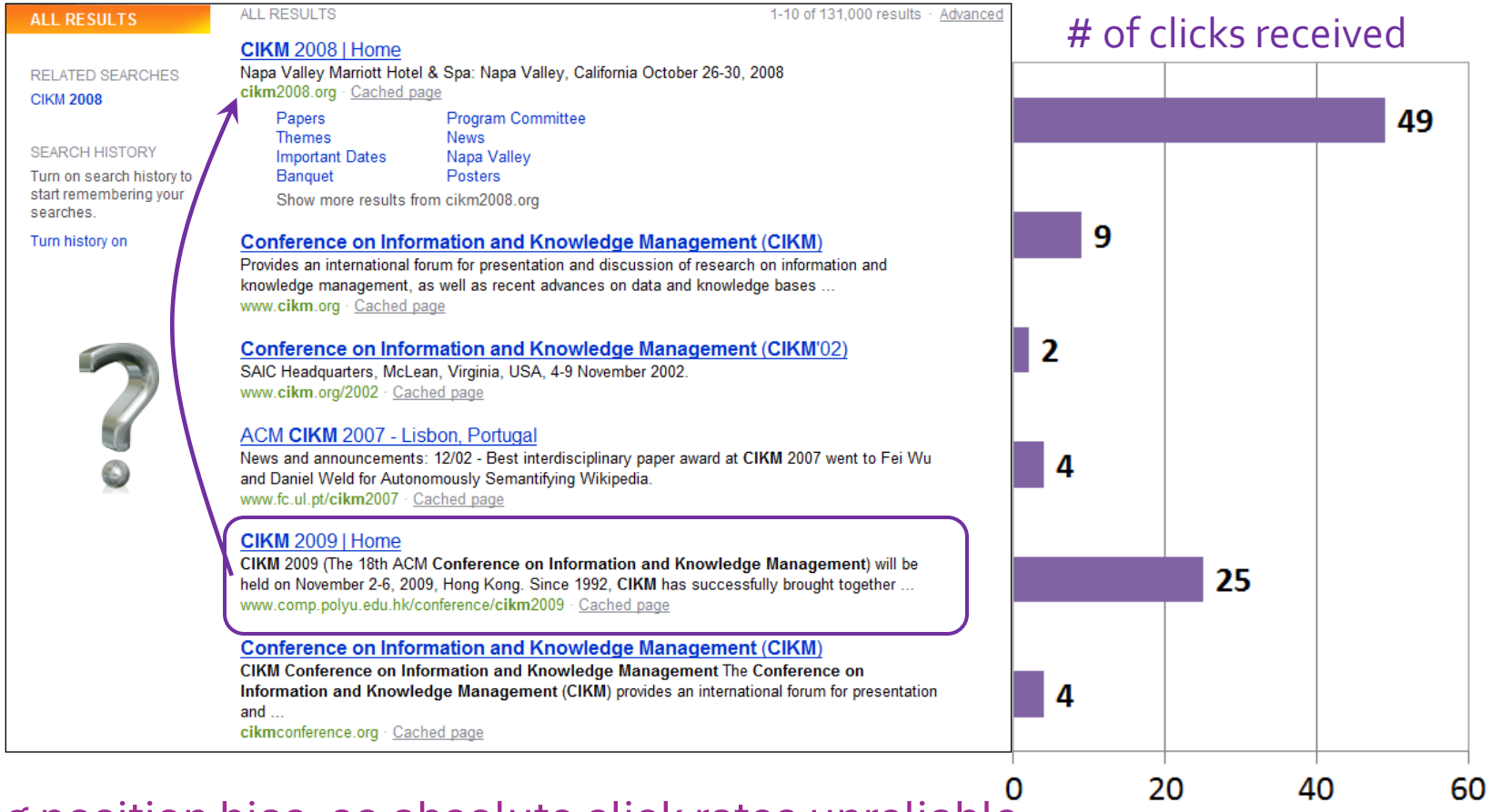
User Behavior

- Adapt ranking to user clicks?



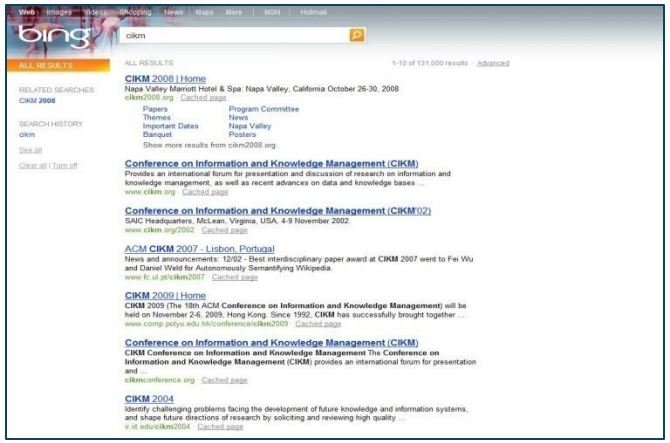
What do clicks tell us?

