Optimal Query

- Assume that the relevant set of documents C_r are known.
- Then the best query is:

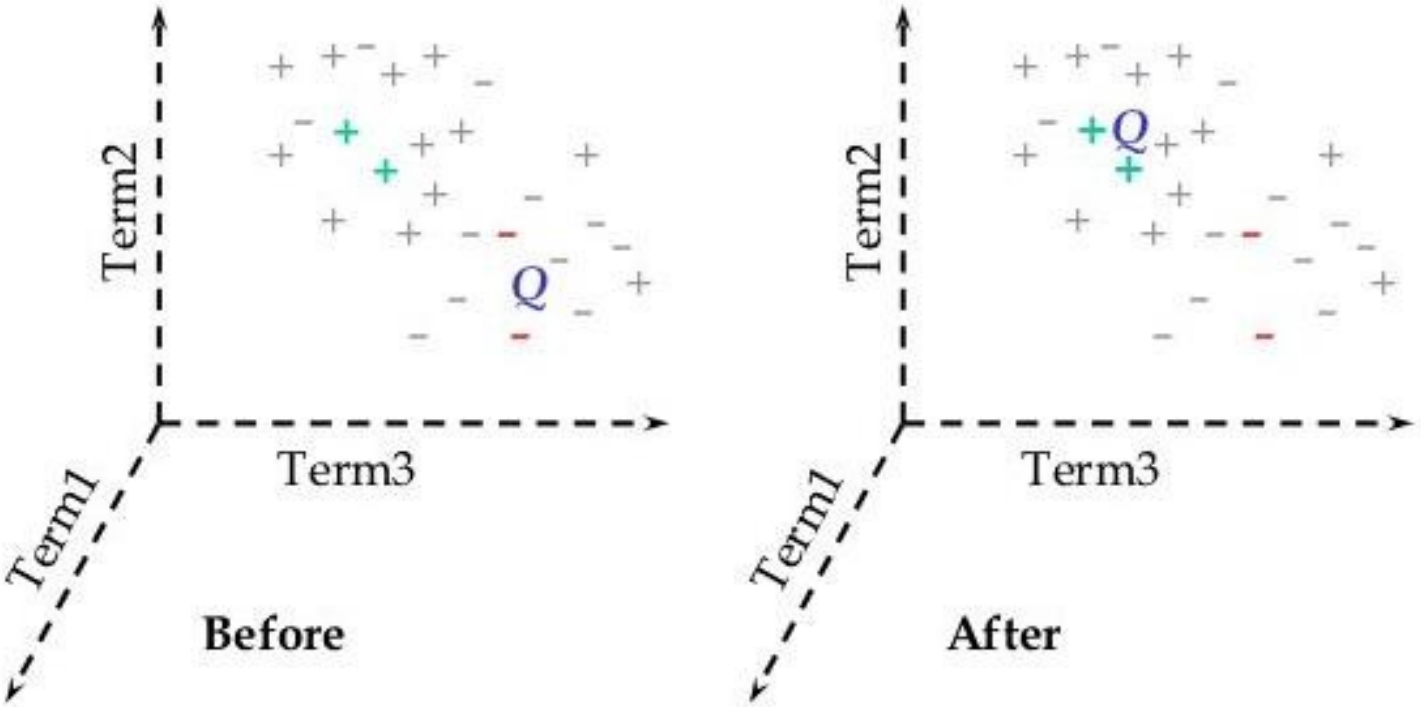
$$\vec{q}_{opt} = \frac{1}{|C_r|} \sum_{\forall \vec{d}_j \in C_r} \vec{d}_j - \frac{1}{N - |C_r|} \sum_{\forall \vec{d}_j \notin C_r} \vec{d}_j$$

Where N is the total number of documents.

Note that even this query may return some irrelevant documents if the relevant and irrelevant are not linearly separable

Relevance Feedback in the Vector Space

How can relevance feedback save time if a person has to read documents?



