Intelligent Information Retrieval

Relevance Feedback

Credits

- Christopher Manning
- Prabhakar Raghavan
- Francesco Ricci

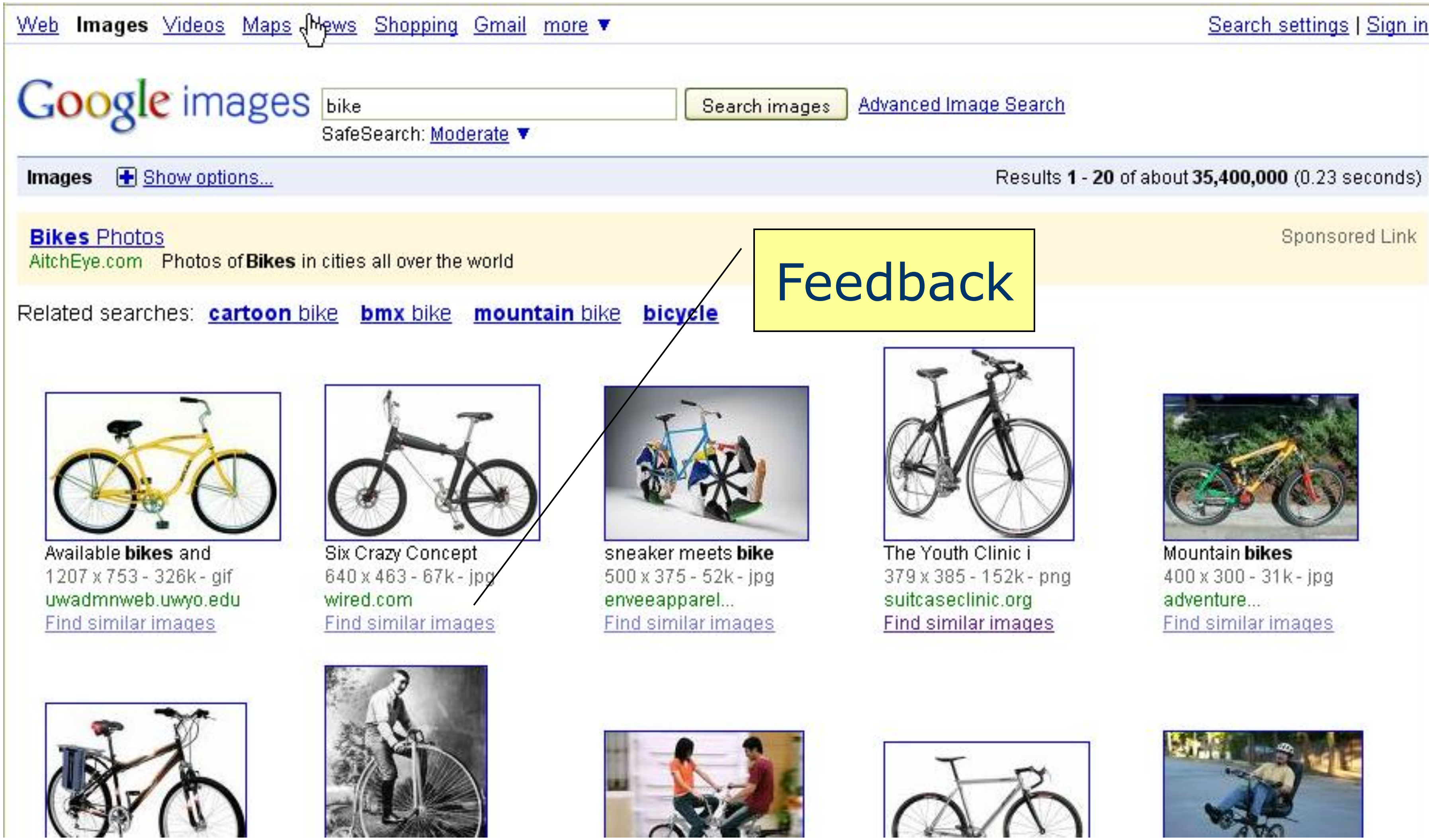
Relevance Feedback

- 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

- The process of query modification is commonly referred as
  - **relevance feedback**, when the user provides information on relevant documents to a query, or
  - **query expansion**, when information related to the query is used to expand it
- We refer to both of them as feedback methods
- Two basic approaches of feedback methods:
  - **explicit feedback**, in which the information for query reformulation is provided directly by the users
  - **implicit feedback**, in which the information for query reformulation is implicitly derived by the system

