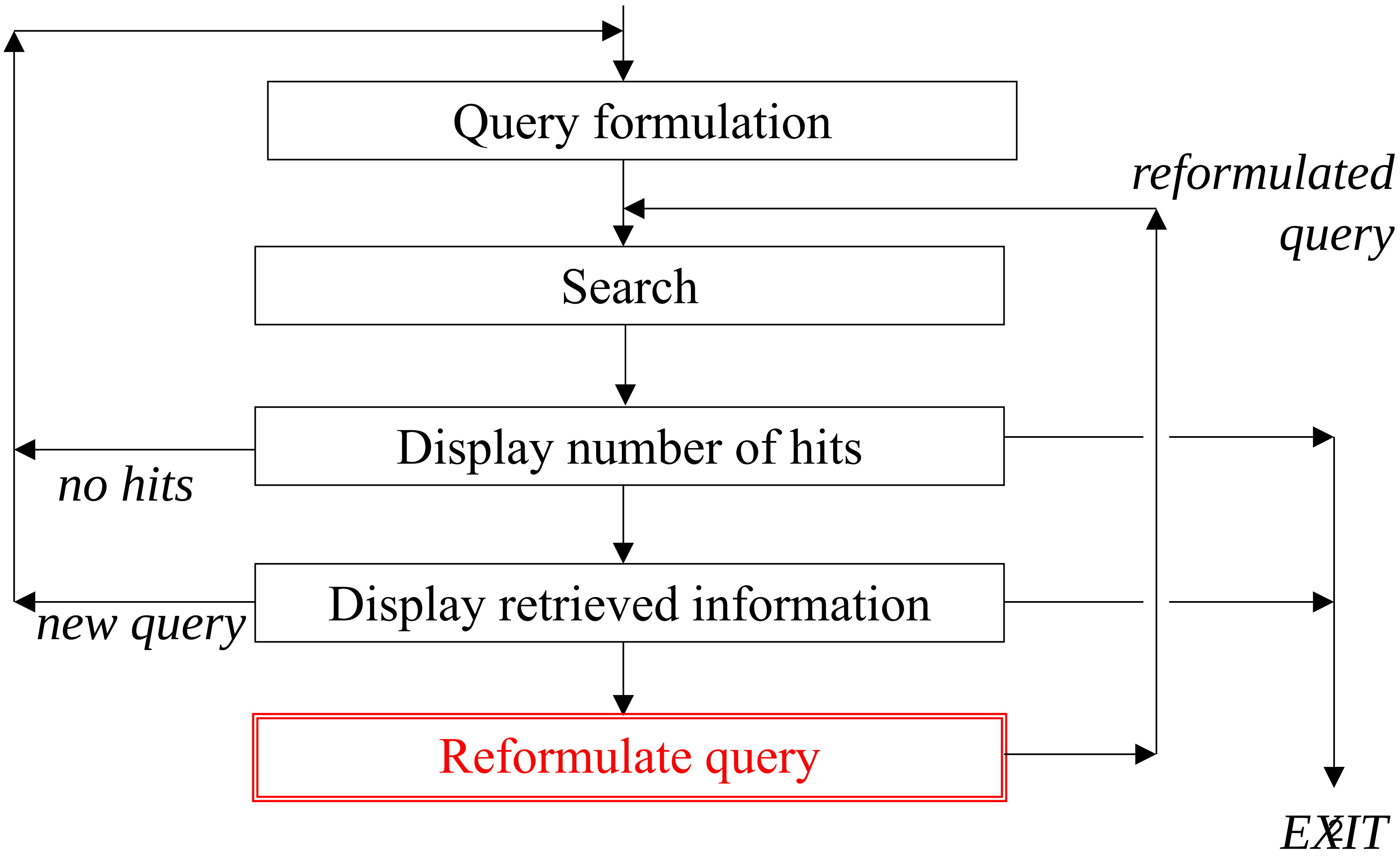
Lecture 9

Query Refinement and Relevance Feedback

Reference:

[Gerald Benoit, Simmons College]

