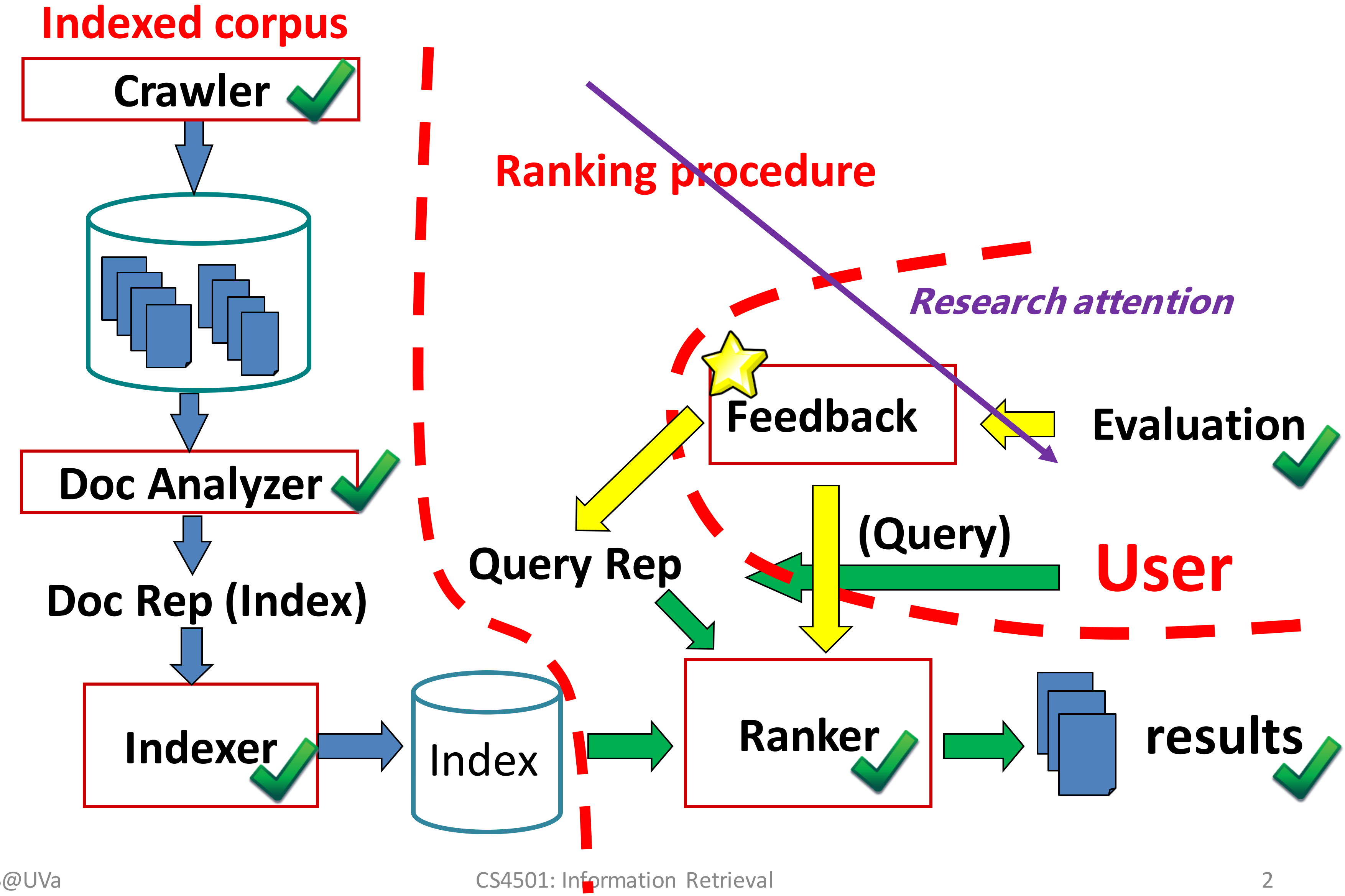
Relevance Feedback

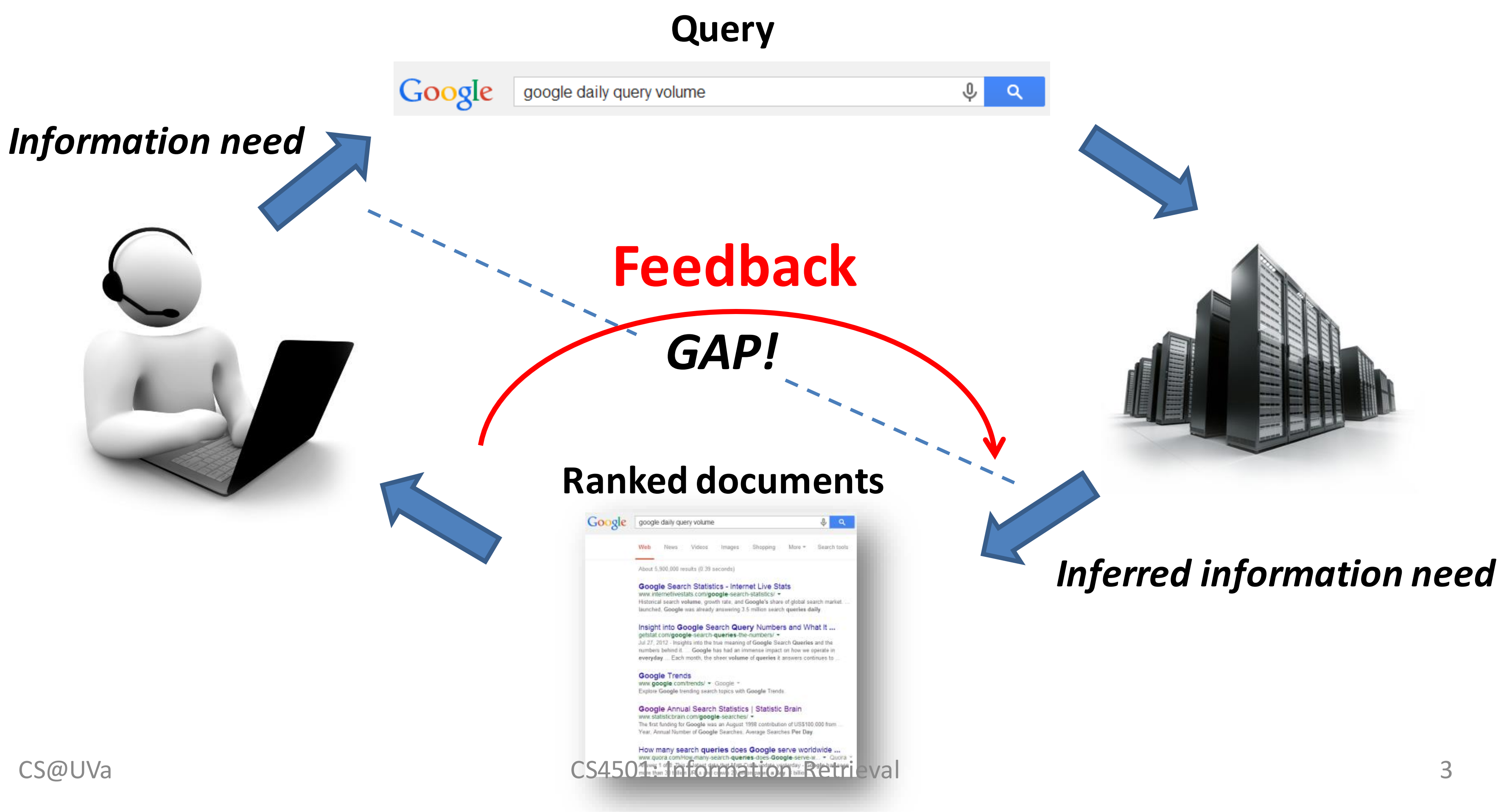
Hongning Wang
CS@UVa

What we have learned so far

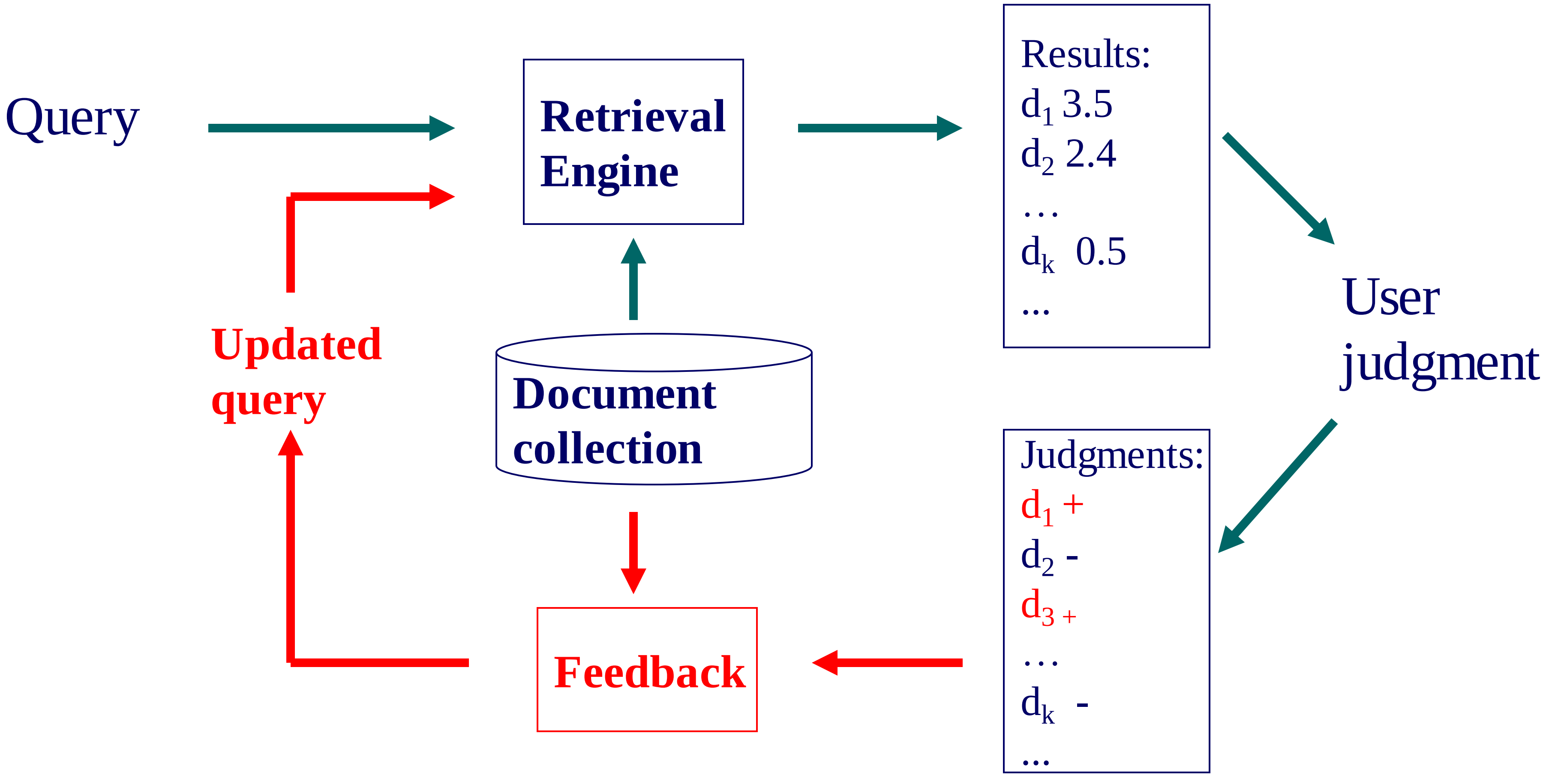


User feedback

- An IR system ~~is~~ should be an interactive system

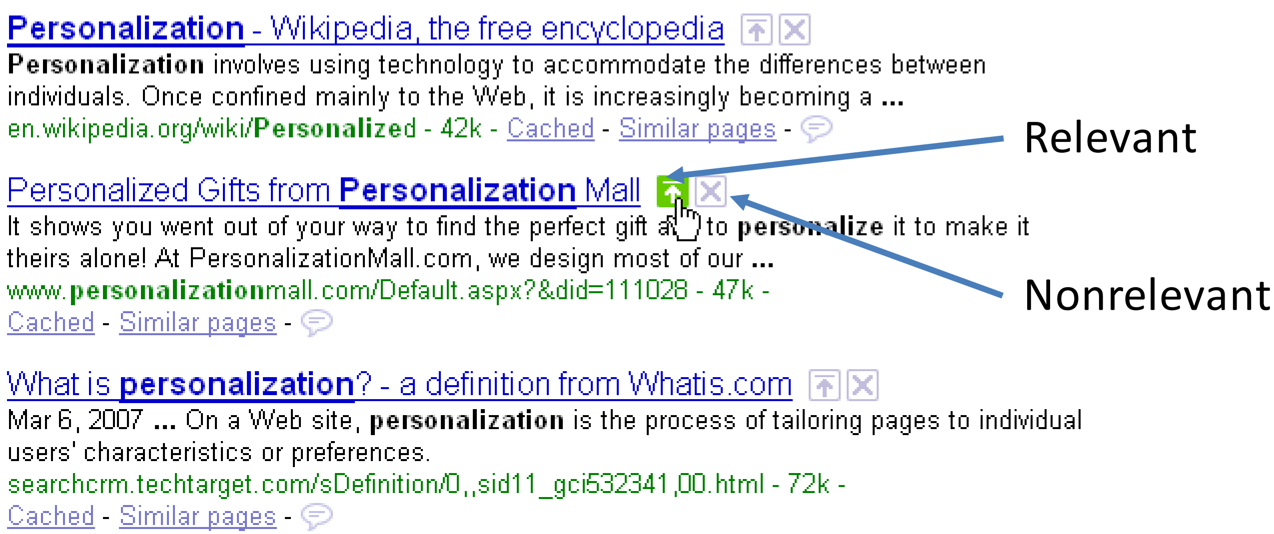