Example: search images



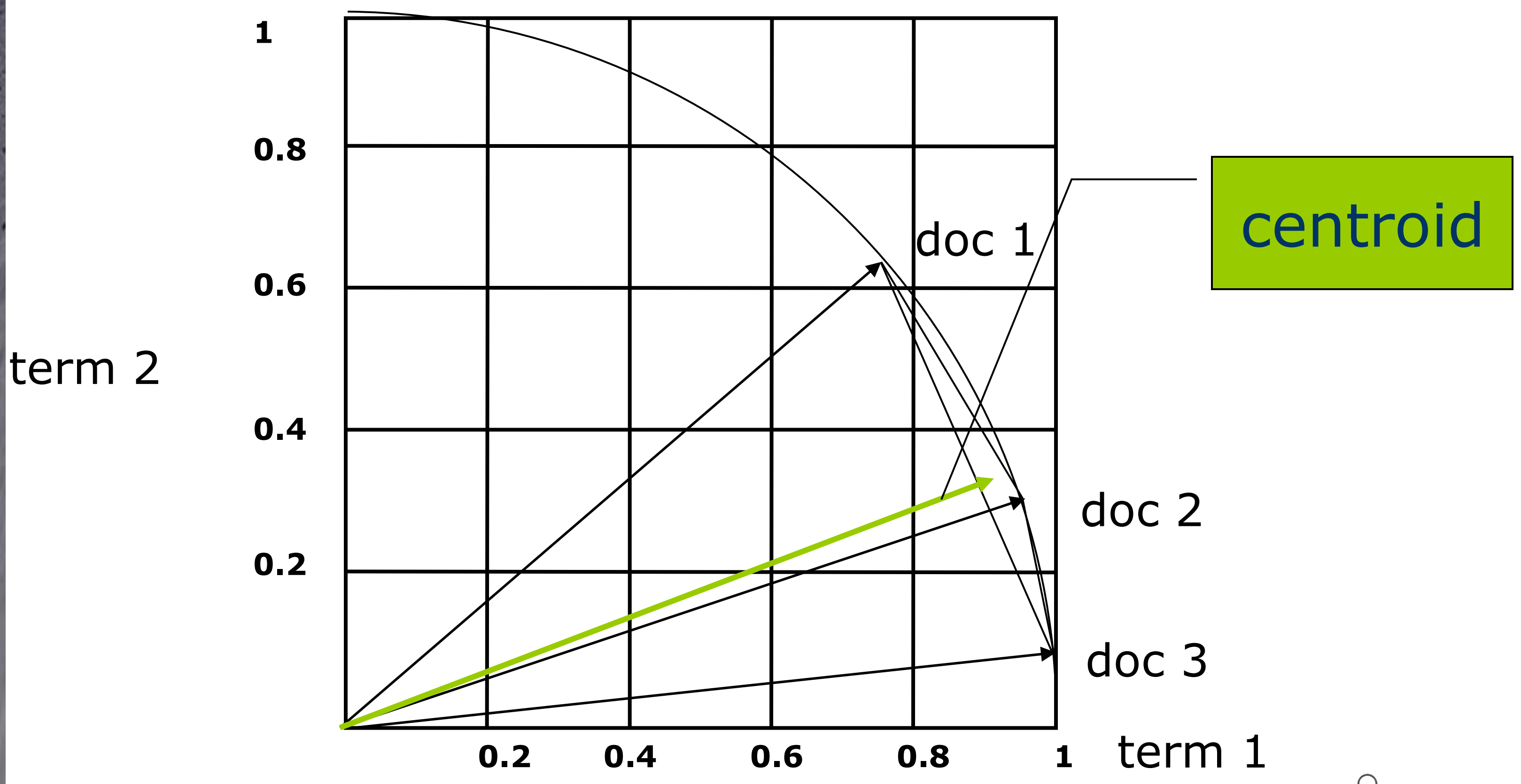
Key concept: Centroid

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

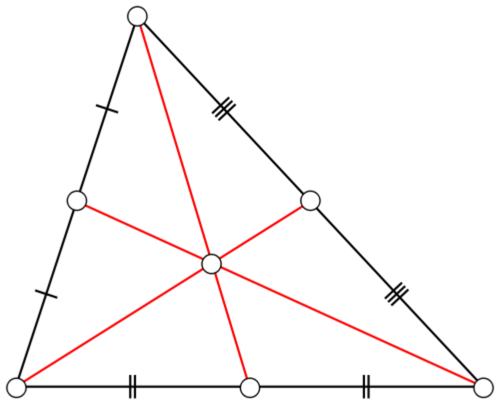
$$\vec{\mu}(C) = \frac{1}{|C|} \sum_{d \in C} \vec{d}$$

where C is a set of documents.

Example



- The centroid is not normalized



Rocchio Algorithm

- Let us define terminology regarding the processing of a given query q, as follows:
  - D_r: set of *relevant* documents among the documents retrieved
  - N_r: number of documents in set D_r