Query Refinement



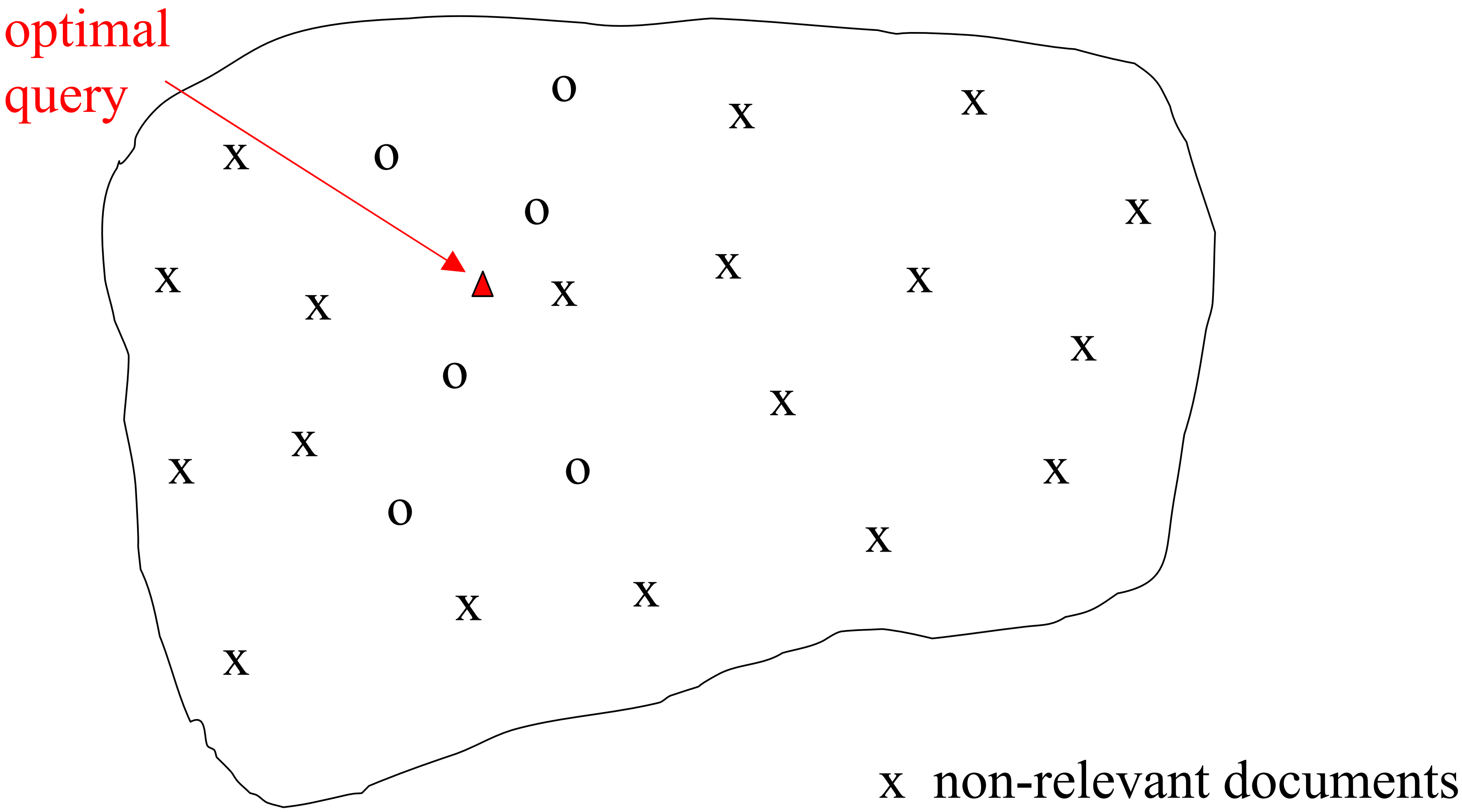
Relevance Feedback: Motivation

- Observations:
  - **A Query only approximates an information need and exactly match the information need is difficult**
  - **Users often start with short queries (poor approximations)**
  - ***People* can improve queries after seeing relevant and non-relevant documents**
    - by adding and removing terms
    - by reweighting terms
    - by adding structure (AND, OR, NOT, PHRASE, etc)
- Question: **Can a better query be created *automatically* by analyzing relevant and nonrelevant documents?**

