Evaluating search engines

CE-324: Modern Information Retrieval

Sharif University of Technology

M. Soleymani

Fall 2018

Most slides have been adapted from: Profs. Manning, Nayak & Raghavan (CS-276, Stanford)

Evaluation of a search engine

- How fast does it index?
  - Number of documents/hour
  - Incremental indexing
- How large is its doc collection?
- How fast does it search?
- How expressive is the query language?
- User interface design issues
- This is all good, but it says nothing about the *quality* of its search

User happiness is elusive to measure

- The key utility measure is user happiness.
  - How satisfied is each user with the obtained results?
  - The most common proxy to measure human satisfaction is ***relevance*** of search results to the posed information

- How do you measure relevance?

Why do we need system evaluation?

- How do we know which of the already introduced techniques are effective in which applications?
  - Should we use stop lists? Should we stem? Should we use inverse document frequency weighting?

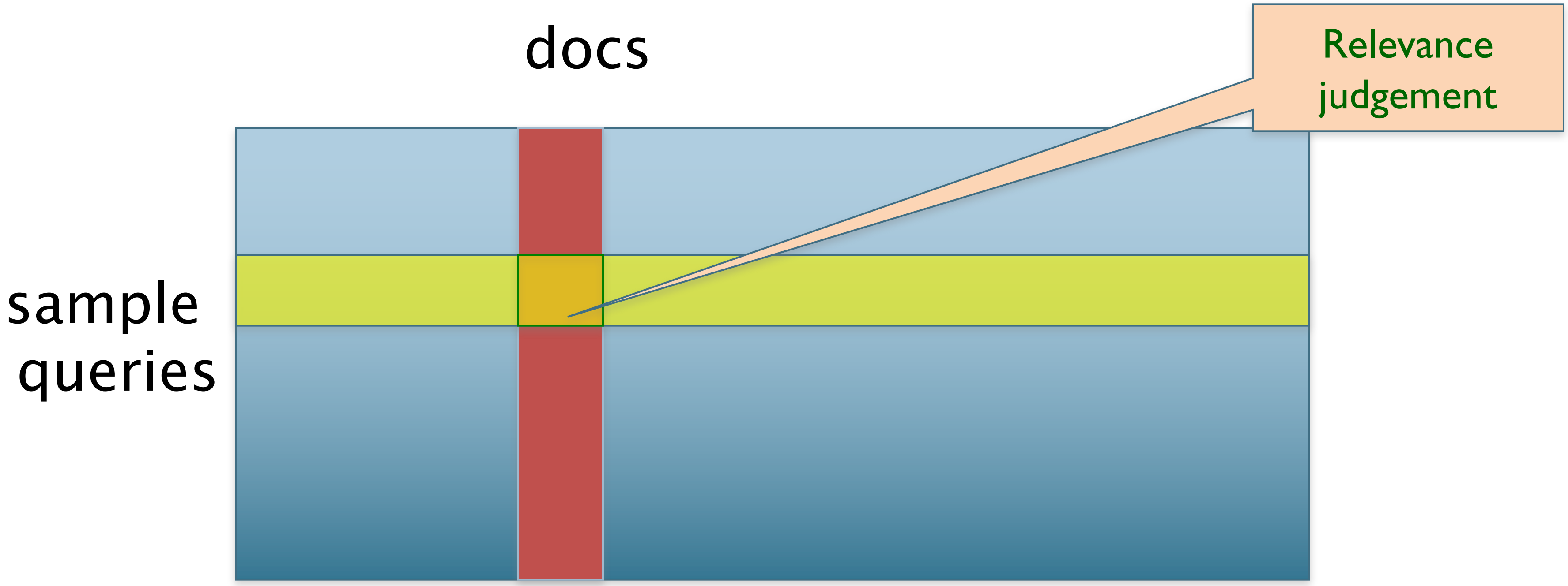
- How can we claim to have built a better search engine for a document collection?

Measuring relevance

- Relevance measurement requires 3 elements:
  1. A benchmark doc collection
  2. A benchmark suite of information needs
  3. A usually binary assessment of either Relevant or Nonrelevant for each information needs and each document
     - Some work on more-than-binary, but not the standard

So you want to measure the quality of a new search algorithm

- Benchmark documents
- Benchmark query suite
- Judgments of document relevance for each query



Relevance judgments

- Binary (relevant vs. non-relevant) in the simplest case, more nuanced (0, 1, 2, 3 …) in others
- What are some issues already?
  - Cost of getting these relevance judjements

Crowd source relevance judgments?

- Present query-document pairs to low-cost labor on online crowd-sourcing platforms
  - Hope that this is cheaper than hiring qualified assessors
- Lots of literature on using crowd-sourcing for such tasks
- Main takeaway – you get some signal, but the variance in the resulting judgments is very high

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., <u>Information need</u>: *My swimming pool bottom is becoming black and needs to be cleaned.*
  - <u>Query</u>: ***pool cleaner***
- Assess whether the doc addresses the underlying need, not whether it has these words

What else?

- Still need test queries
  - Must be germane to docs available
  - Must be representative of actual user needs
  - Random query terms from the documents generally not a good idea
  - Sample from query logs if available

- Classically (non-Web)
  - Low query rates – not enough query logs
  - Experts hand-craft "user needs"

Some public 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

Typical TREC

Standard relevance benchmarks

- TREC: NIST has run a large IR test bed for many years
- Reuters and other benchmark doc collections
- Human experts mark, for each query and for each doc, Relevant or Nonrelevant
  - or at least for subset of docs that some systems (participating in the competitions) returned for that query
- Binary (relevant vs. non-relevant) in the simplest case, more nuanced (0, 1, 2, 3 …) in others

Unranked retrieval evaluation: Precision and Recall

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

	Relevant	Nonrelevant
Retrieved	tp	fp
Not Retrieved	fn	tn

Precision P = tp/(tp + fp)

Recall R = tp/(tp + fn)

Accuracy measure for evaluation?

- **Accuracy:** fraction of classifications that are correct
  - evaluation measure in machine learning classification works
- The **accuracy** of an engine:
  - (tp + tn) / (tp + fp + fn + tn)
- Given a query, an engine classifies each doc as “Relevant” or “Nonrelevant”
- Why is this not a very useful evaluation measure in IR?

Why not just use accuracy?

- How to build a 99.9999% accurate search engine on a low budget….
  - The snoogle search engine below always returns 0 results (“No matching results found”), regardless of the query
  - Since many more non-relevant docs than relevant ones

- People *want to find something* and have a certain tolerance for junk.

Precision/Recall

- Retrieving all docs for all queries!
  - High recall but low precision
- 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

A combined measure: F

- Combined measure: **F measure**
  - allows us to trade off precision against recall
  - weighted harmonic mean of P and R

$$\beta^2 = \frac{1-\alpha}{\alpha}$$

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

- What value range of weights recall higher than precision?

A combined measure: F

- People usually use balanced F (β = 1 or α = ½)