  - D_n: set of *non-relevant* documents among the documents retrieved
  - N_n: number of documents in set D_n
  - C_r: set of relevant docs among all documents in the collection
  - N: number of documents in the collection
  - α, β, γ: tuning constants

Rocchio Algorithm

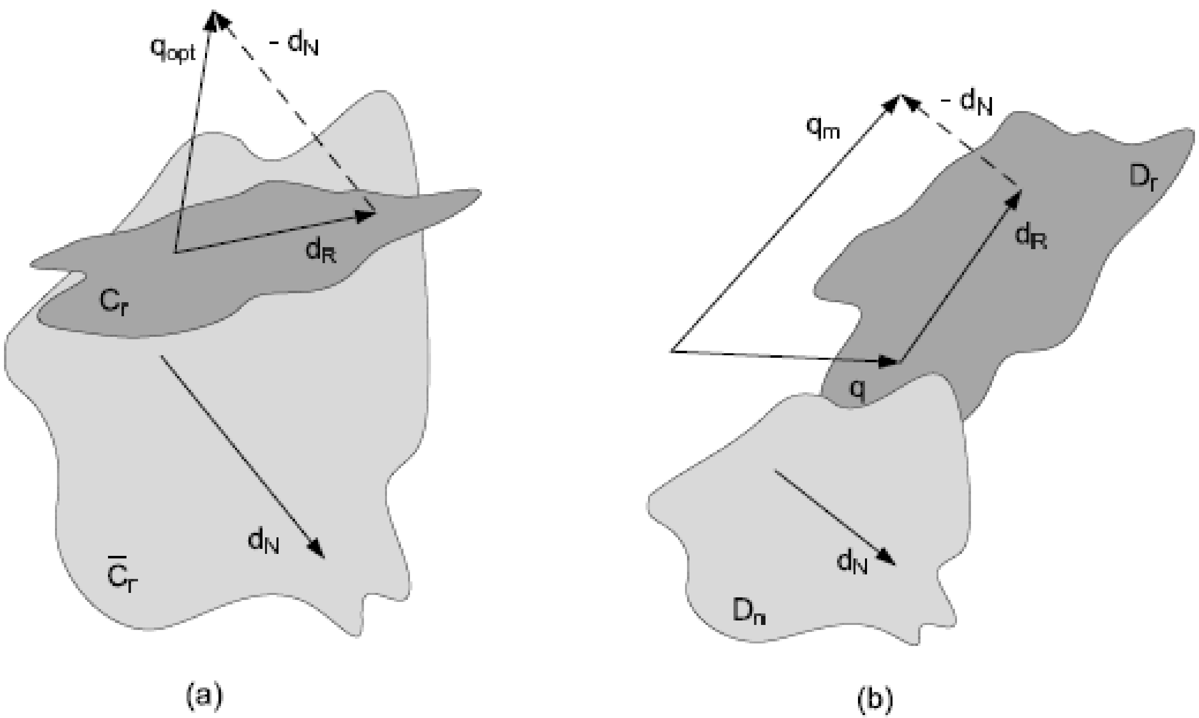
- The Rocchio algorithm uses the vector space model to pick a relevance feedback query
- Tries to separate docs marked relevant and non-relevant – the solution is:

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

- Problem: we don't know the truly relevant docs (C_r).