Types of Relevance Feedback

- “Real” relevance feedback
  - System returns results
  - User provides some feedback
  - System returns different—better, we hope—results
- “Assumed” relevance feedback
  - System gets results but does not return them
  - Uses returned results to “guess” what was probably meant
  - Modifies query without supervision
  - System returns enhanced—and we hope better—result list
- Occurs in different models
  - Vector space is used most often (we’ll focus on it)
  - Language modeling
    - Excellent success with “assumed” relevance (relevance models)
    - Less obviously good results for “real” feedback

Theoretically Best Query



Theoretically Best Query

For a specific query, Q, let:

D_R be the set of all relevant documents

D_{N-R} be the set of all non-relevant documents

$sim(Q, D_R)$ be the mean similarity between query Q and documents in D_R

$sim(Q, D_{N-R})$ be the mean similarity between query Q and documents in D_{N-R}

The theoretically best query would maximize:

$$F = sim(Q, D_R) - sim(Q, D_{N-R})$$

Estimating the Best Query

In practice, D_R and $D_{N\text{-}R}$ are not known. (The objective is to find them.)

However, the results of an initial query can be used to estimate $sim(Q, D_R)$ and $sim(Q, D_{N\text{-}R})$.

Rocchio's Modified Query

Modified query vector

= Original query vector

+ Mean of ***relevant*** documents found by original query

- Mean of ***non-relevant*** documents found by original query

Query Modification

$$\boldsymbol{Q}_1 = \boldsymbol{Q}_0 + \frac{1}{n_1}\sum_{i=1}^{n_1}\boldsymbol{R}_i - \frac{1}{n_2}\sum_{i=1}^{n_2}\boldsymbol{S}_i$$

$\boldsymbol{Q}_0$ = vector for the initial query
$\boldsymbol{Q}_1$ = vector for the modified query
$\boldsymbol{R}_i$ = vector for relevant document i
$\boldsymbol{S}_i$ = vector for non-relevant document i
n_1 = number of relevant documents
n_2 = number of non-relevant documents

Rocchio 1971

Adjusting Parameters 1: Relevance Feedback

$$\boldsymbol{Q}_1 = \alpha\, \boldsymbol{Q}_0 + \beta \frac{1}{n_1} \sum_{i=1}^{n_1} \boldsymbol{R}_i - \gamma \frac{1}{n_2} \sum_{i=1}^{n_2} \boldsymbol{S}_i$$

α, β and γ are weights that adjust the importance of the three vectors.

If γ = 0, the weights provide **positive feedback**, by emphasizing the relevant documents in the initial set.

If β = 0, the weights provide **negative feedback**, by reducing the emphasis on the non-relevant documents in the initial set.

Relevance Feedback in the Vector Space: Example

Original Query:

(5, 0, 3, 0, 1)

Document D1, Relevant:

(2, 1, 2, 0, 0)

Document D2, Non-relevant:

(1, 0, 0, 0, 2)