$$F = F_{\beta=1}$$

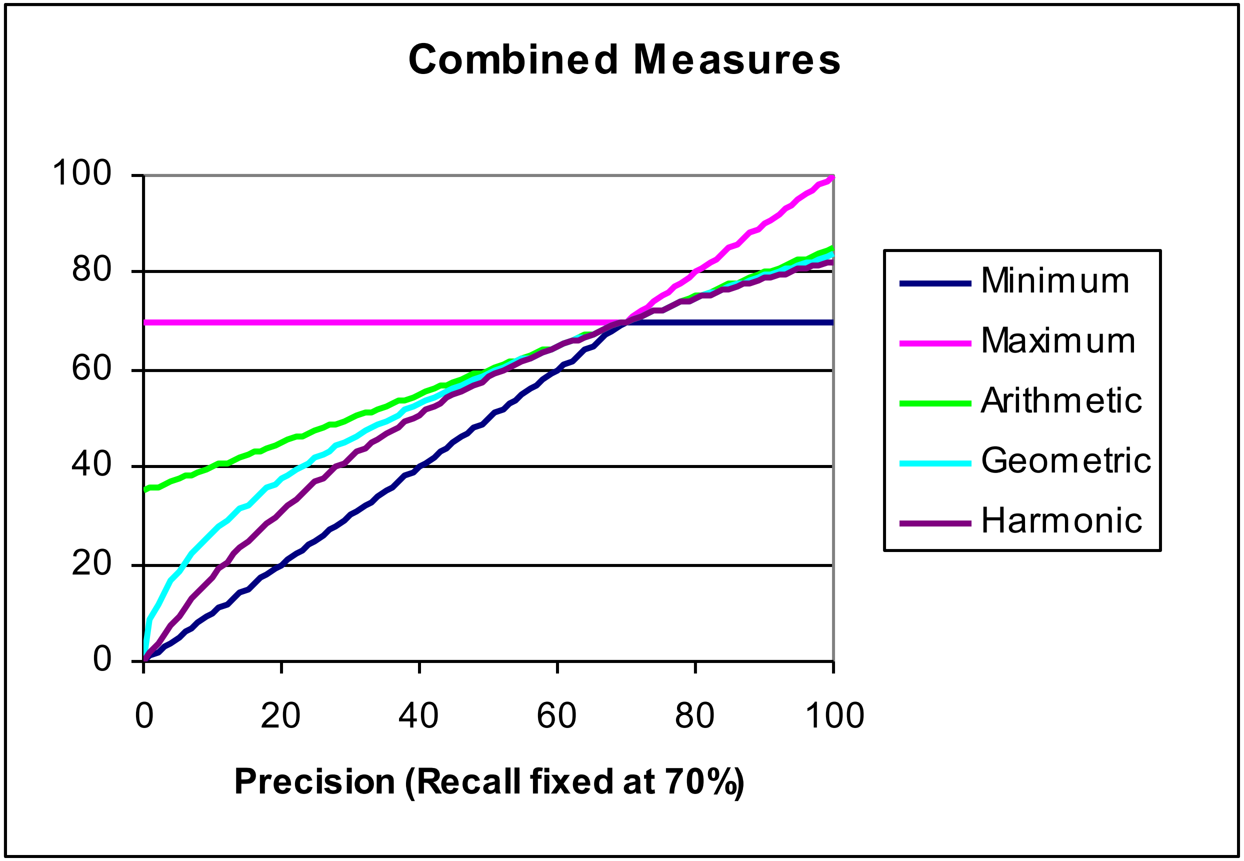
$$F = \frac{2PR}{P + R}$$

- harmonic mean of P and R: $\frac{1}{F} = \frac{1}{2}\left(\frac{1}{P} + \frac{1}{R}\right)$

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.
  - F (harmonic mean) is a kind of smooth minimum.

F_1 and other averages



Harmonic mean is a conservative average

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

Evaluating ranked results

- Precision, recall and F are measures for (unranked) sets.
  - We can easily turn set measures into measures of ranked lists.

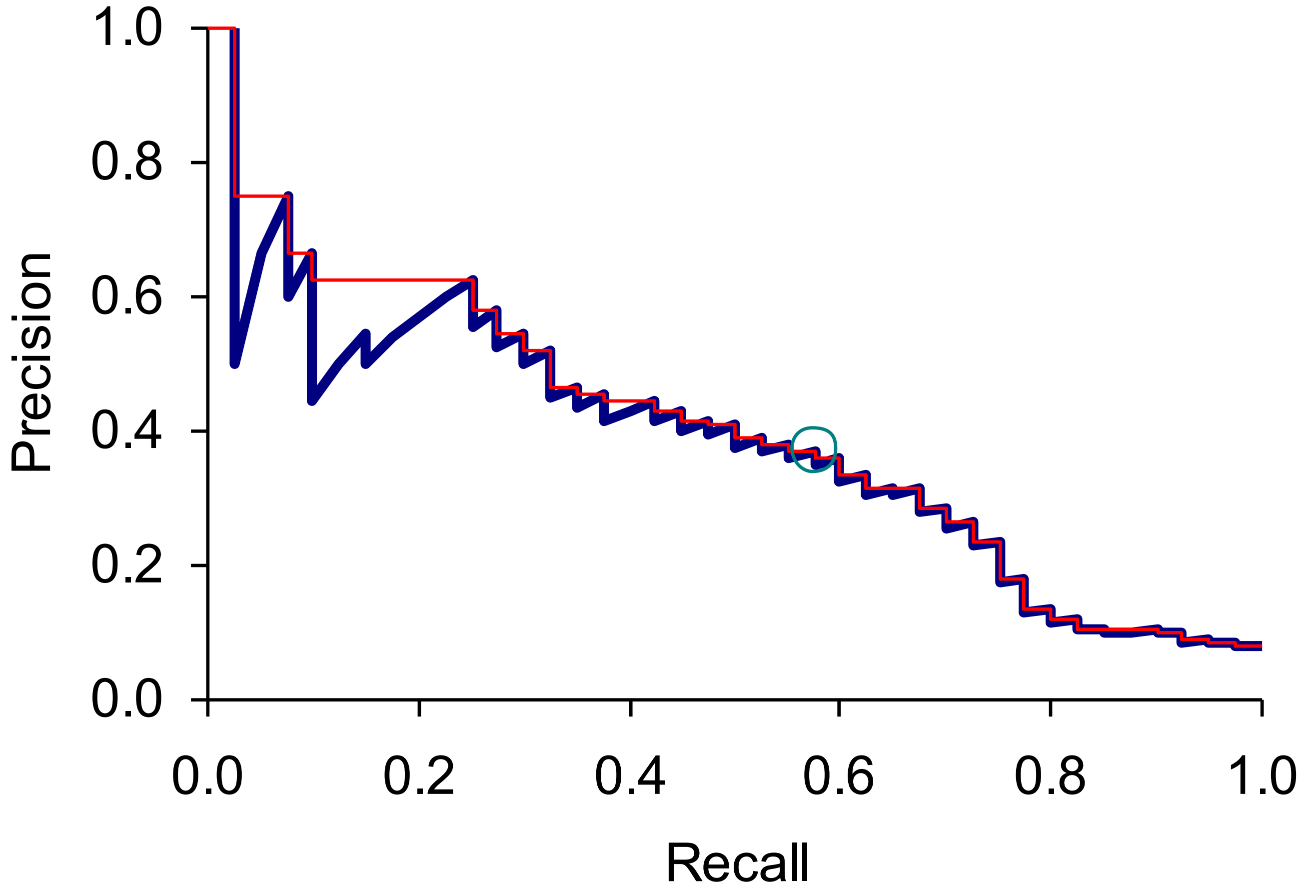
- Evaluation of ranked results:
  - Taking various numbers of top returned docs (recall levels)
    - Sets of retrieved docs are given by the top k retrieved docs.
      - Just compute the set measure for each "prefix": the top 1, top 2, top 3, top 4, and etc results
  - Doing this for precision and recall gives you a ***precision-recall curve***

Rank-Based Measures

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

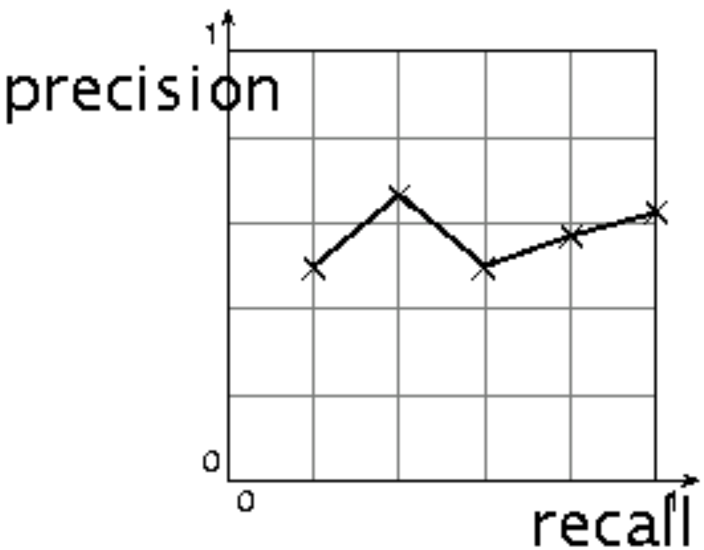
- Multiple levels of relevance
  - Normalized Discounted Cumulative Gain (NDCG)