Rocchio Algorithm

- C_r is not known a priori
- To solve this problem, we can formulate an initial query and to incrementally change the initial query vector



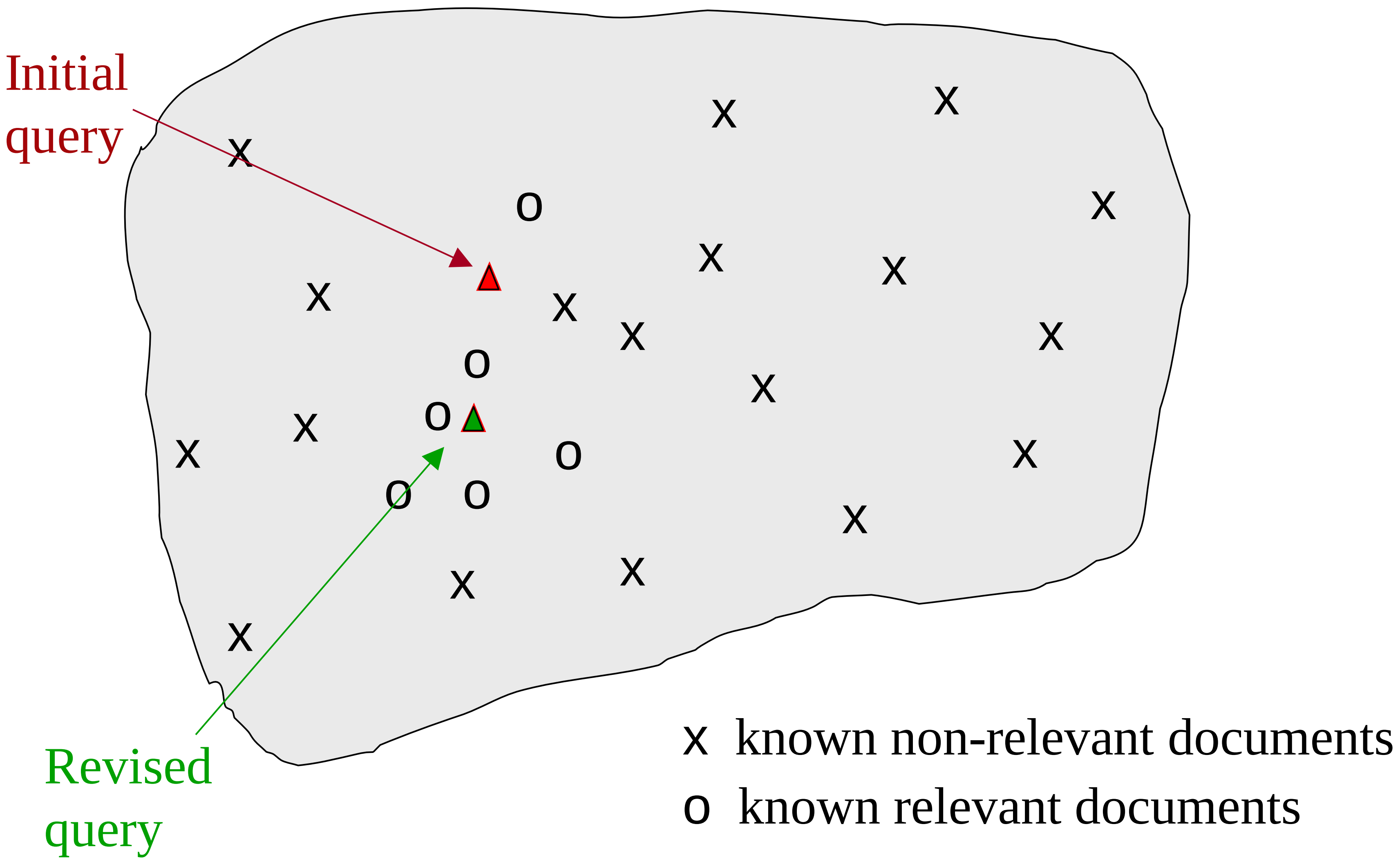
Rocchio 1971 Algorithm (SMART)