- Tools needed for non-trivial cases

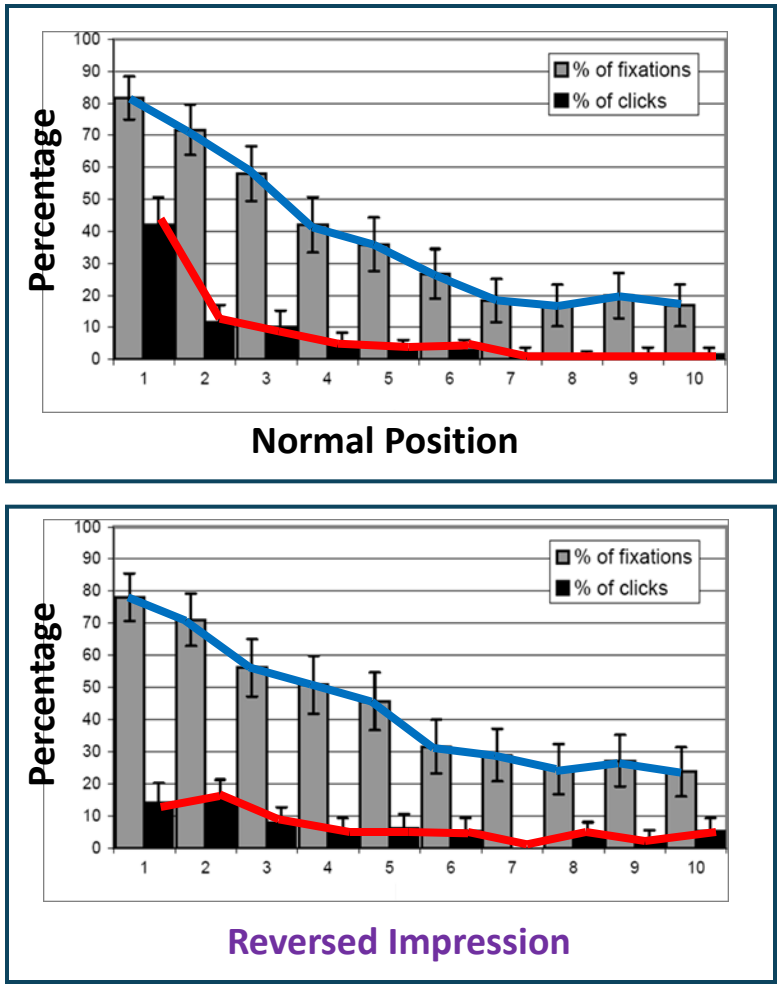


Strong position bias, so absolute click rates unreliable

Eye-tracking User Study



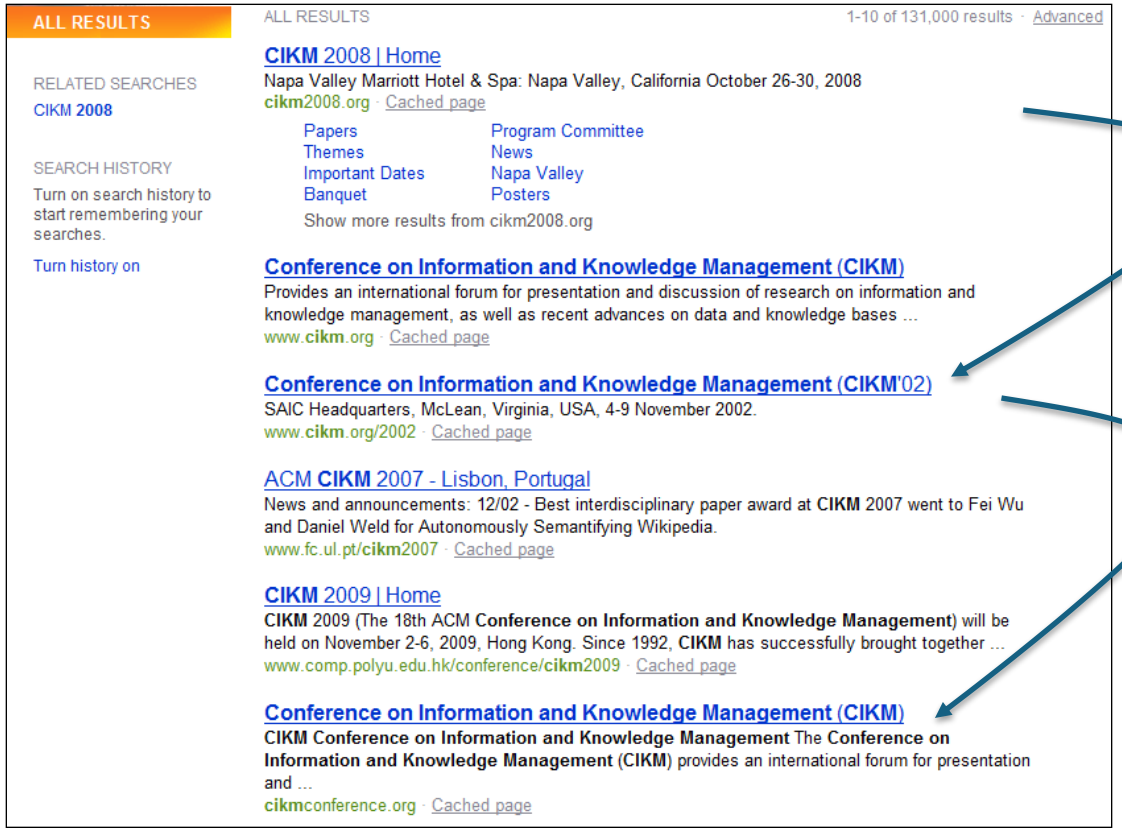
Click Position-bias



- Higher positions receive more user attention (eye fixation) and clicks than lower positions.
- This is true even in the extreme setting where the order of positions is reversed.
- "Clicks are informative but biased".

[Joachims+07]

Relative vs absolute ratings



User's click sequence

Hard to conclude Result1 > Result3
Probably can conclude Result3 > Result2

User behavior

- User behavior is an intriguing source of relevance data
  - Users make (somewhat) informed choices when they interact with search engines
  - Potentially a lot of data available in search logs

- But there are significant caveats
  - User behavior data can be very noisy
  - Interpreting user behavior can be tricky
  - Spam can be a significant problem
  - Not all queries will have user behavior

Until the next time ☺