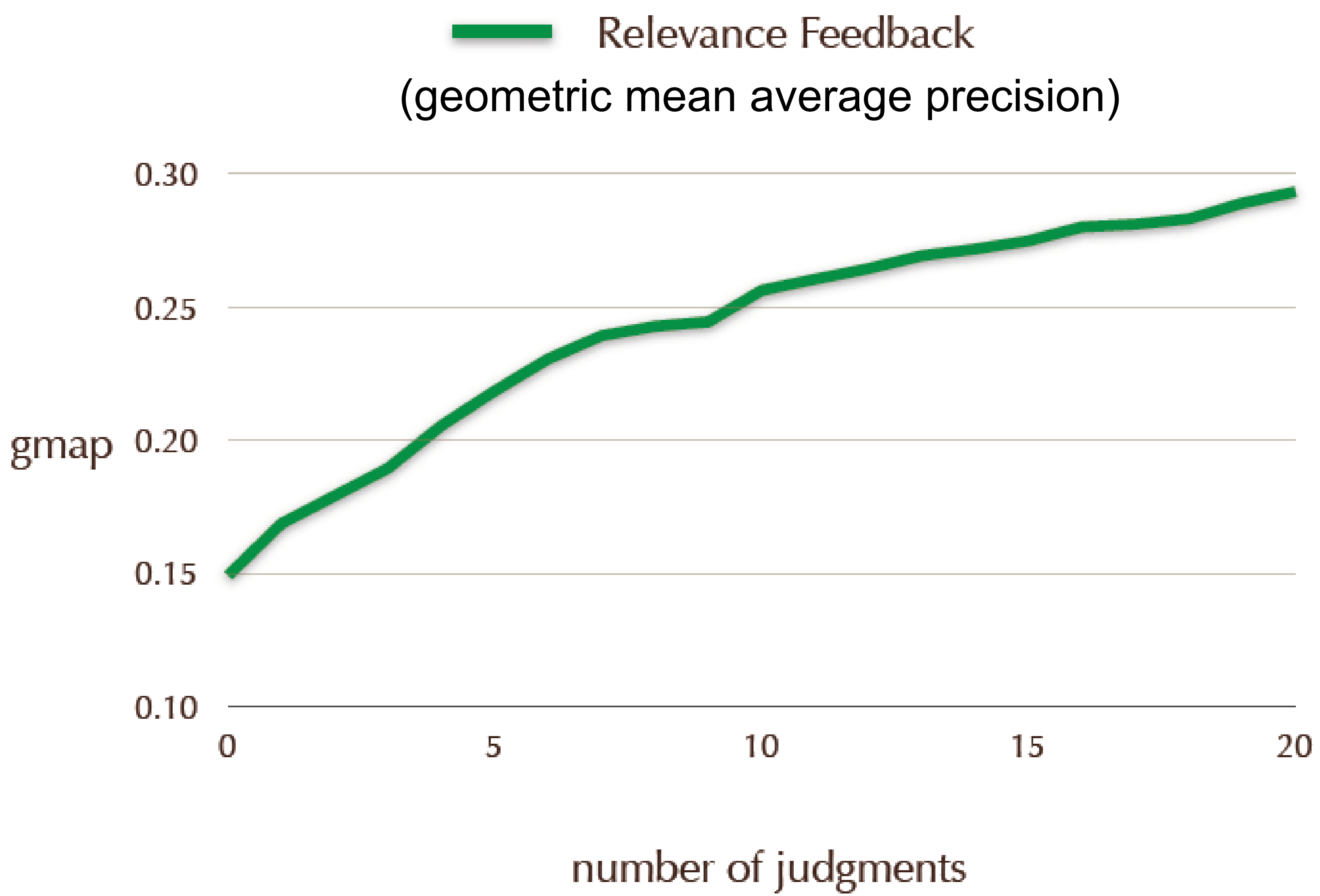
$$\alpha = 1, \beta = 0.5, \lambda = 0.25$$

$$\begin{bmatrix} 5.75 \\ 0.50 \\ 4.00 \\ 0.00 \\ 0.50 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \\ 3 \\ 0 \\ 1 \end{bmatrix} + 0.50 \begin{bmatrix} 2 \\ 1 \\ 2 \\ 0 \\ 0 \end{bmatrix} - 0.25 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 2 \end{bmatrix}$$

Q'= Q + 0.5 D1 - 0.25 D2

= (5, 0, 3, 0, 1) + 0.5 (2, 1, 2, 0, 0) - 0.25 (1, 0, 0, 0, 2)

= (5.75, 0.50, 4.00, 0.0, 0.5)



[Díaz and Metzler 06]

Relevance Feedback: Clickthrough Data

Relevance feedback methods have suffered from the unwillingness of users to provide feedback.

Joachims and others have developed methods that use Clickthrough data from online searches.