Relevance feedback



Relevance feedback in real systems

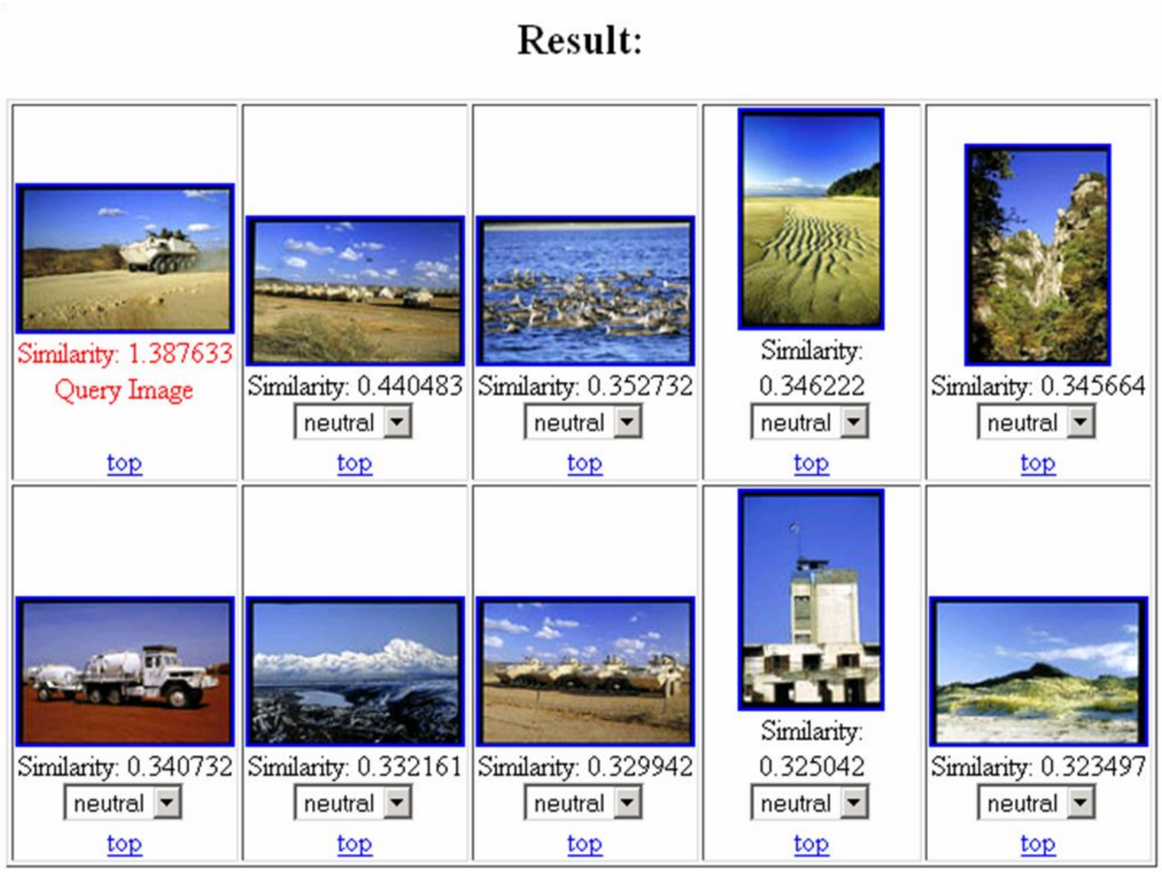
- Google used to provide such functions



  – Guess why?

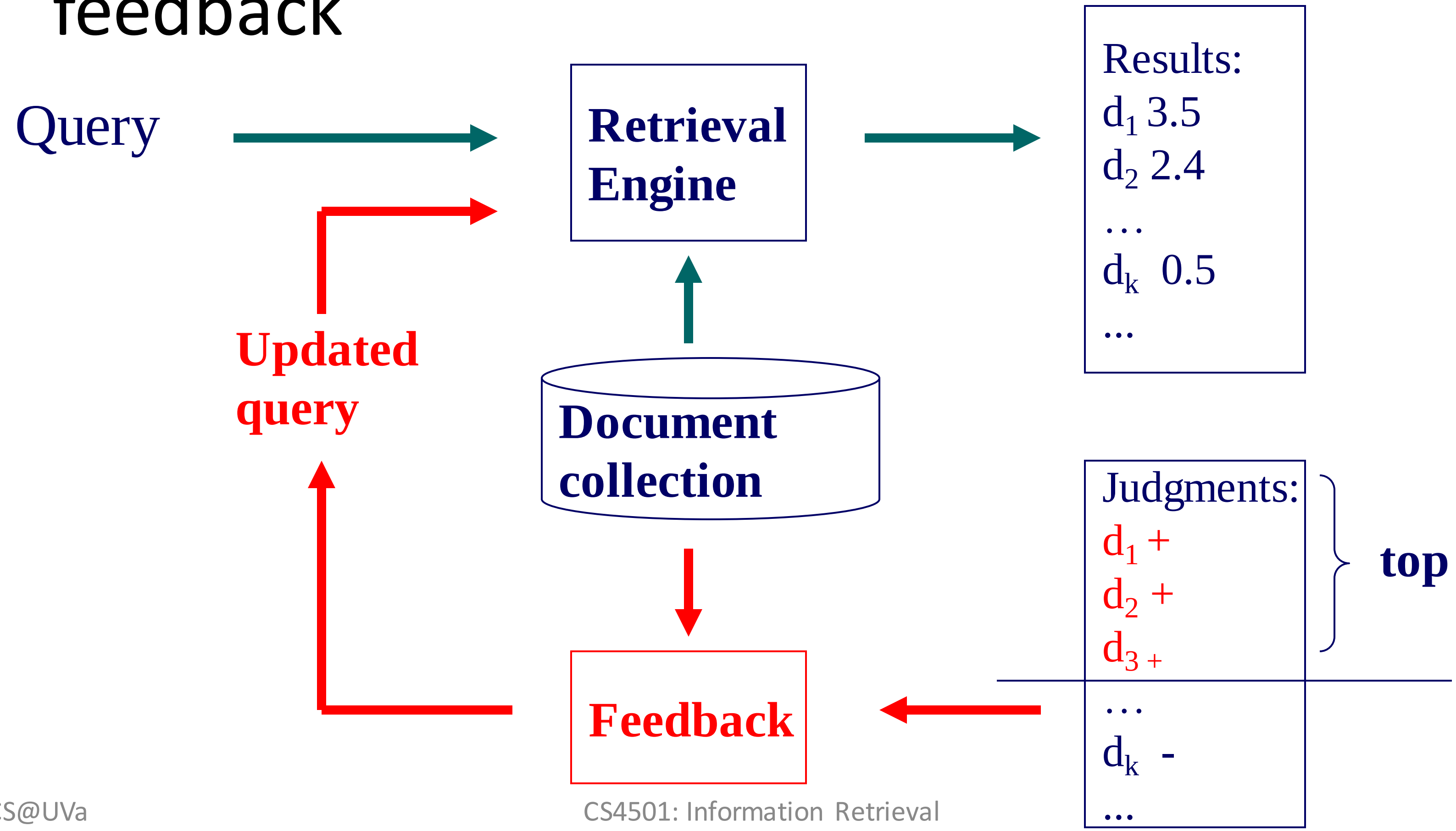
Relevance feedback in real systems

- Popularly used in image search systems



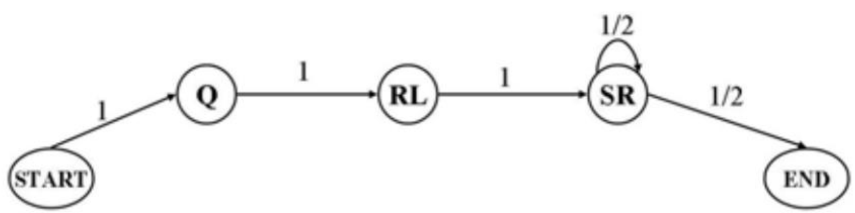
Pseudo feedback

- What if the users are reluctant to provide any feedback

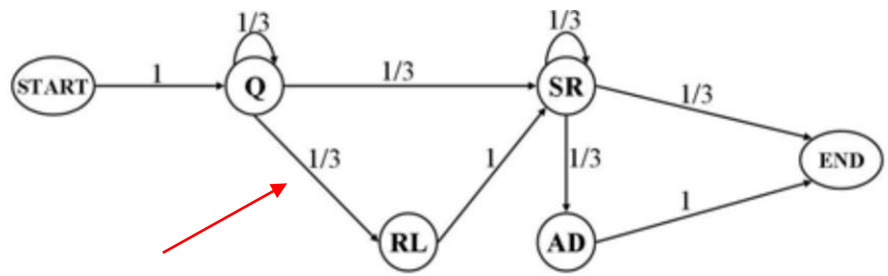


Recap: Beyond DCG: User Behavior as a Predictor of a Successful Search

- Modeling users' sequential search behaviors with Markov models
  - A model for successful search patterns