- Used in practice:

$$\vec{q}_m = \alpha \vec{q}_0 + \frac{\beta}{N_r} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \frac{\gamma}{N_n} \sum_{\vec{d}_j \in D_n} \vec{d}_j$$

- D_r = set of <u>known</u> relevant doc vectors
- D_n = set of <u>known</u> irrelevant doc vectors
  - ✓ These are different from C_r!
- q_m = modified query vector; q_0 = original query vector; α,β,γ: weights (hand-chosen or set empirically)
- New query moves toward relevant documents and away from irrelevant documents.

Relevance feedback on initial query



Subtleties to note

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

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
- Information needs may change during the interaction (so what?).

Evaluation of relevance feedback strategies

- Use q_0 and compute precision and recall graph
- Use q_m and compute precision recall graph
  - 1) Assess on all documents in the collection
    - Spectacular improvements, but … it's cheating!
    - Known relevant documents ranked higher
    - Must evaluate with respect to documents not seen by user
  - 2) Use documents in residual collection (all docs minus those assessed relevant)
    - Measures usually 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

- Most satisfactory – use two collections each with their own relevance assessments (i.e., split randomly the collection in two parts)
  - ✓ 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** – or using similar user effort
- 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

$\alpha/\beta/\gamma$??

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

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: e.g. if the top results of a query are all about a subtopic
- Several iterations can cause query drift
- Why?