Concept:

Suppose that a query delivers a set of hits to a user.

If a user skips a link *a* and clicks on a link *b* ranked lower, then the user preference reflects $rank(b) < rank(a)$.

Clickthrough Example

Ranking Presented to User:

User clicks on 1, 3 and 4

1. Kernel Machines
   http://svm.first.gmd.de/
2. Support Vector Machine
   http://jbolivar.freeservers.com/
3. SVM-Light Support Vector Machine
   http://ais.gmd.de/~thorsten/svm light/
4. An Introduction to Support Vector Machines
   http://www.support-vector.net/
5. Support Vector Machine and Kernel ... References
   http://svm.research.bell-labs.com/SVMrefs.html

Ranking: (3 < 2) and (4 < 2)

Joachims

Relevance Feedback: Assumed

- True relevance feedback is supervised
  - Feedback is done based on *genuine* user annotations
- What happens if we try to guess what is relevant?
  - Assume many top ranked documents are relevant
    - Optionally find a collection of probably non-relevant documents
  - Modify query on that assumption
  - Re-run that new query and show results to user
  - What happens?
- Pseudo-relevance feedback
  - Blind relevance feedback
  - Local feedback
  - …

Local Context Analysis (LCA)

[Xu and Croft, 1996]

- Assumed relevance feedback
- Major focus is on getting better terms for expansion
  - Finding terms to consider
  - Selection of terms
  - Weighting of selected terms

Finding candidate terms

- Run query to retrieve passages
  - Similar to most “assumed” relevance work
  - Passage-retrieval
    - Minimizes spurious（欺骗性的） concepts that occur in lengthy documents
  - Uses 300-word passages
- Select expansion concepts from retrieved set

Selecting candidate terms

- Parse document collection
- Generate part of speech tagging
  - The/AT bill/NN has/HVZ been/BEN reworked/VBN since/CS it/PPS was/BEDZ introduced/VBN ,/, in/IN order/NN to/TO meet/VB some/DTI employer/NN objections/NNS ./. But/CC the/AT measure/NN still/RB is/BEZ opposed/VBN by/IN the/AT construction/NN industry/NN ,/, which/WDT argues/VBZ that/CS it/PPS would/MD impose/VB unionism/NN and/CC higher/JJ costs/NNS on/IN much/AP of/IN the/AT industry/NN 's/$ work/NN ./.
- Select only noun phrases
  - Shown to be critical in most retrieval systems
  - Generally particularly useful for expansion
  - Could easily be extended if useful
    - Adjective-noun phrases, verbs, …
  - Note that tagging is automated, so makes mistakes!

Weighting terms

- Want “concepts” that occur near query words
  - The more query words they occur near, the better
  - Count co-occurrences in 300-word windows of text (passages)
    - To avoid coincidental co-occurrence in a large document
- Uses the following ad-hoc function to weight concepts
- Here *N* is the number of passages,

$$
\begin{aligned}
f(c,Q) &= \prod_{w_i \in Q} (0.01 + \text{co\_degree}(c,w_i))^{\overbrace{idf(w_i)}^{\text{Importance of word}}} \\
\text{co\_degree}(c,w) &= \max\left(\frac{n_{cw} - En(c,w) - 1}{n_c}, 0\right) \\
En(c,w) &= \frac{n_w n_c}{N} \\
idf(w) &= \min(1.0, \log_{10}(N/n_w)/5)
\end{aligned}
$$

Importance of word

Measure co-occurrence

Floor the IDF component

Slow its growth

- Dev
- Inco
  - V
- Vari
  - L
  - V ... ept

Figure 1 shows an example query expanded by local context analysis.

```
#WSUM(1
      1 #WSUM (1 1 status 1 nuclear 1 proliferation 1 treaties
                 1 violations 1 monitoring)
      2 #WSUM (1
               1        #PHRASE(nuclear non proliferation treaty)
               0.987143 treaty
               0.974286 weapon
               0.961429 pakistan
               0.948571 missile
               0.935714 iraq
               0.922857 proliferation
               0.91     #PHRASE(non proliferation treaty)
               0.897143 #PHRASE(international atomic energy agency)
               0.884286 india
               0.871429 warhead
               0.858571 uranium
               0.845714 disarmament
               0.832857 china
               0.82     #PHRASE(chemical weapon)
               0.807143 spread
               ...
               ))
```

Figure 1: Query expansion by local context analysis for TREC topic 202 "Status of nuclear proliferation treaties, violations and monitoring". #PHRASE is an INQUERY operator to construct phrases.