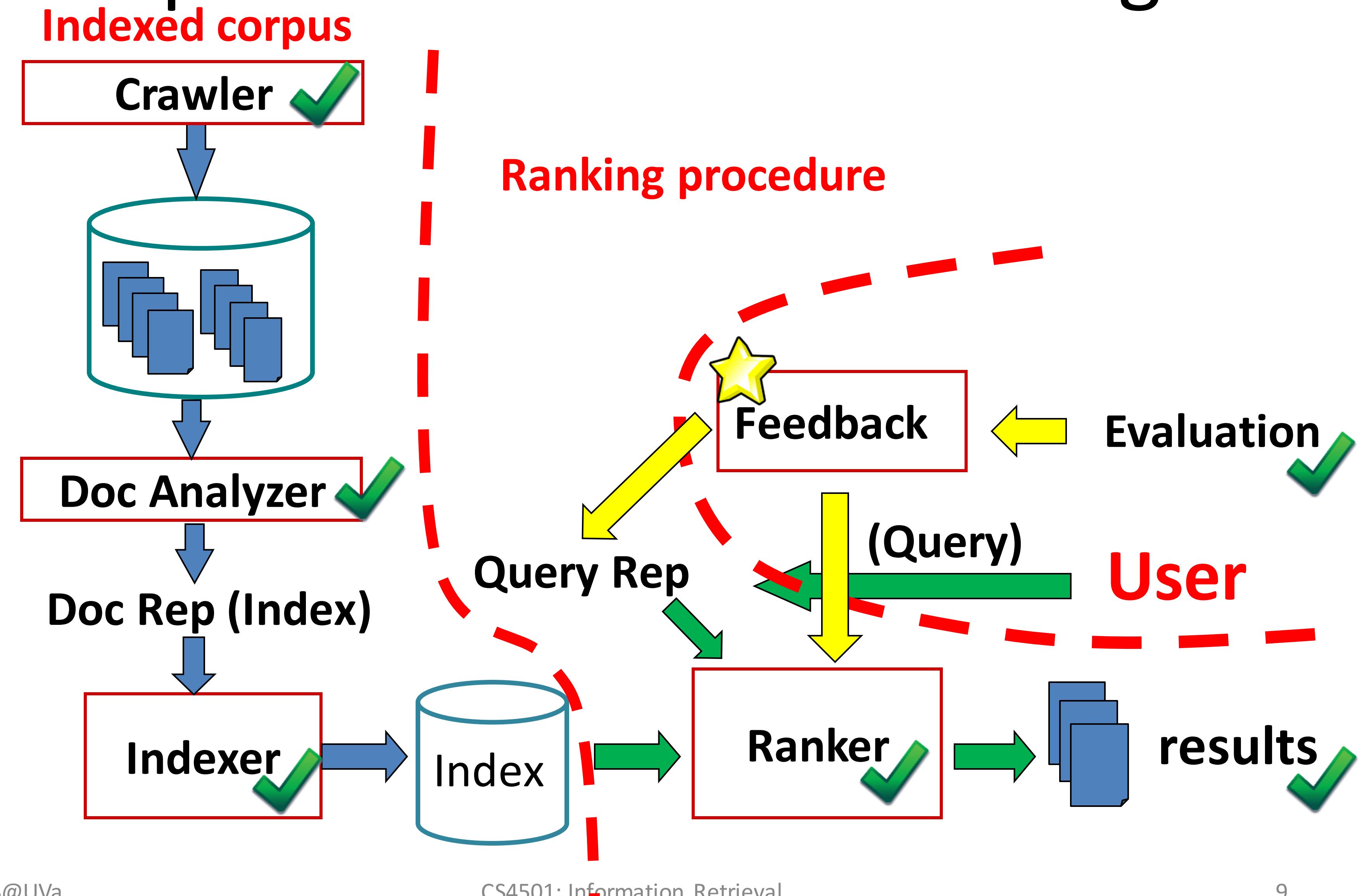


  - A model for unsuccessful search patterns

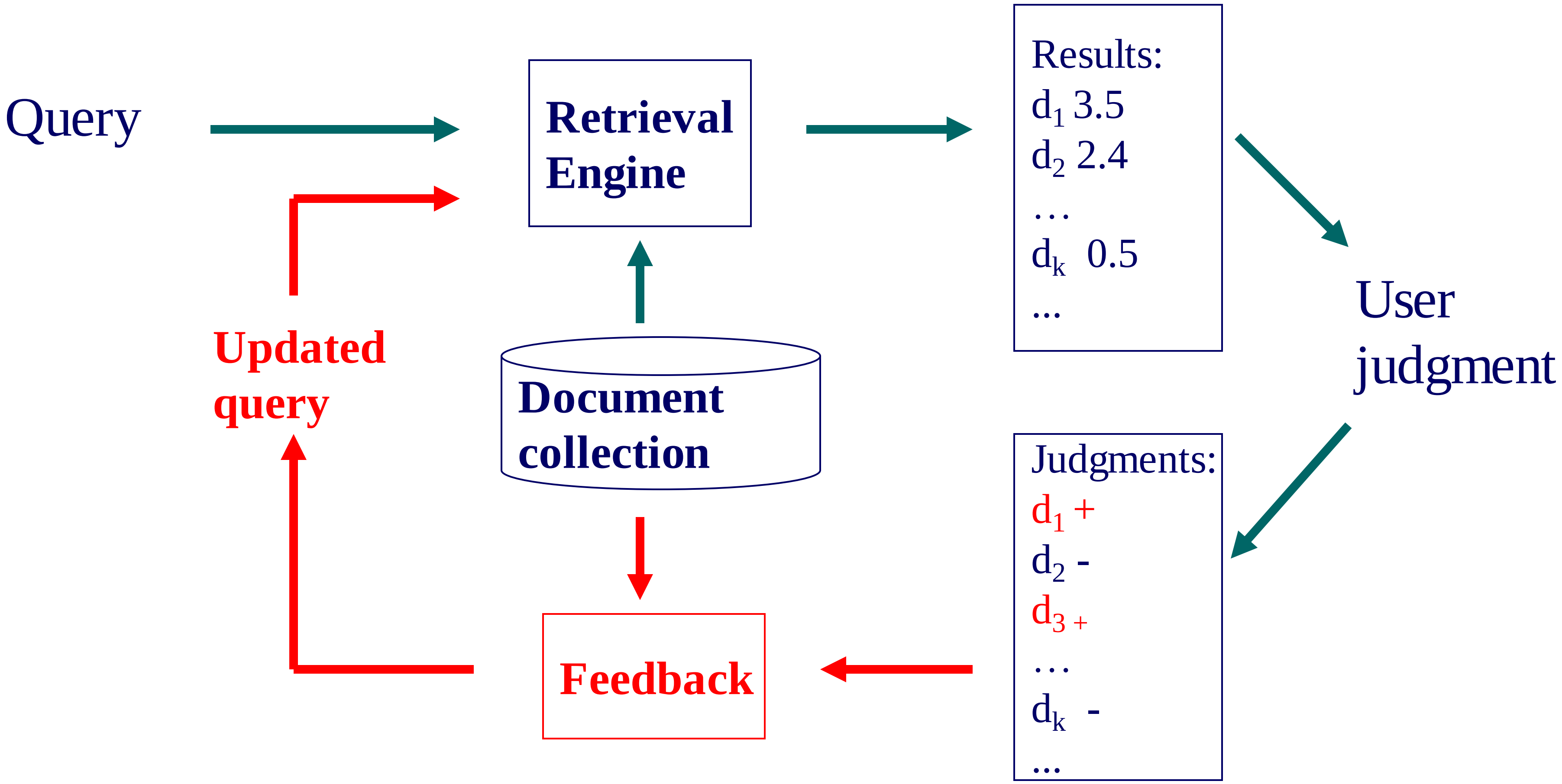


ML for parameter estimation on annotated data set

Recap: feedback in a search engine

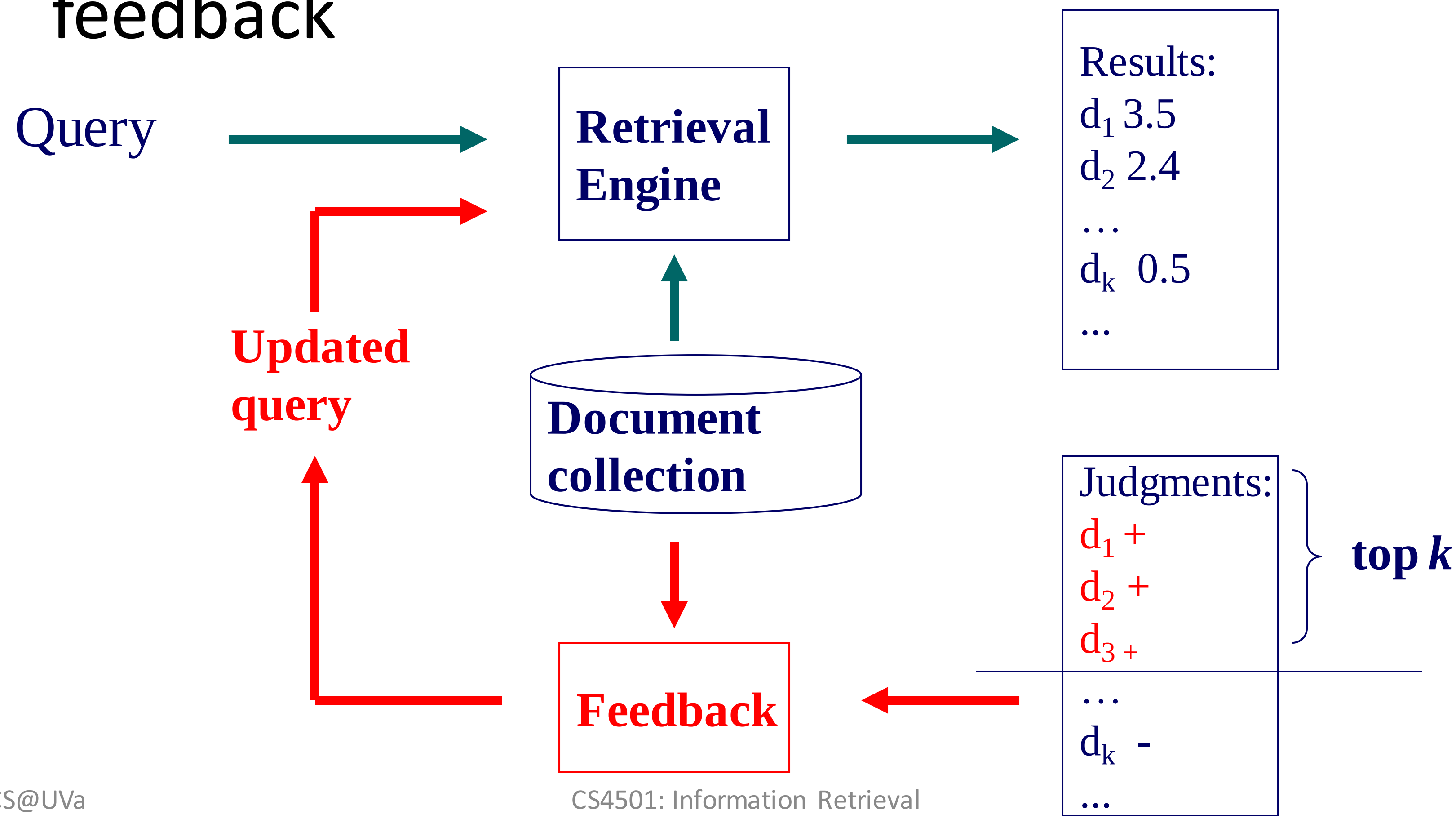


Recap: relevance feedback



Recap: pseudo feedback

- What if the users are reluctant to provide any feedback



Basic idea in feedback

- Query expansion
  - Feedback documents can help discover related query terms
  - E.g., query="information retrieval"
    - Relevant or pseudo-relevant docs may likely share very related words, such as "search", "search engine", "ranking", "query"