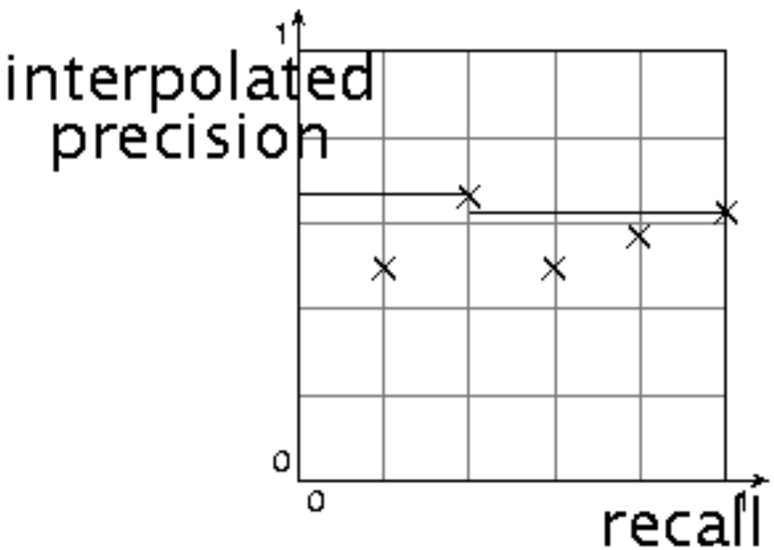
A precision-recall curve



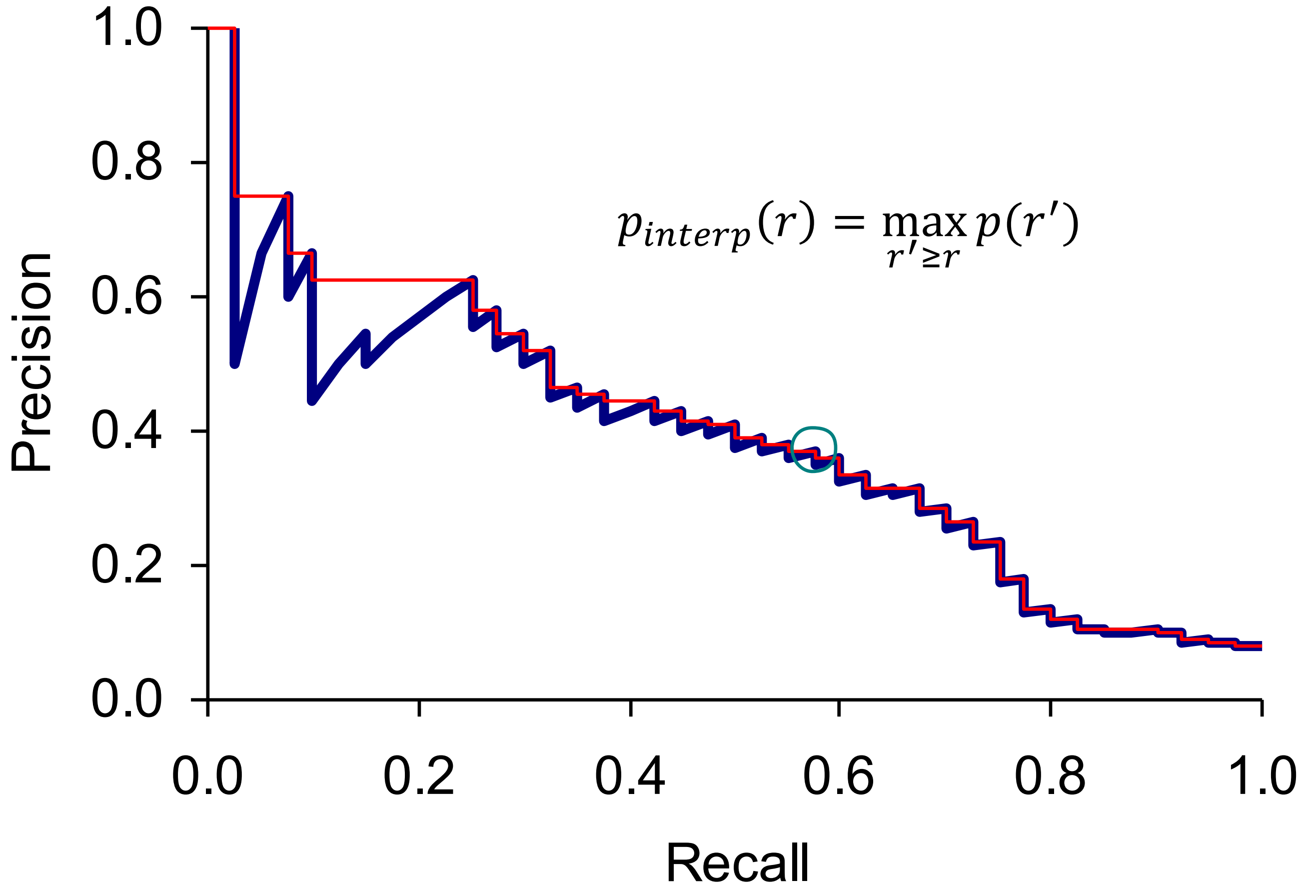
Interpolated precision

- Interpolation: 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.
  - If locally precision increases with increasing recall, then you should get to count that…





An interpolated precision-recall curve



Averaging over queries

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

Binary relevance evaluation

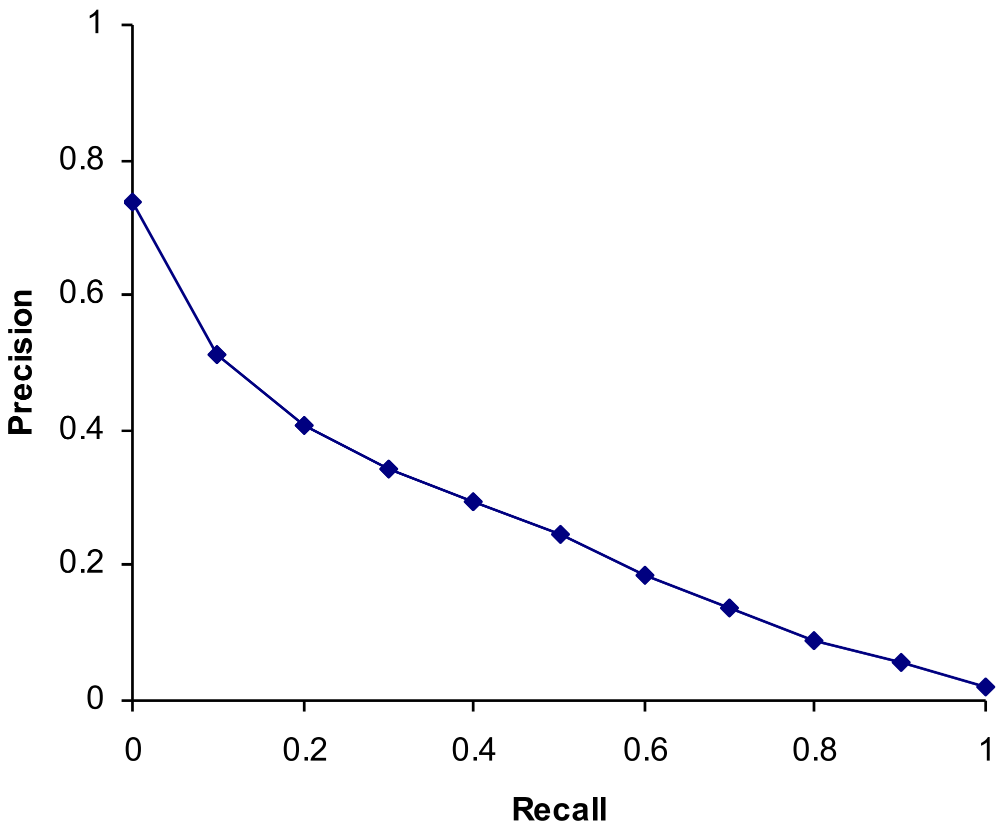
- Graphs are good, but people want summary measures!
  - 11-point interpolated average precision
    - Precision at fixed retrieval level
    - MAP
    - R-precision

11-point interpolated average precision

- The standard measure in the early TREC competitions
- Precision at 11 levels of recall varying from 0 to 1
  - by tenths of the docs using interpolation and average them
- Evaluates performance at all recall levels (0, 0.1, 0.2, …,1)

Typical (good) 11 point precisions

- SabIR/Cornell 8A1
  - 11pt precision from TREC 8 (1999)



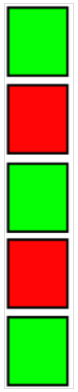
Precision-at-k

- **Precision-at-*k***: Precision of top *k* results
  - Set a rank threshold K
  - Ignores documents ranked lower than K
- Perhaps appropriate for most of web searches
  - people want good matches on the first one or two results pages
- Does not need any estimate of the size of relevant set
  - But: averages badly and has an arbitrary parameter of *k*

Precision-at-k

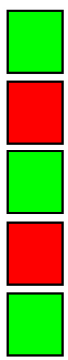
- Compute % relevant in top K

- Examples
  - Prec@3 of 2/3
  - Prec@4 of 2/4
  - Prec@5 of 3/5



- In similar fashion we have Recall@K

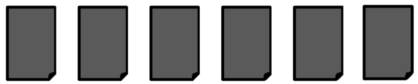
Average precision

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

Mean Average Precision (MAP)

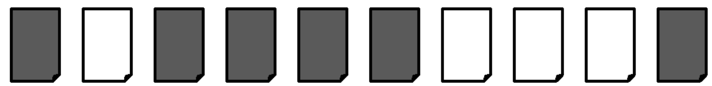
- MAP is Average Precision across multiple queries/rankings
- Mean Average Precision (MAP)
  - Average precision is obtained for the top *k* docs, each time a relevant doc is retrieved
  - MAP for query collection is arithmetic average
    - Macro-averaging: each query counts equally