Example of expansion concepts

- T
- E

Does it work?

- TREC-3 and TREC-4 ad-hoc queries
- With and without LCA expansion

Precision (% change) – 50 queries

Recall	baseline	corpus-query
0	82.2	85.3 (+3.8)
10	57.3	65.1 (+13.5)
20	46.2	54.7 (+18.5)
30	39.1	46.8 (+19.9)
40	32.7	40.0 (+22.1)
50	27.5	34.6 (+25.9)
60	22.6	28.4 (+25.2)
70	18.0	23.0 (+27.3)
80	13.3	17.4 (+30.7)
90	7.9	10.7 (+34.4)
100	0.5	0.7 (+36.9)
average	31.6	37.0 (+17.0)

TREC-3

Precision (% change) – 49 queries

Recall	baseline	corpus-query
0	71.0	70.4 (−0.8)
10	49.3	54.3 (+10.0)
20	40.4	45.0 (+11.6)
30	33.3	37.7 (+13.4)
40	27.3	32.6 (+19.4)
50	21.6	27.4 (+26.7)
60	14.8	20.8 (+40.7)
70	9.5	13.6 (+43.3)
80	6.2	8.2 (+33.8)
90	3.1	4.2 (+34.2)
100	0.4	0.6 (+36.7)
average	25.2	28.6 (+13.7)

TREC-4

Summary

- Relevance feedback
  - Real or assumed
- Real relevance feedback
  - Usually improves effectiveness significantly
  - Not always stable with very few documents judged
  - Difficult to incorporate into a usable system
  - "Documents like this one" is a simple instance
- Assumed relevance feedback
  - Also called "pseudo relevance feedback" or "local feedback"
    - Or "quasi-relevance feedback" or …
  - Rocchio-based approaches effective but unstable
  - LCA comparably effective (maybe better) but more stable
  - Relevance models provide formal framework

Learning to Rank (a quick glimpse)

--Improving search performance by large amount of examples

Ref: partially based on
Pandu Nayak and Prabhakar Raghavan, Stanford University

Learning to Rank

Assume:

distribution of queries P(Q)

distribution of target rankings for query P(R | Q)

Given:

collection D of documents

independent, identically distributed training sample (q_i, r_i)

Design:

set of ranking functions F

loss function $l(r_a, r_b)$

learning algorithm

Goal:

find $f \in F$ that minimizes $\int l(f(q), r)\,dP(q, r)$

Joachims

Machine learning for IR ranking

- This “good idea” has been actively researched – and actively deployed by the major web search engines – in the last few years
- Why didn’t it happen earlier?
  - Modern supervised ML has been around for about 20 years…
  - Naïve Bayes has been around for about 50 years…

Why weren’t early attempts very successful/influential?

- Limited training data
  - Especially for real world use (as opposed to writing academic papers), it was very hard to gather test collection， queries and relevance judgments that are representative of real user needs and judgments on documents returned
    - This has changed, both in academia and industry
- Poor machine learning techniques
- Insufficient customization to IR problem
- Not enough features for ML to show value
- The Web provided impetus(动力) with constantly evolving spam

Why wasn’t ML much needed?