    - Expand the original query with such words will increase recall and sometimes also precision

Basic idea in feedback

- Learning-based retrieval
  - Feedback documents can be treated as supervision for ranking model update
  - Will be covered in the lecture of “learning-to-rank”

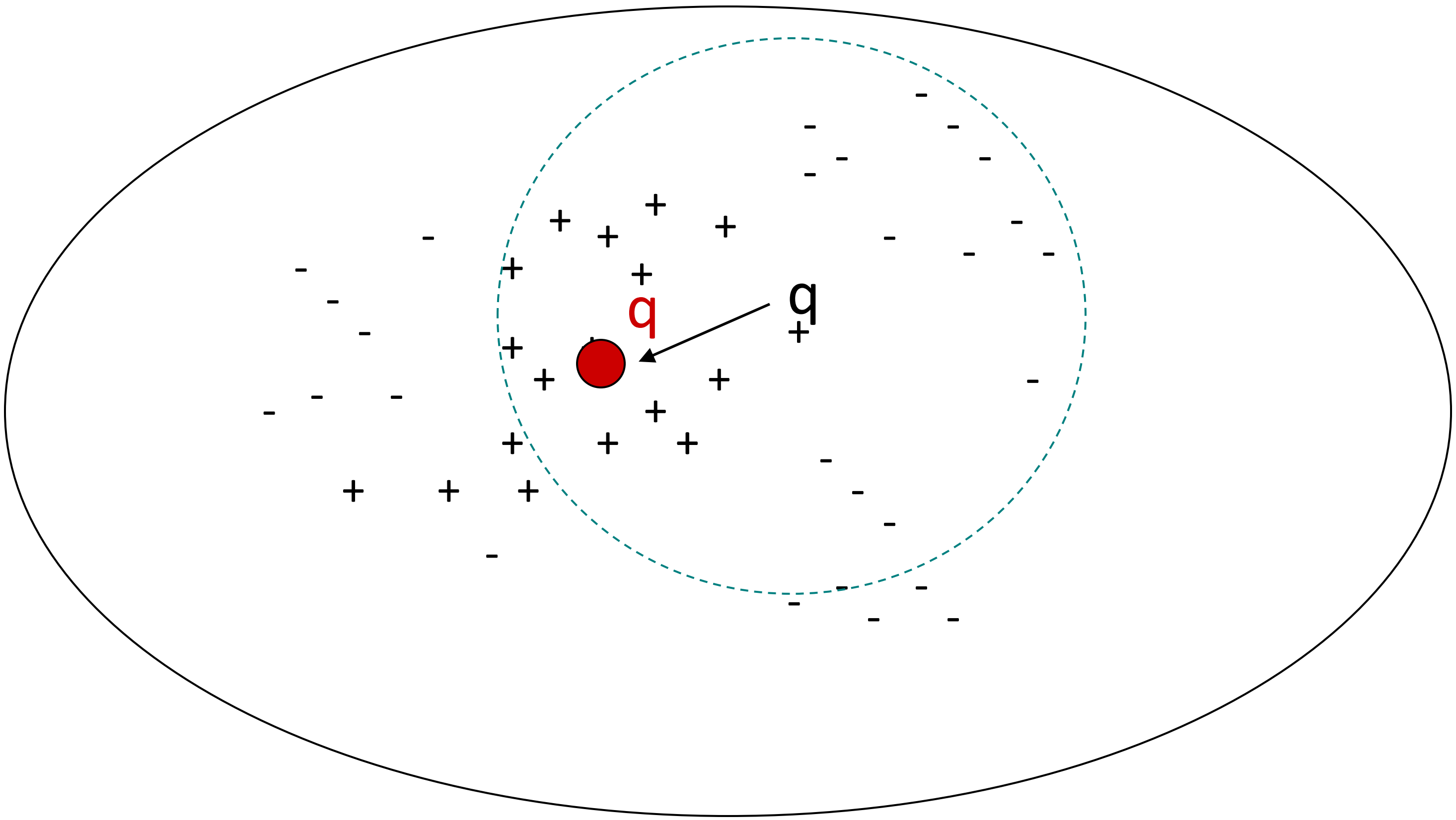
Feedback techniques

- Feedback as query expansion
  - Step 1: Term selection
  - Step 2: Query expansion
  - Step 3: Query term re-weighting
- Feedback as training signal
  - Will be covered later in learning to rank