Relevance Feedback

- Hypothesis: initial query only approximates information need (i.e., the optimal query)
- So:

  1)Get an initial query

  2)Retrieve documents according to query

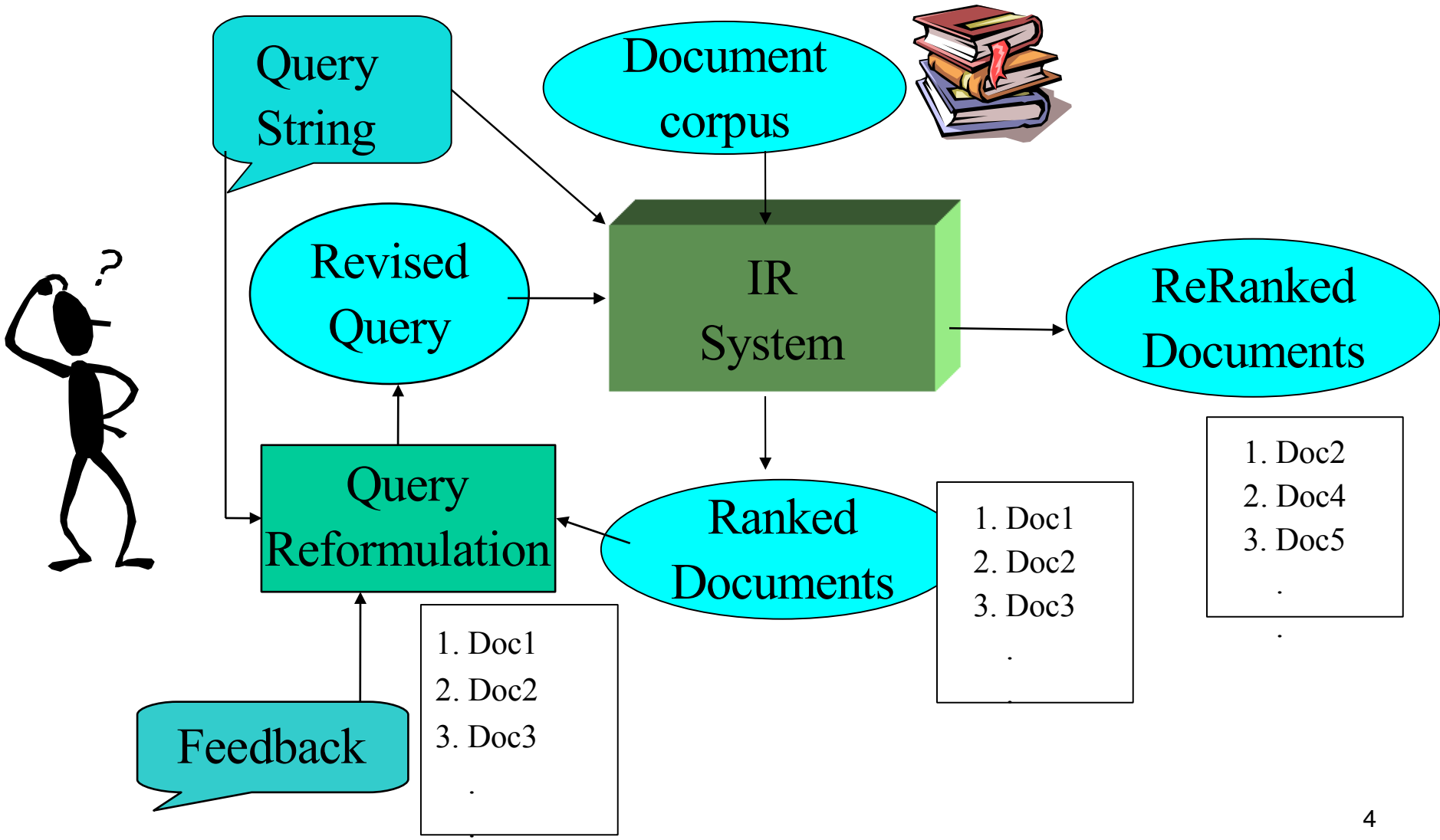
  3)Get indication of [ir]relevant documents

  4)Reformulate the query.

  5)Goto step 2

- Assumes
  - relevance judgements are binary
  - Other things?

Relevance Feedback Architecture



Standard Rochio Method

- Since all relevant documents unknown, just use the **known** relevant (D_r) and irrelevant (D_n) sets of documents and include the initial query q.

$$\vec{q}_m = \hat{I}\pm \quad \vec{q} + \frac{\hat{I}^2}{|D_r|} \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \frac{\hat{I}^3}{|D_n|} \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j$$

Î: Tunable weight for initial query.

$\hat{I}^2$:Tunable weight for relevant documents.

$\hat{I}^3$: Tunable weight for irrelevant documents.

Improving Rocchio

- Ide Regular Method
- Idea -- since more feedback should perhaps increase the degree of reformulation, do not normalize for amount of feedback:

$$\vec{q}_m = \hat{I}\pm \quad \vec{q} + \hat{I}^2 \quad \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \hat{I}^3 \quad \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j$$

Î: Tunable weight for initial query.