- Traditional ranking functions in IR used a very small number of features, e.g.,
  - Term frequency
  - Inverse document frequency
  - Document length
- It was easy to tune weighting coefficients by hand
  - And people did

Why is ML needed now

- Modern systems – especially on the Web – use a great number of features:
    - Arbitrary useful features – not a single unified model
  - Log frequency of query word in anchor text?
  - Query word in color on page?
  - # of images on page?
  - # of (out) links on page?
  - PageRank of page?
  - URL length?
  - URL contains “~” ?
  - Page edit recently?
  - Page length?
- Major web search engines publicly state that they use “hundreds” of such features – and they keep changing

Simple example

- Consider the presence of query terms in the Title (T) and the Body (B) of a document
  - Boolean indicator (0/1) of whether the query term occurs in the Title (s_T) or Body (s_B)
- We'll compute a score in [0,1] for each doc *d* and for each query *q* using a linear combination of s_T and s_B

$$score(d,q) = g s_T(d,q) + (1-g) s_B(d,q)$$

- Thus our scores are all 0, g, 1-g or 1.
- g is a parameter to be learned from examples

We are given examples

- Created by human judges

Example	DocID	Query	s_T	s_B	Judgment
Φ_1	37	linux	1	1	Relevant
Φ_2	37	penguin	0	1	Non-relevant
Φ_3	238	system	0	1	Relevant
Φ_4	238	penguin	0	0	Non-relevant
Φ_5	1741	kernel	1	1	Relevant
Φ_6	2094	driver	0	1	Relevant
Φ_7	3191	driver	1	0	Non-relevant

- We quantize the human relevance judgments to be 1 or 0 respectively, for Relevant and Non-relevant
  - The scores we compute will be 0, g, 1-g or 1 – how do we tell how good our scoring function is?

Least square errors

- For each human-judged example, we compute its score:

$$score(d_j, q_j) = g s_T(d_j, q_j) + (1-g) s_B(d_j, q_j)$$

- Then we can compute a total error of the squared errors defined as:

$$\varepsilon(g, \Phi_j) = (r(d_j, q_j) - score(d_j, q_j))^2$$

- We will pick *g* to minimize the total error.

Choosing *g*

- In our simple setting, all that matters is the number of examples* of each equivalence class
- Define:
- n01r = # examples with s_T=0, s_B=1, judgment = Rel
- n01n = # examples with s_T=0, s_B=1, judgment = NonRel
- n10r = # examples with s_T=1, s_B=0, judgment = Rel
- n10n = # examples with s_T=1, s_B=0, judgment = NonRel
- (and similarly n00r, n00n, n11r and n11n corresponding to the 4 other equivalence classes)

* this may not hold for other sets of features, e.g., the # characters in the query

Choosing *g*

- The n01 examples with s_T=0, s_B=1 combined contribute a total least-squared error of

$$[1-(1-g)]^2 n_{01r} + [0-(1-g)]^2 n_{01n}$$

- Similarly, add up the error contributions of the other 3 combinations of s_T and s_B for a total error of

$$(n_{01r} + n_{10n}) g^2 + (n_{10r} + n_{01n})(1-g)^2 + n_{00r} + n_{11n}$$

Choosing *g* is now elementary calculus

- Differentiating the total error *wrt* *g* we get the optimal value for *g* to be

$$\frac{n_{10r} + n_{01n}}{n_{10r} + n_{10n} + n_{01r} + n_{01n}}$$

Generalizing this simple example

- More (than 2) features
- Non-Boolean features
  - What if the title contains some but not all query terms …
  - Categorical features (query terms occur in plain, boldface, italics, etc)
- Scores are nonlinear combinations of features
- Multilevel relevance judgments (Perfect, Good, Fair, Bad, etc)
- Complex error functions
- Not always a unique, easily computable setting of score parameters

Machine Learning: Algorithms

The choice of algorithms is a subject of active research.

Some effective methods include:

- Naïve Bayes
- Rocchio Algorithm
- C4.5 Decision Tree [popular in OLAP]
- Neural Networks [feed forward; back-propagation]
- Genetic Algorithms [evolutionary]
- k-Nearest Neighbors [image recognition]
- Support Vector Machine
- Deep Learning

Issues with Machine Learning Approaches

- very unbalanced class distribution
  – number of relevant documents is very small compared to non-relevant documents
- difficult to model non-relevant class
- machine learning approaches do not scale well (NN for billions of documents?)

Homework 8