Average precision: example



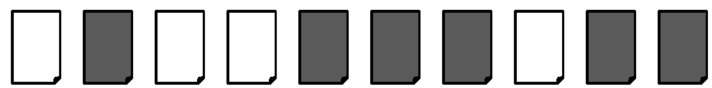
= the relevant documents

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

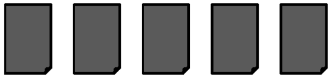


Recall	0.0	0.17	0.17	0.17	0.33	0.5	0.67	0.67	0.83	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.5	0.57	0.5	0.56	0.6

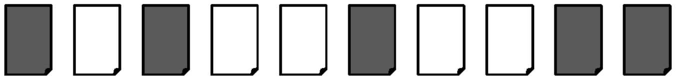
$$\text{Ranking } \#1\text{:}\ \ (1.0 + 0.67 + 0.75 + 0.8 + 0.83 + 0.6)/6 = 0.78$$

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

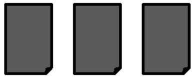
MAP: example



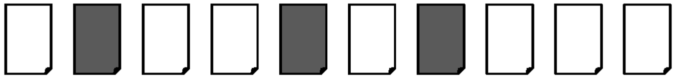
= relevant documents for query 1

Ranking #1 

Recall	0.2	0.2	0.4	0.4	0.4	0.6	0.6	0.6	0.8	1.0
Precision	1.0	0.5	0.67	0.5	0.4	0.5	0.43	0.38	0.44	0.5



= relevant documents for query 2

Ranking #2 

Recall	0.0	0.33	0.33	0.33	0.67	0.67	1.0	1.0	1.0	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.33	0.43	0.38	0.33	0.3

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

$$average\ precision\ query\ 2 = (0.5 + 0.4 + 0.43)/3 = 0.44$$

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

MAP

- Q: set of information needs
- Set of relevant docs to $q_j \in Q$: $d_{j,1}, d_{j,2}, \dots, d_{j,K}$
- R_{jk}: set of ranked retrieval results from the top until reaching $d_{j,k}$

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{i=1}^{m_j} \text{Precision}(R_{kj})$$

MAP

- Now perhaps most commonly used measure in research papers

- Good for web search?
  - MAP assumes user is interested in finding many relevant docs for each query
  - MAP requires many relevance judgments in text collection

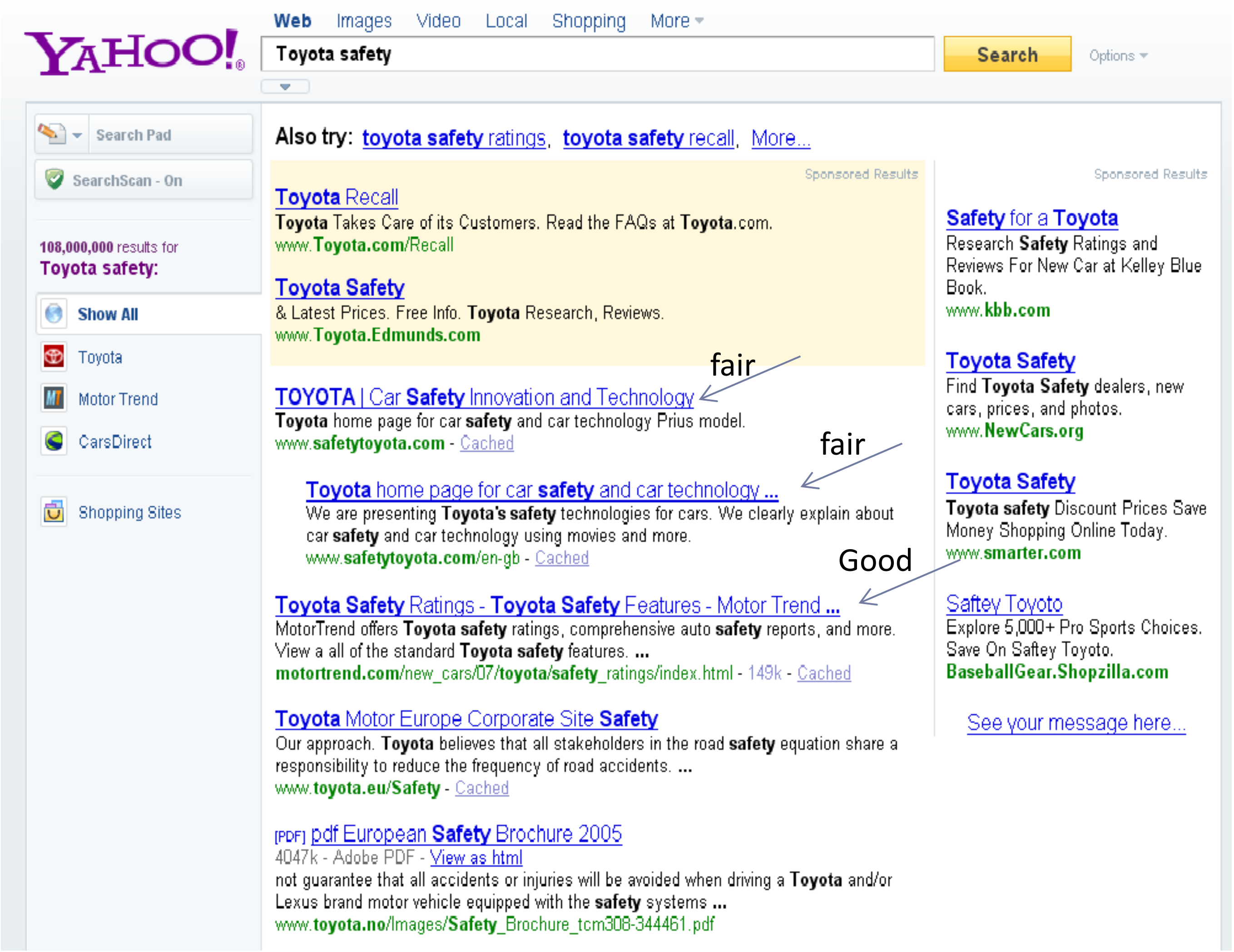
R-precision

- Rel: A known (though perhaps incomplete) set of relevant docs

- Calculate precision of the top $|Rel|$ docs returned
  - r relevant among the top $|Rel|$ results $\Rightarrow$ for this set $P = R = \frac{r}{|Rel|}$

- Perfect system could score 1.0.

Beyond binary relevance



Discounted Cumulative Gain

- Popular measure for evaluating web search and related tasks

- Two assumptions:
  - **Highly relevant** docs are more useful
  - The lower ranked position of a relevant doc, the less useful it is for the user

Discounted Cumulative Gain

- Uses *graded relevance* as a measure of usefulness
  - More than two levels (i.e. relevant and non-relevant)

- Gain is accumulated starting at the top of the ranking and may be reduced, or *discounted*, at lower ranks

- Typical discount is 1/log *(rank)*
  - With base 2, the discount at rank 4 is 1/2, and at rank 8 it is 1/3

Summarize a Ranking: DCG

- Cumulative Gain (CG) at rank n
  - Let the ratings of the n docs be $r_1, r_2, \ldots r_n$ (in ranked order)
  - $CG = r_1+r_2+\ldots r_n$

- Discounted Cumulative Gain (DCG) at rank n
  - $DCG = r_1 + r_2/\log_2 2 + r_3/\log_2 3 + \ldots\ r_n/\log_2 n$
    - We may use any base for the logarithm

Discounted Cumulative Gain

- *DCG* is the total gain accumulated at a particular rank *p*:

$$DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i}$$

- Alternative formulation:

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log(1+i)}$$

  - used by some web search companies
  - emphasis on retrieving highly relevant documents

DCG Example

- 10 ranked documents judged on 0-3 relevance scale:

  3 2 3 0 0 1 2

- discounted gain:

  3 2/1 3/1.59 0 0 1/2.59 2/2.81

  = 3 2 1.89 0 0 0.39 0.71

- DCG:

  3 5 6.89 6.89 6.89 7.28 7.99

Summarize a Ranking: NDCG

- NDCG(q,k) is computed over the k top search results (similar to p@k)

- NDCG normalizes DCG at rank *k* by the DCG value at rank *k* of the ideal ranking
  - Ideal ranking: first returns docs with the highest relevance level, then the next highest relevance level, etc

- Normalization useful for contrasting queries with varying numbers of relevant results