$\hat{I}^2$:Tunable weight for relevant documents.

$\hat{I}^3$: Tunable weight for irrelevant documents.

More Improvements

- Ide “dec Hi” msthod
- Bias towards rejecting **just** the highest ranked of the irrelevant documents:

$$\vec{q}_m = \hat{I}\pm \qquad \vec{q} + \hat{I}^2 \qquad \sum_{\forall \dot{d}_j \dot{=} D_r} \vec{d}_j - \hat{I}^3 \qquad \max_{non-relevant} \left| \vec{d}_j \right|$$

Î: Tunable weight for initial query.

$\hat{I}^2$:Tunable weight for relevant documents.

$\hat{I}^3$: Tunable weight for irrelevant documents.

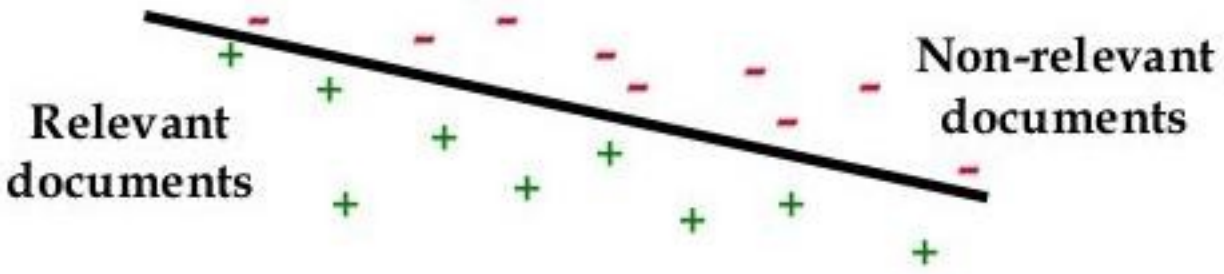
How is this like “query zoning” from Schapire's paper?

Yet More Improvements – Schapire's 3

- Improved term weighting
  - Replace standard tf-idf weights with weights based on a probabilistic model of word usage *in context*. The words get credit for "interesting usage" not simply appearance. (Stop words are rarely used in interesting ways)
- Query Zoning
  - Idea is to use only those examples marked as irrelevant that are close to the relevant
  - i.e., near misses
- Dynamic Feedback Optimization
  - Change term weights using feedback so the values of words change with feedback

Relevance Feedback: Machine Learning Approaches

- An unstructured vector query is a linear discriminator
  - e.g., $w_1 * t_1 + w_2 * t_2 + \ldots + w_n * t_n$
- The goal is to learn weights that separate the relevant documents from the non-relevant documents



- If the documents are *linearly separable*, a learning algorithm can be chosen that is guaranteed to converge to an optimal query
- If the documents are not linearly separable, a learning algorithm can be chosen that minimizes the total amount of error

Scharpire's weak learners

- Derive a “rule of thumb” which is a single word or phrase that distinguishes that can be used with some reliability to split yes from no.
- For example, if trying to sort cs pages from non-cs pages a weak hypothesis might be the presence of word “computer”

AdaBoost

Input:
N documents and labels: $\langle (d_1, y_1), \ldots, (d_N, y_N) \rangle$ where $y_i \in \{-1, +1\}$;
integer T specifying number of iterations

Initialize $D_1(i)$ (for classification error, $D_1(i) = 1/N$)
Do for $s = 1, 2, \ldots, T$:

1. Call WeakLearn and get a weak hypothesis h_s

2. Calculate the error of h_s: $\epsilon_s = \sum_{i:h_s(d_i) \neq y_i} D_s(i)$.

3. Set $\alpha_s = \frac{1}{2} \ln \left(\frac{1 - \epsilon_s}{\epsilon_s} \right)$.

4. Update distribution:

$$D_{s+1}(i) = \frac{D_s(i) \exp(-\alpha_s y_i h_s(d_i))}{Z_s}$$

$$= \frac{D_s(i)}{Z_s} \times \begin{cases} e^{-\alpha_s} & \text{if } h_s(d_i) = y_i \\ e^{\alpha_s} & \text{if } h_s(d_i) \neq y_i \end{cases}$$

where Z_s is a normalization factor.

Output the final hypothesis:

$$h_{fin}(d) = \text{sign} \left(\sum_{s=1}^{T} \alpha_s h_s(d) \right).$$

Comparison of Methods

- Overall, experimental results indicate no clear preference for any one of the specific methods.
- All methods generally improve retrieval performance (recall & precision) with feedback.
- Generally just let tunable constants equal 1.