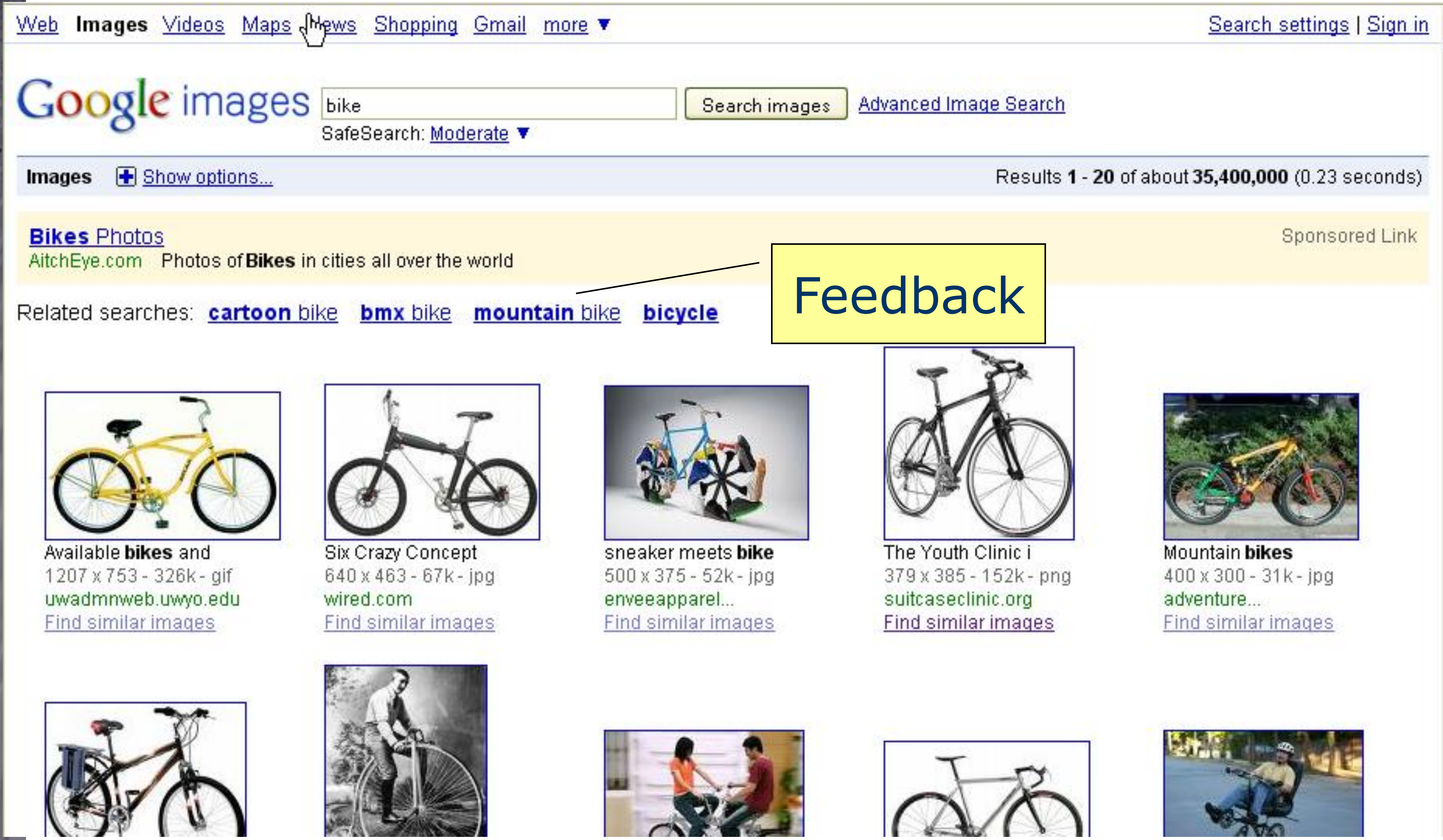
Indirect relevance feedback

- Ranked higher documents 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.