- NDCG is now quite popular in evaluating Web search

NDCG - Example

4 documents: d_1, d_2, d_3, d_4

i	Ground Truth		Ranking Function$_1$		Ranking Function$_2$	
	Document Order	r_i	Document Order	r_i	Document Order	r_i
1	d4	2	d3	2	d3	2
2	d3	2	d4	2	d2	1
3	d2	1	d2	1	d4	2
4	d1	0	d1	0	d1	0
	$NDCG_{GT}$=1.00		$NDCG_{RF1}$=1.00		$NDCG_{RF2}$=0.9203	

$$DCG_{GT} = 2 + \left(\frac{2}{\log_2 2} + \frac{1}{\log_2 3} + \frac{0}{\log_2 4} \right) = 4.6309$$

$$DCG_{RF1} = 2 + \left(\frac{2}{\log_2 2} + \frac{1}{\log_2 3} + \frac{0}{\log_2 4} \right) = 4.6309$$

$$DCG_{RF2} = 2 + \left(\frac{1}{\log_2 2} + \frac{2}{\log_2 3} + \frac{0}{\log_2 4} \right) = 4.2619$$

$$MaxDCG = DCG_{GT} = 4.6309$$

NDCG: Example

- Perfect ranking:
  - 3, 3, 3, 2, 2, 2, 1
- ideal DCG values:
  - 3, 6, 7.89, 8.89, 9.75, 10.52, 10.88
- Actual DCG: (3 2 3 0 0 1 2)
  - 3, 5, 6.89, 6.89, 6.89, 7.28, 7.99
- NDCG values (divide actual by ideal):
  - 1, 0.83, 0.87, 0.76, 0.71, 0.69
  - NDCG $\leq$ 1 at any rank position

What if the results are not in a list?

- Suppose there's only one Relevant Document

- Scenarios:
  - known-item search
  - navigational queries
  - looking for a fact

- Search duration ~ Rank of the answer
  - measures a user's effort

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

Evaluation at large search engines

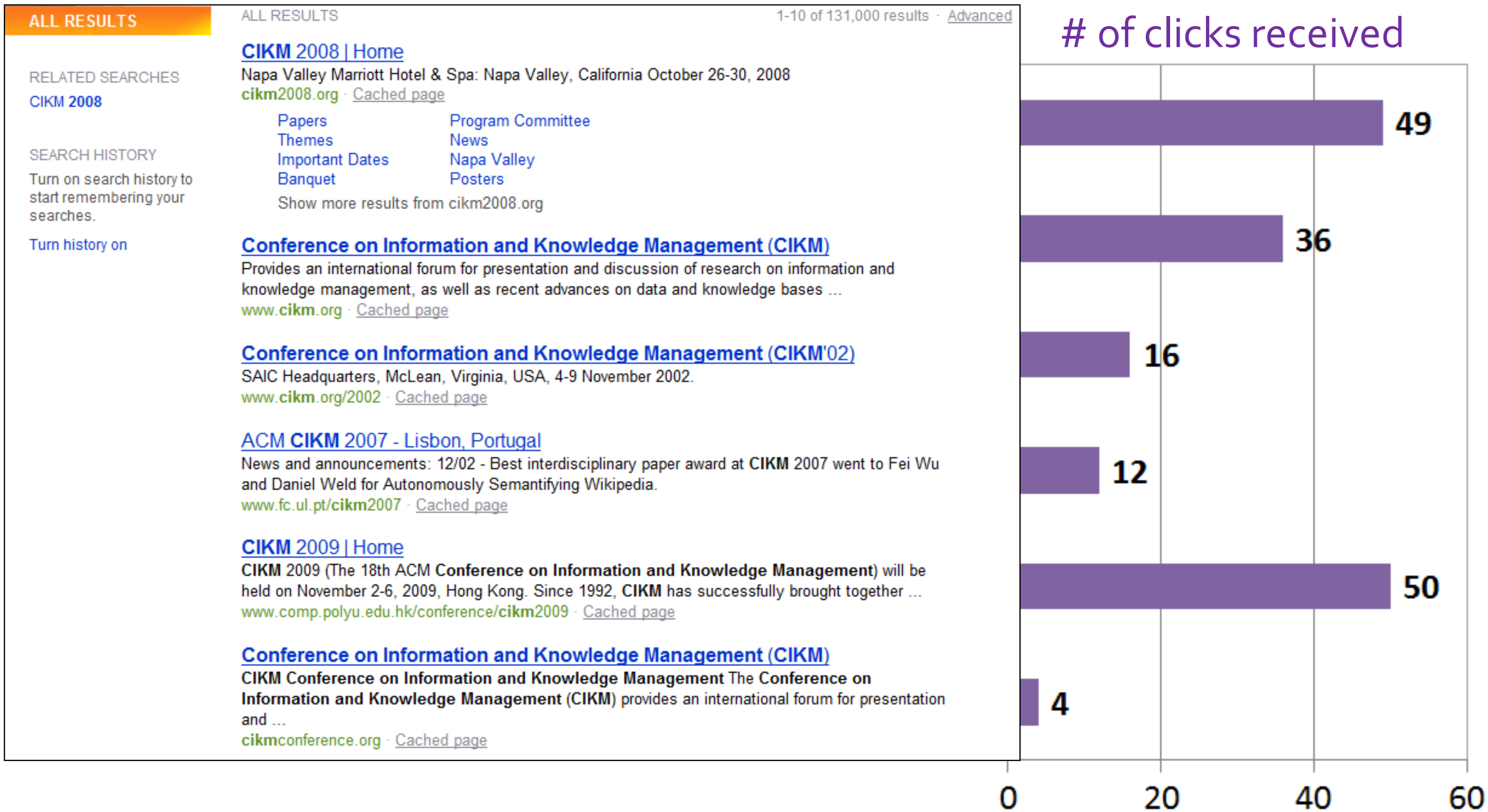
- Recall is difficult to measure on the web
  - Search engines often use precision at top k (e.g., k = 10).
  - or NDCG
- Search engines also use non-relevance-based measures.
  - User clicks
- A/B testing

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, vs underlying need
- So – what alternatives do we have?

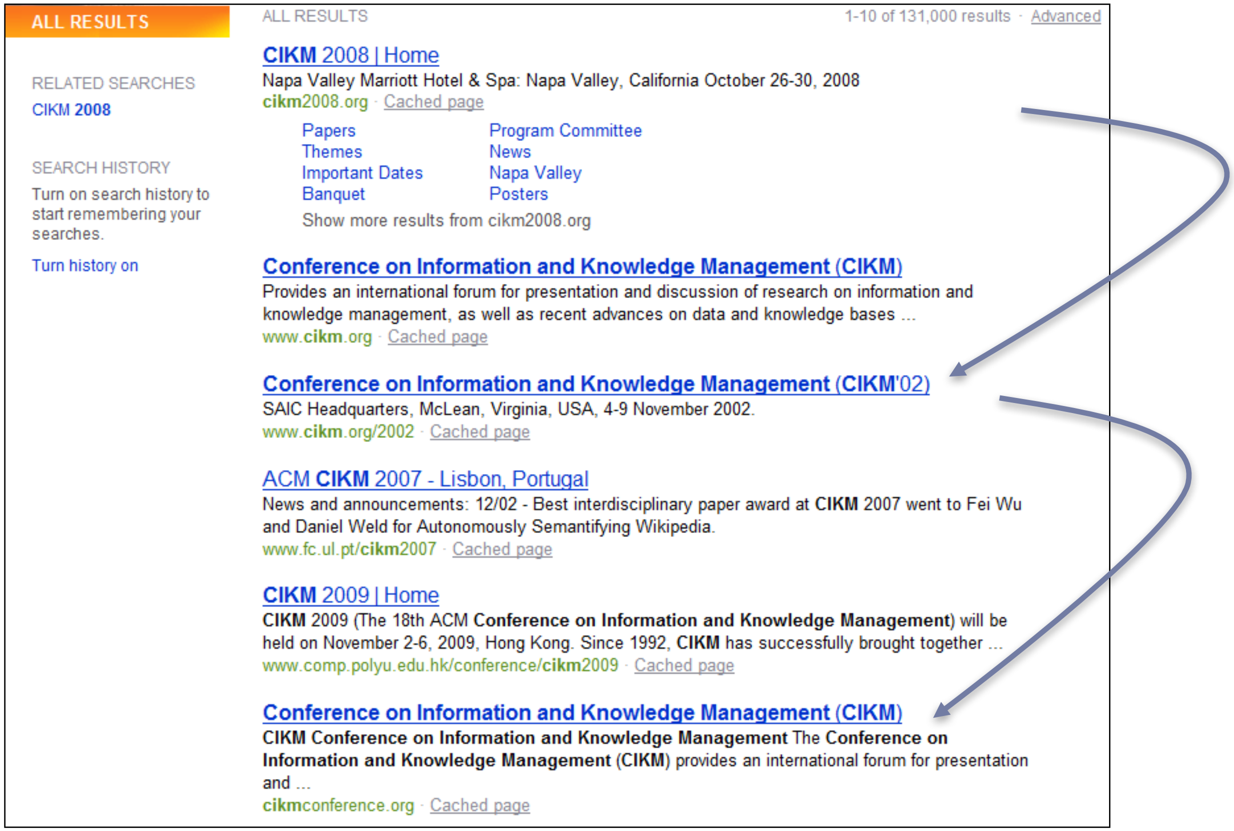
Using user Clicks

What do clicks tell us?