Relevance feedback in vector space models

- General idea: query modification
  - Adding new (weighted) terms
  - Adjusting weights of old terms
- The most well-known and effective approach is Rocchio [Rocchio 1971]

Illustration of Rocchio feedback



Formula for Rocchio feedback

- Standard operation in vector space

Parameters

Modified query

$$\vec{q}_m = \alpha \vec{q} + \frac{\beta}{|D_r|} \sum_{\forall \vec{d}_i \in D_r} \vec{d}_i - \frac{\gamma}{|D_n|} \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j$$

Original query

Rel docs

Non-rel docs

Rocchio in practice

- Negative (non-relevant) examples are not very important (why?)
- Efficiency concern
  - Restrict the vector onto a lower dimension (i.e., only consider highly weighted words in the centroid vector)
- Avoid "training bias"
  - Keep relatively high weight on the original query
- Can be used for relevance feedback and pseudo feedback
- Usually robust and effective

Feedback in probabilistic models

Classic Prob. Model $O(R=1|Q,D) \propto \frac{P(D|Q,R=1)}{P(D|Q,R=0)}$ **Rel. doc model** / **NonRel. doc model**

Language Model $O(R=1|Q,D) \propto P(Q|D,R=1)$ ← **"Rel. query" model**

Parameter Estimation

$$\left.\begin{matrix}(q_1,d_1,1)\\(q_1,d_2,1)\\(q_1,d_3,1)\end{matrix}\right\} P(D|Q,R=1)$$

$$\left.\begin{matrix}(q_1,d_4,0)\\(q_1,d_5,0)\end{matrix}\right\} P(D|Q,R=0)$$

$$\left.\begin{matrix}(q_3,d_1,1)\\(q_4,d_1,1)\\(q_5,d_1,1)\\(q_6,d_2,1)\\(q_6,d_3,0)\end{matrix}\right\} P(Q|D,R=1)$$

Initial retrieval:

- P(D|Q,R=1): query as rel doc
- P(Q|D,R=1): doc as rel query

Feedback:

- P(D|Q,R=1) can be improved for the current query and future doc
- P(Q|D,R=1) can be improved for the current doc and future query

Document generation model

$$Odd(R=1|Q,D) \propto \prod_{i=1}^{k} \frac{P(A_i = d_i | Q, R=1)}{P(A_i = d_i | Q, R=0)}$$

Terms occur in doc

Terms do not occur in doc

$$= \boxed{\prod_{i=1,d_i=1}^{k} \frac{P(A_i = 1 | Q, R=1)}{P(A_i = 1 | Q, R=0)}} \boxed{\prod_{i=1,d_i=0}^{k} \frac{P(A_i = 0 | Q, R=1)}{P(A_i = 0 | Q, R=0)}}$$

Important tricks

$$\approx \prod_{i=1,d_i=q_i=1}^{k} \frac{p_i}{u_i} \prod_{i=1,d_i=0,q_i=1}^{k} \frac{1-p_i}{1-u_i}$$

Assumption: terms not occurring in the query are equally likely to occur in relevant and nonrelevant documents, i.e., $p_t=u_t$

$$= \prod_{i=1,d_i=q_i=1}^{k} \frac{p_i(1-u_i)}{u_i(1-p_i)} \cancel{\prod_{i=1,q_i=1}^{k} \frac{1-p_i}{1-u_i}}$$

document	relevant(R=1)	nonrelevant(R=0)
term present A_i=1	p_i	u_i
term absent A_i=0	1-p_i	1-u_i

Robertson-Sparck Jones Model

(Robertson & Sparck Jones 76)

$$\log O(R=1|Q,D) \overset{Rank}{\approx} \sum_{i=1,d_i=q_i=1}^{k} \log \frac{p_i(1-u_i)}{u_i(1-p_i)} = \sum_{i=1,d_i=q_i=1}^{k} \log \frac{p_i}{1-p_i} + \log \frac{1-u_i}{u_i}$$ (RSJ model)

Two parameters for each term A_i:

$p_i = P(A_i=1|Q,R=1)$: prob. that term A_i occurs in a relevant doc

$u_i = P(A_i=1|Q,R=0)$: prob. that term A_i occurs in a non-relevant doc

How to estimate these parameters?

Suppose we have relevance judgments,

$$\hat{p}_i = \frac{\#(rel.\ doc\ with\ A_i) + 0.5}{\#(rel.doc) + 1} \qquad \hat{u}_i = \frac{\#(nonrel.\ doc\ with\ A_i) + 0.5}{\#(nonrel.doc) + 1}$$

"+0.5" and "+1" can be justified by Bayesian estimation as priors

P(D|Q,R=1) can be improved for the current query and future doc

Per-query estimation!

Feedback in language models

- Recap of language model
  - Rank documents based on *query likelihood*

$$\log p(q|d) = \sum_{w_i \in q} \log \boxed{p(w_i|d)}$$

Document language model

$$where,\ \ q = w_1 w_2 \ldots w_n$$

  - Difficulty
    - Documents are given, i.e., $p(w|d)$ is fixed

Feedback in language models

- Approach
  - Introduce a probabilistic query model
  - Ranking: measure distance between query model and document model
  - Feedback: query model update

Q: Back to vector space model?
A: Kind of, but in different perspective.

Kullback-Leibler (KL) divergence based retrieval model

- Probabilistic similarity measure
  - $sim(q; d) \propto -KL(\theta_q || \theta_d)$

$$-\boxed{\sum_w p(w|\theta_q)\log p(w|\theta_q)} + \sum_w \boxed{p(w|\theta_q)} \log \boxed{p(w|\theta_d)}$$

Query-specific quality, ignored for ranking

Query language model, need to be estimated

Document language model, we know how to estimate

Background knowledge

- Kullback-Leibler divergence
  – A non-symmetric measure of the difference between two probability distributions P and Q
  – $KL(P||Q) = \int P(x) \log \frac{P(x)}{Q(x)} dx$
    - It measures the expected number of extra bits required to code samples from P when using a code based on Q
    - P usually refers to the "true" data distribution, Q refers to the "approximated" distribution
  – Properties
    - Non-negative
    - $KL(P||Q) = 0$, iff $P = Q$ almost everywhere