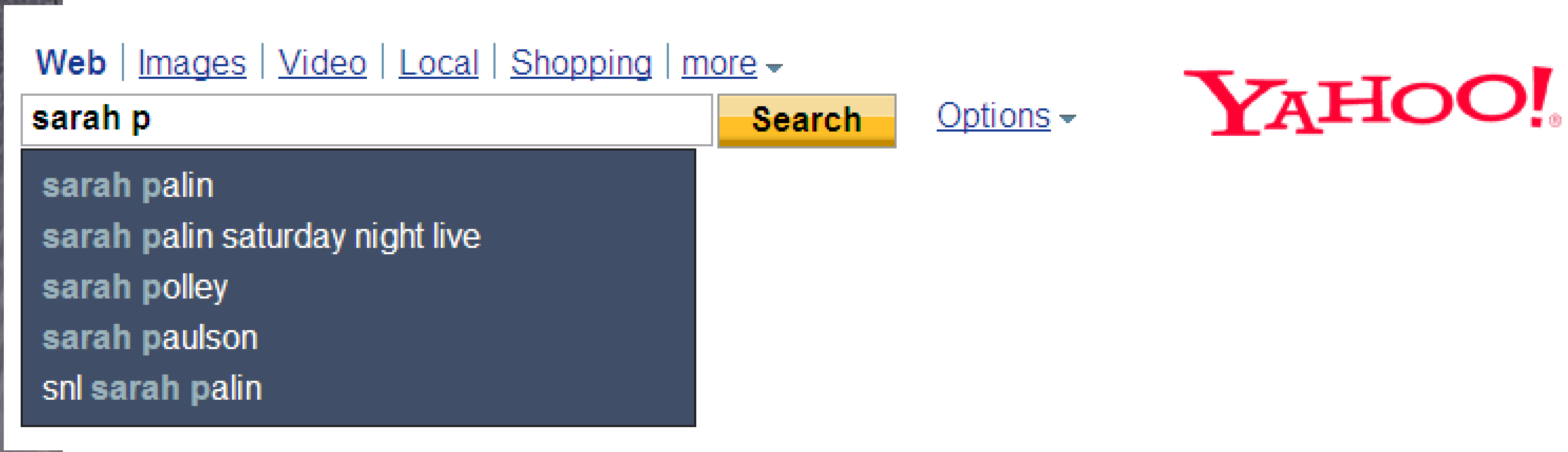
Query Expansion

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

Example: search images



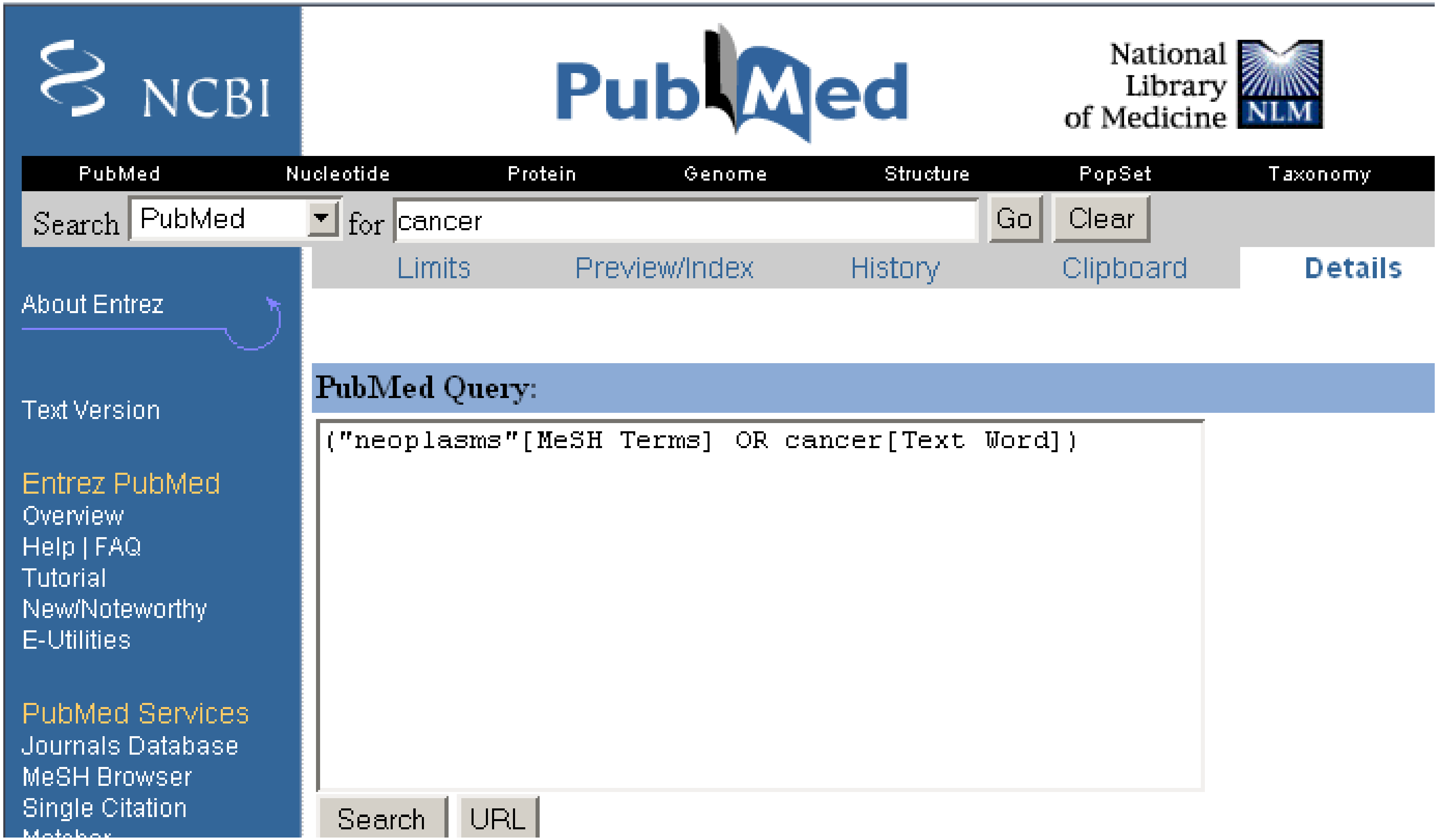
Query assist



How do we augment the user query?

- Manual thesaurus
  - E.g. MedLine: **physician**, syn: *doc, doctor, MD, medico*
  - Can be related queries rather than just synonyms
- **Global Analysis:** static; based on 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

Example of manual thesaurus



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.

Query assist

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