Strong position bias, so absolute click rates unreliable

Relative vs absolute ratings



User's click sequence

Hard to conclude Result1 > Result3

Probably can conclude Result3 > Result2

Pairwise relative ratings

- Pairs of the form: DocA better than DocB for a query
  - Doesn't mean that DocA relevant to query

- Now, rather than assess a rank-ordering wrt per-doc relevance assessments

- Assess in terms of conformance with historical pairwise preferences recorded from user clicks

A/B testing: refining a deployed system

- **Purpose**: Test a single innovation
- **Prerequisite**: You have a large search engine up and running.
- **Method**: Divert a small proportion of traffic (e.g., 1%) to the new system that includes the innovation
  - So most users use old system

A/B testing at web search engines

- Have most users use old system
- Divert a small proportion of traffic (e.g., 1%) to an experiment to evaluate an innovation
  - Full page experiment
  - Interleaved experiment

Comparing two rankings via clicks (Joachims 2002)

Query: [support vector machines]

Ranking A	Ranking B
Kernel machines	Kernel machines
SVM-light	SVMs
Lucent SVM demo	Intro to SVMs
Royal Holl. SVM	Archives of SVM
SVM software	SVM-light
SVM tutorial	SVM software

Interleave the two rankings

This interleaving starts with B

Kernel machines
Kernel machines
SVMs
SVM-light
Intro to SVMs
Lucent SVM demo
Archives of SVM
Royal Holl. SVM
SVM-light

...

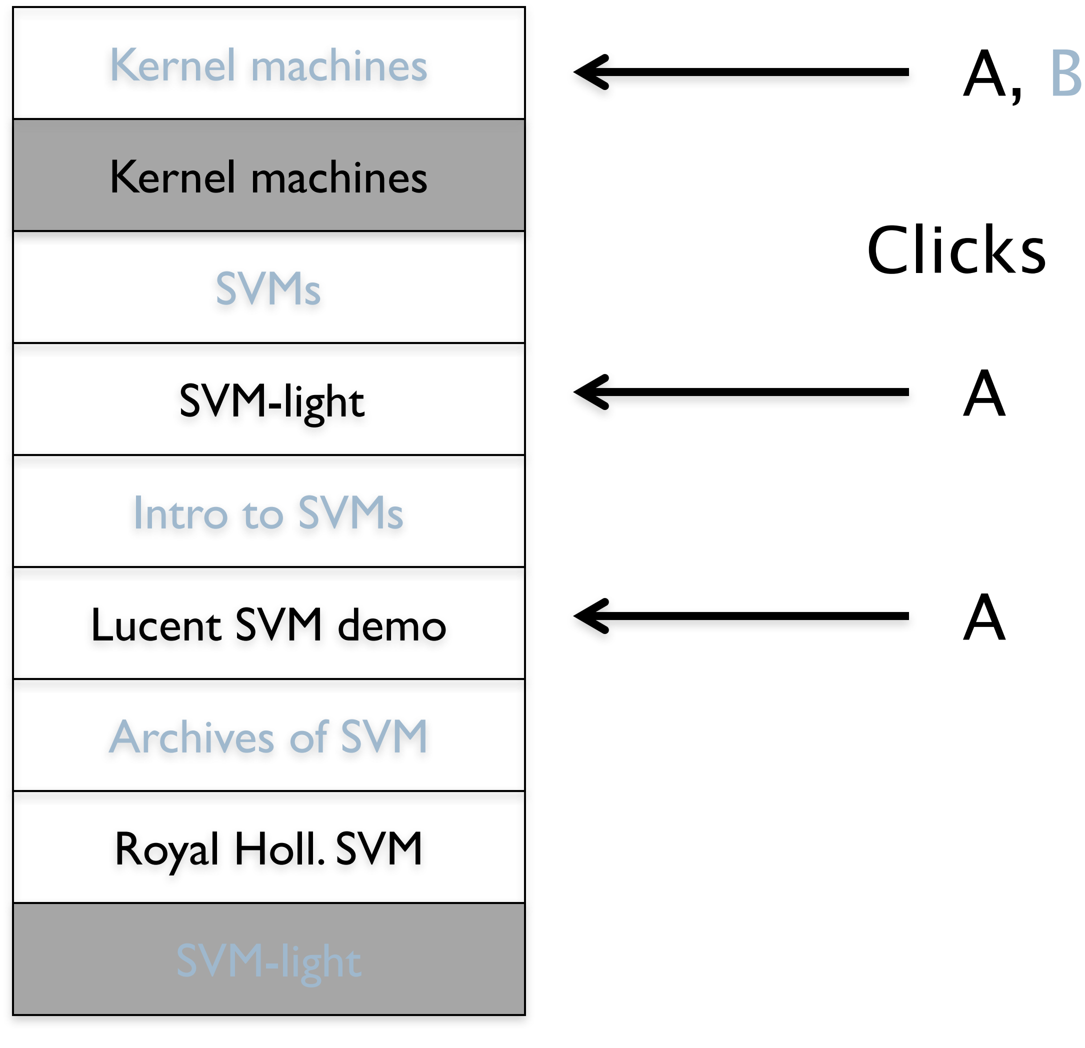
Remove duplicate results

Kernel machines
Kernel machines
SVMs
SVM-light
Intro to SVMs
Lucent SVM demo
Archives of SVM
Royal Holl. SVM
SVM-light

...

Count user clicks

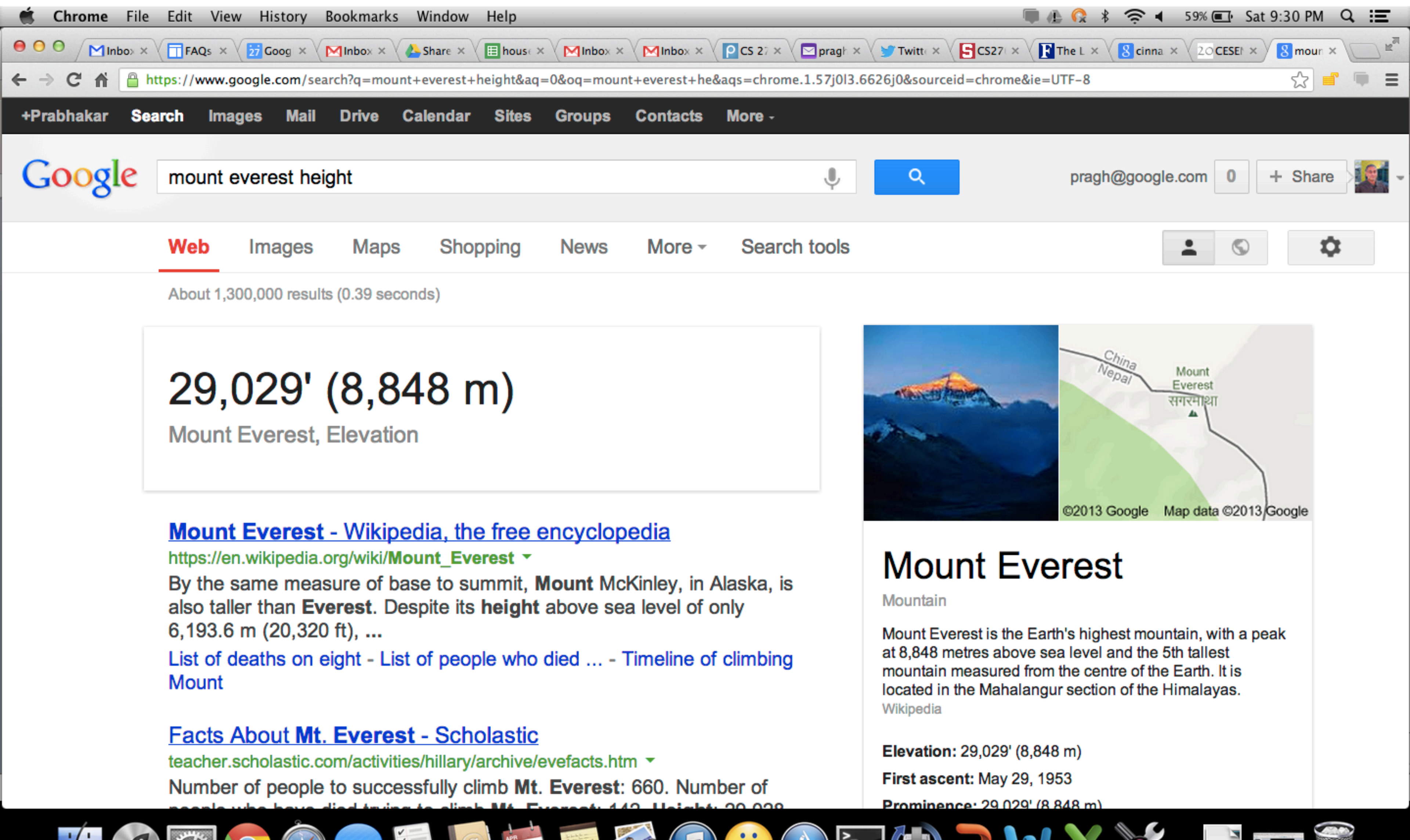
Ranking A: 3
Ranking B: 1



Interleaved ranking

- Present interleaved ranking to users
  - Start randomly with ranking A or ranking B to evens out presentation bias
- Count clicks on results from A versus results from B
- Better ranking will (on average) get more clicks

Facts/entities (what happens to clicks?)