Explains why $sim(q; d) \propto -D(\theta_q||\theta_d)$

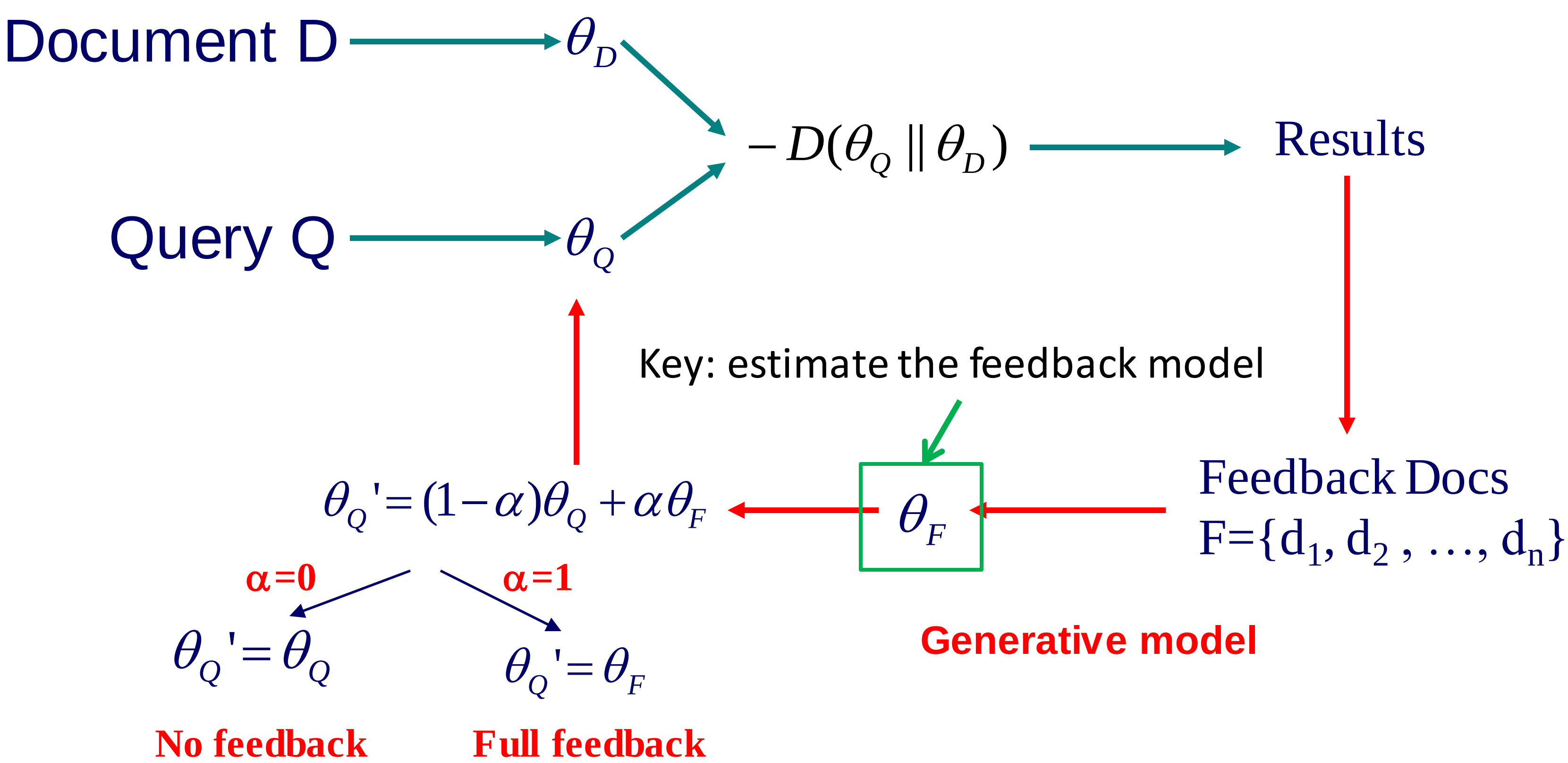
Kullback-Leibler (KL) divergence based retrieval model

- Retrieval ≈ estimation of θ_q and θ_d
  - $sim(q;d) \propto$

    same smoothing strategy

    $\sum_{w \in d, p(w|\theta_q)>0} p(w|\theta_q) \boxed{\log \frac{p(w|d)}{\alpha_d p(w|C)} + \log \alpha_d}$
  - A generalized version of query-likelihood language model
    - $p(w|\theta_q)$ is the empirical distribution of words in a query

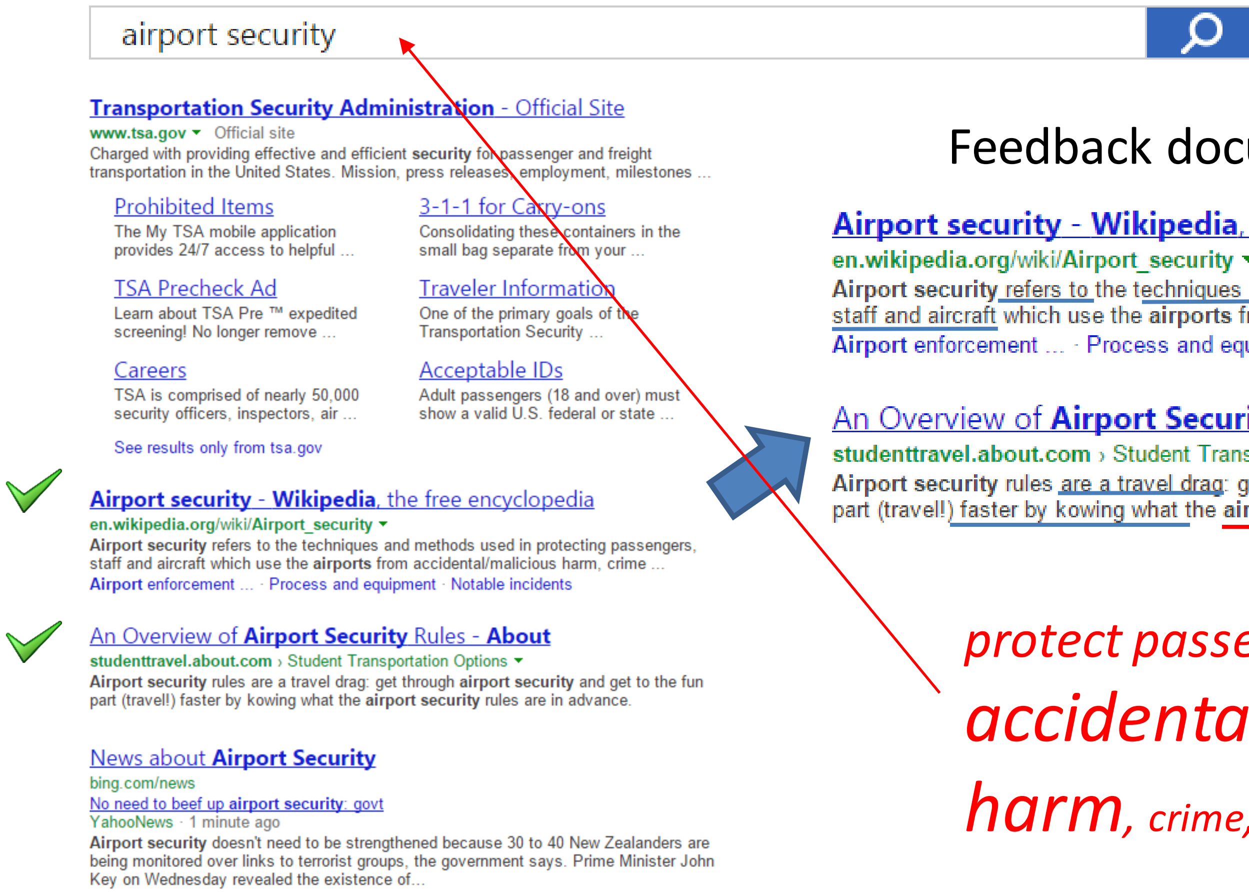
Feedback as model interpolation



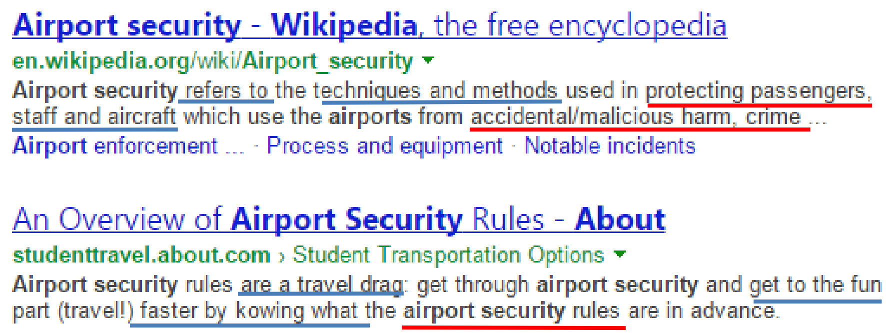
Q: Rocchio feedback in vector space model?