Comparing two rankings to a baseline ranking

- Given a set of pairwise preferences *P*
- We want to measure two rankings *A* and *B*
- Define a proximity measure between *A* and *P* (and likewise, between B and P)
  - Proximity measure should reward agreements with *P* and penalize disagreements
- Want to declare the ranking with better proximity to be the winner

Kendall tau distance

- X: # of agreements between a ranking (say *A*) and *P*
- Y: # of disagreements
- Then the Kendall tau distance between *A* and *P* is

$$\frac{X - Y}{X + Y}$$

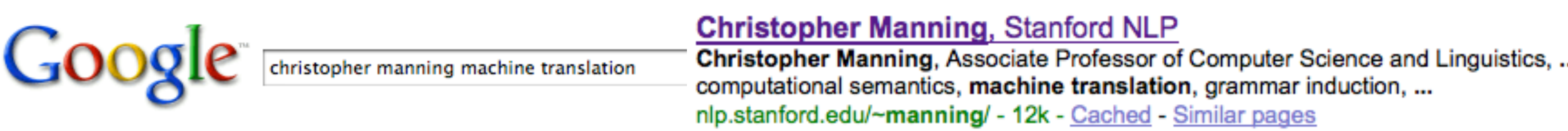
- Example:
  - P = {(1,2), (1,3), (1,4), (2,3), (2,4), (3,4))}
  - A=(1,3,2,4)
  - Then X=5, Y=1 …

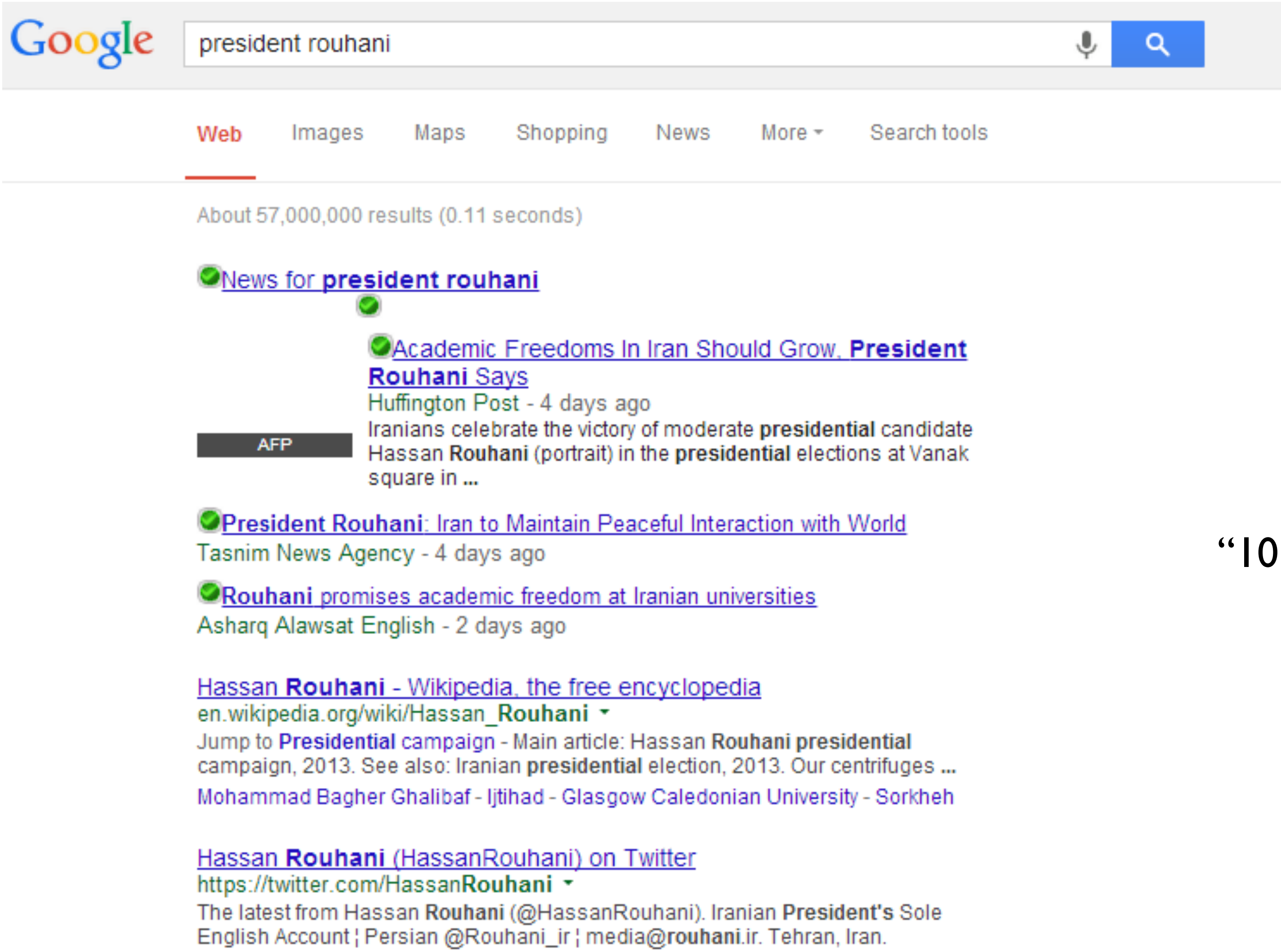
Other factors than relevance

Result summery or snippet

- Having ranked docs matching a query, we wish to present a results list that is informative to the user
  - Usually, a list of doc titles plus a short summary (snippet)

- **Snippet**: a short summary of the document that is designed so as to allow the user to decide its relevance





"10 blue links"

Result summery or snippet

- Title is often automatically extracted from doc metadata.
  - Or field and zone
- What about summaries?
  - This description is crucial.
  - User can identify good/relevant hits based on description.
- Two basic kinds:
  - Static
  - Dynamic

Summaries

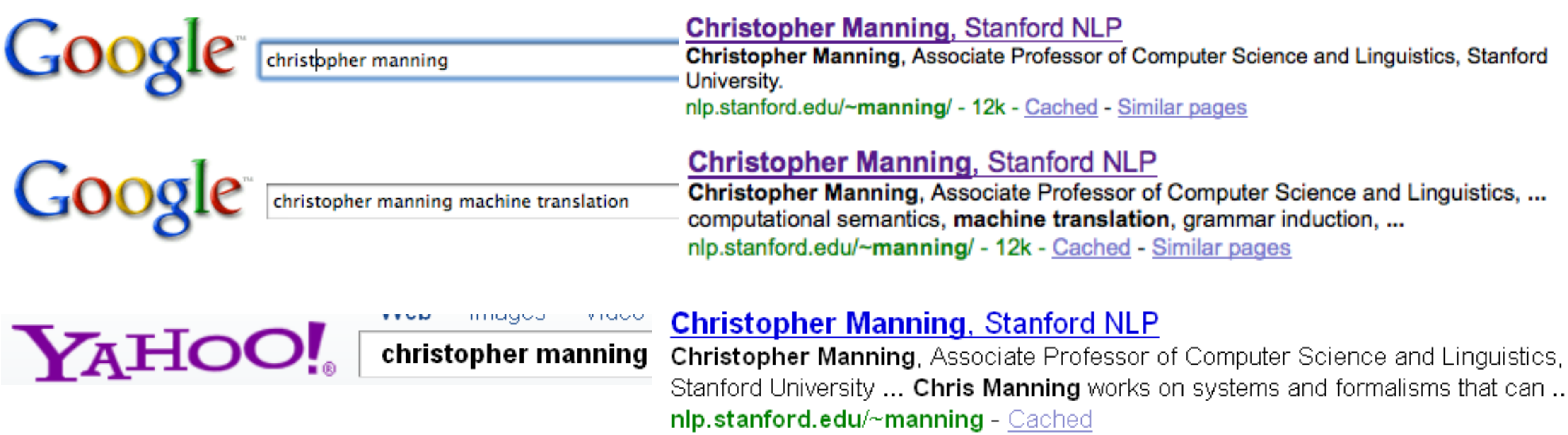
- **Static summary** of a doc is always the same, regardless of the query that hit the doc
- **Dynamic summary** is a *query-dependent* attempt to explain why doc was retrieved for query at hand

Static summaries

- In typical systems, static summary is a subset of doc.
  - Simplest heuristic: e.g., title & the first 50 words of the doc
    - Summary cached at indexing time
  - More sophisticated: extract from each doc a set of “key” sentences
    - Simple NLP heuristics to score each sentence and 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 doc that contain several of the query terms
  - "KWIC" snippets: Keyword in Context
- Requires a high disk space to save docs or at-least their prefixes
  - However, they can greatly improve the usability of IR systems.