A: Very similar, but with different interpretations.

Feedback in language models



Feedback documents



protect passengers,
accidental/malicious
harm, crime, rules

Generative mixture model of feedback

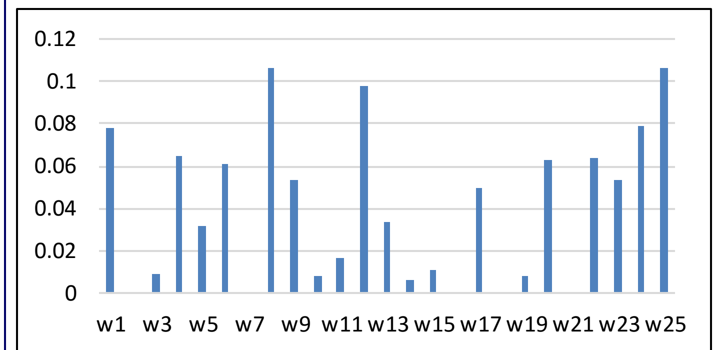
Background words

λ P(feedback) → $P(w|C)$

$1-\lambda$ P(feedback) → $P(w|\theta_F)$

Topic words

$F=\{d_1, \ldots, d_n\}$



$$\log p(d_F) = \sum_{d,w} c(w,d) \log[(1-\lambda)p(w|\theta_F) + \lambda p(w|C)]$$

λ = Noise ratio in feedback documents

Maximum Likelihood $\bar{\theta}_F = argmax_{\theta} \log p(d_F)$

How to estimate θ_F?

Known
Background
p(w|C)

the 0.2
a 0.1
we 0.01
to 0.02
…
flight 0.0001
company 0.00005
…

Unknown
query topic
$p(w|\theta_F)$=?

"airport security"

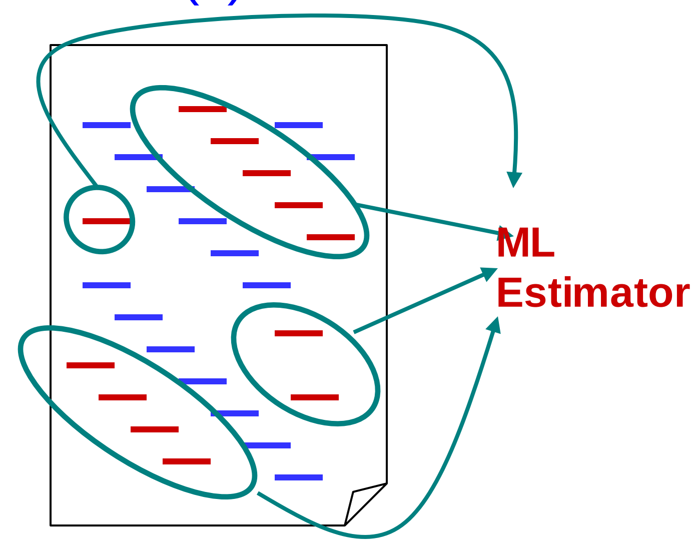
…
accident =?
regulation =?
passenger=?
rules =?
…

fixed

λ=0.7

λ=0.3

Feedback
Doc(s)



Suppose, we know the identity of each word; but we don't…

Appeal to EM algorithm

Identity ("hidden") variable: $z_i \in \{1 \text{ (background)}, 0 \text{(topic)}\}$

	z_i
the	**1**
paper	**1**
presents	**1**
a	**1**
text	**0**
mining	**0**
algorithm	**0**
the	**1**
paper	**0**
...	**...**

Suppose the parameters are all known, what's a reasonable guess of z_i?

- depends on λ (why?)
- depends on $p(w|C)$ and $p(w|\theta_F)$ (how?)

$$p(z_i=1|w_i) = \frac{p(z_i=1)p(w_i|z_i=1)}{p(z_i=1)p(w_i|z_i=1)+p(z_i=0)p(w_i|z_i=0)}$$

$$= \frac{\lambda p(w_i|C)}{\lambda p(w_i|C)+(1-\lambda)p(w_i|\theta_F)} \qquad \text{E-step}$$

$$p^{new}(w_i|\theta_F) = \frac{c(w_i,F)(1-p^{(n)}(z_i=1|w_i))}{\sum_{w_j \in vocabulary} c(w_j,F)(1-p^{(n)}(z_j=1|w_j))} \qquad \text{M-step}$$

Why in Rocchio we did not distinguish a word's identity?

A toy example of EM computation

$$p^{(n)}(z_i=1|w_i)=\frac{\lambda p(w_i|C)}{\lambda p(w_i|C)+(1-\lambda)p^{(n)}(w_i|\theta_F)}$$

Expectation-Step:
Augmenting data by guessing hidden variables

$$p^{(n+1)}(w_i|\theta_F)=\frac{c(w_i,F)(1-p^{(n)}(z_i=1|w_i))}{\sum_{w_j\in vocabulary}c(w_j,F)(1-p^{(n)}(z_j=1|w_j))}$$

Maximization-Step
With the "augmented data", estimate parameters using maximum likelihood

Assume λ=0.5

Word	#	P(w|C)	Iteration 1		Iteration 2		Iteration 3	
			$P(w\|\theta_F)$	P(z=1)	$P(w\|\theta_F)$	P(z=1)	$P(w\|\theta_F)$	P(z=1)
The	4	0.5	**0.25**	0.67	**0.20**	0.71	**0.18**	0.74
Paper	2	0.3	**0.25**	0.55	**0.14**	0.68	**0.10**	0.75
Text	4	0.1	**0.25**	0.29	**0.44**	0.19	**0.50**	0.17
Mining	2	0.1	**0.25**	0.29	**0.22**	0.31	**0.22**	0.31
Log-Likelihood			-16.96		-16.13		-16.02	

Example of feedback query model

Open question: how do we handle negative feedback?

- Query: “airport security”
  - Pesudo feedback with top 10 documents

λ=0.7

W	p(W| θ_F)
the	0.0405
security	0.0377
airport	0.0342
beverage	0.0305
alcohol	0.0304
to	0.0268
of	0.0241
and	0.0214
author	0.0156
bomb	0.0150
terrorist	0.0137
in	0.0135
license	0.0127
state	0.0127
by	0.0125

λ=0.9

W	p(W| θ_F)
security	0.0558
airport	0.0546
beverage	0.0488
alcohol	0.0474
bomb	0.0236
terrorist	0.0217
author	0.0206
license	0.0188
bond	0.0186
counter-terror	0.0173
terror	0.0142
newsnet	0.0129
attack	0.0124
operation	0.0121
headline	0.0121

What you should know

- Purpose of relevance feedback
- Rocchio relevance feedback for vector space models
- Query model based feedback for language models