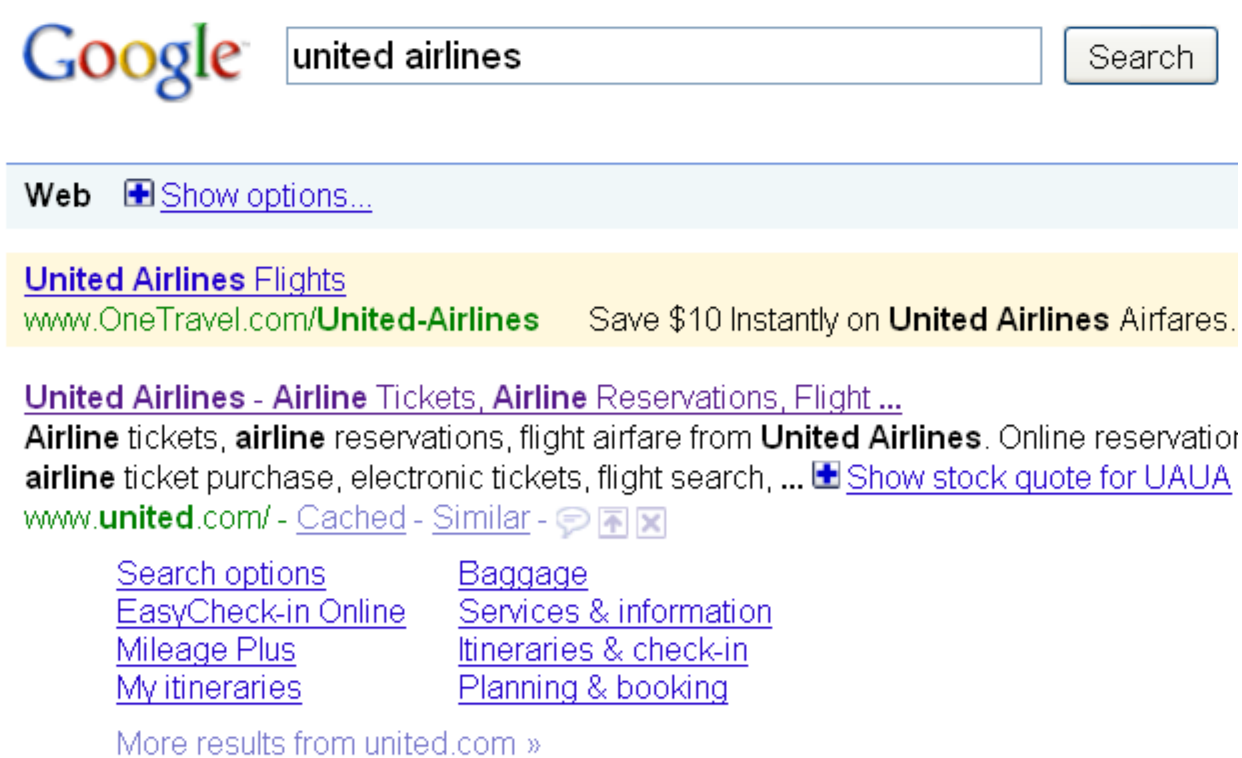


Techniques for dynamic summaries

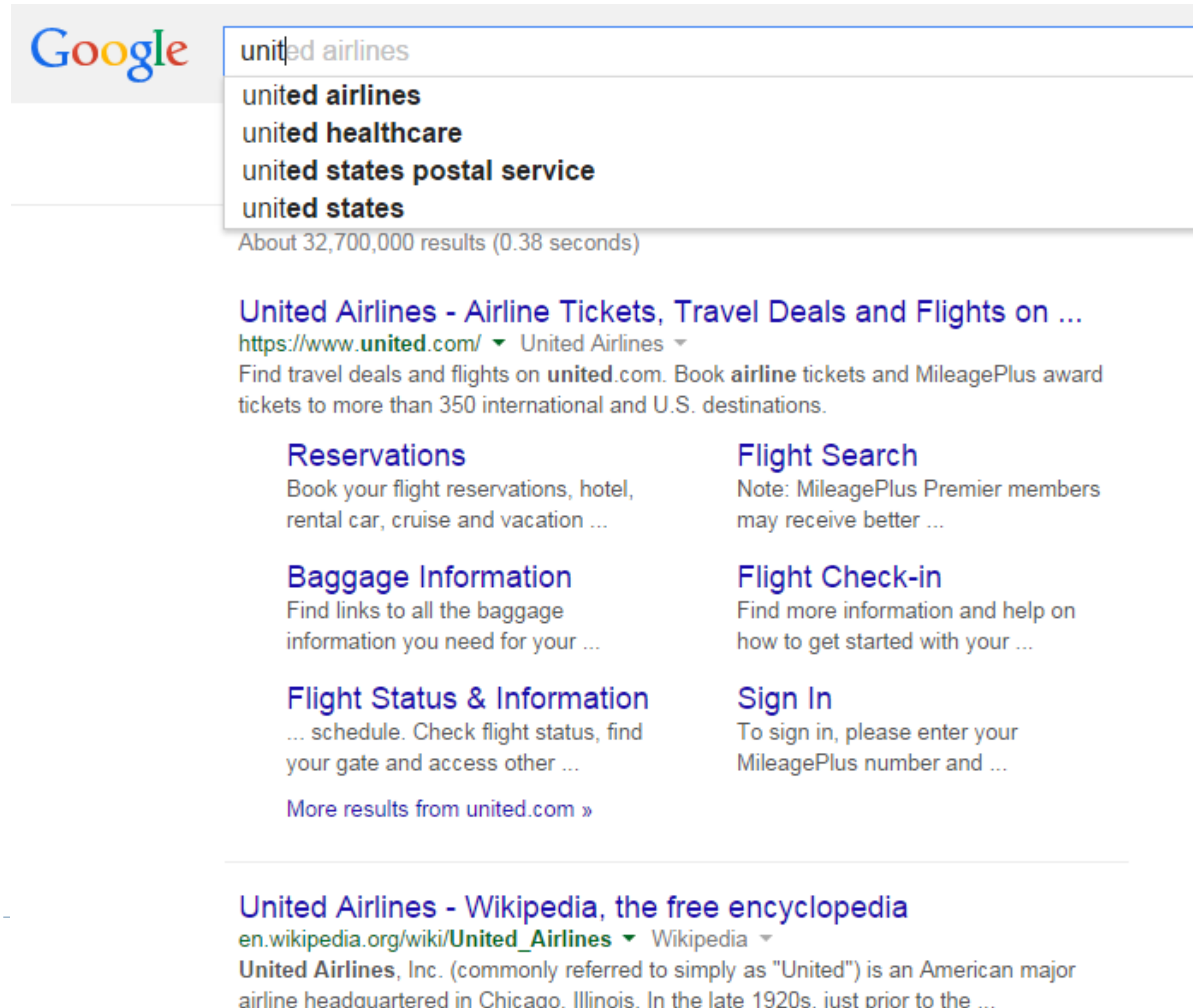
- Find small windows in doc that contain query terms
  - Requires fast window lookup in a doc cache
- Score each window wrt query
  - Use various features such as window width, position, etc.
  - Combine features through a scoring function
- Challenges in evaluation: judging summaries
  - Pairwise comparisons rather than binary relevance assessments

Quicklinks

- Example *navigational query:* ***united airlines***
  - user's need likely satisfied on www.united.com
  - Quicklinks provide navigational cues on that home page



Alternative results presentations?



Resources for this lecture

- IIR 8
- MIR Chapter 3
- MG 4.5
- Carbonell and Goldstein 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. SIGIR 21.