Comparing Rocchio+ to AdaBoost

- Results are about the same

AdaBoost $\sum_{s=1}^{T} alpha_s * h_s(d)$

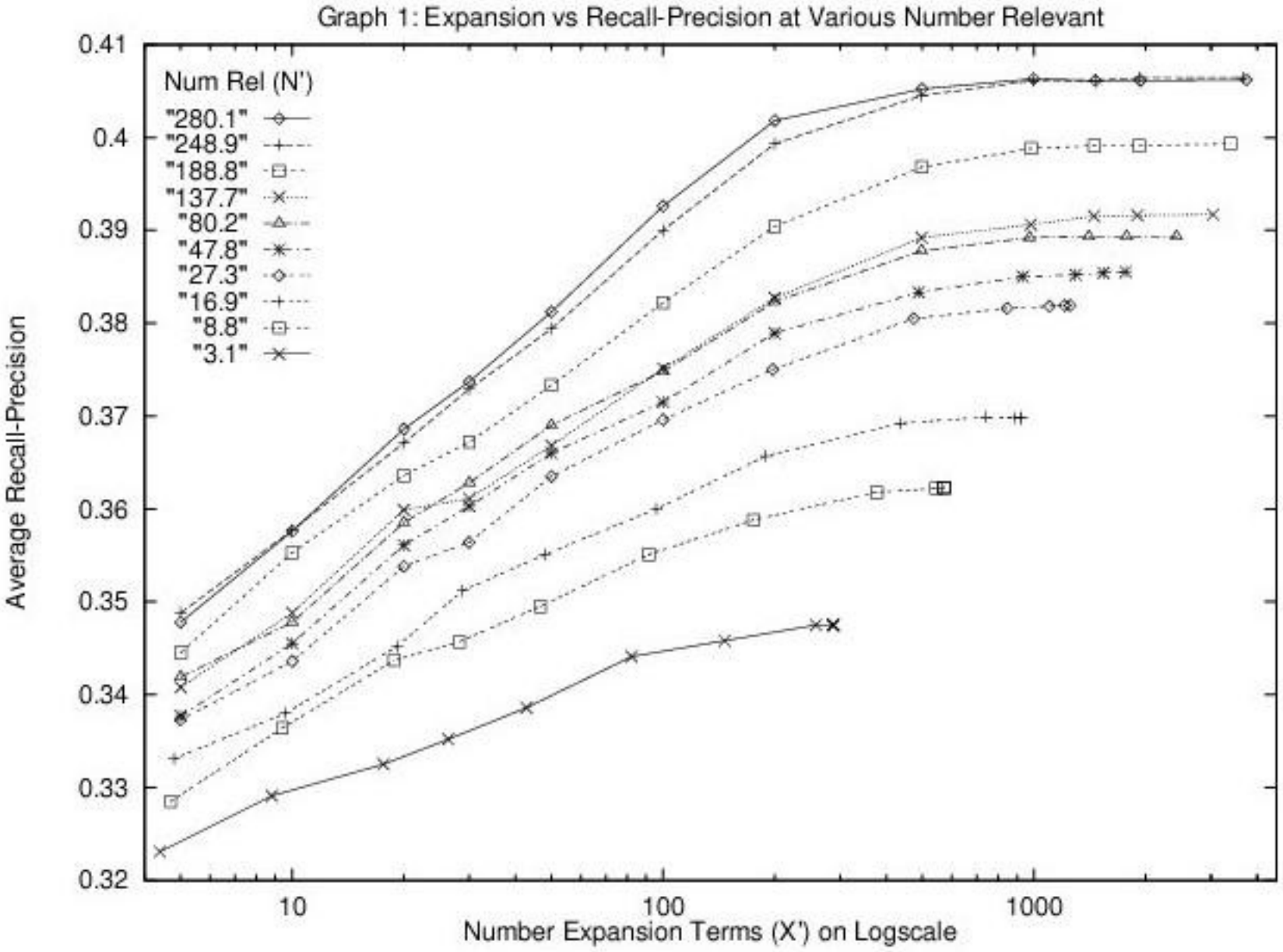
Rocchio $h_s \times d$

Where h_s is a weak learner hypothesis for adaboost and a roochio vector space for Rocchio.

But, importantly both methods learn a single separating hyperplane Rocchio will tend to have lots of small, nonzero terms whereas adaboost will have few nonzero terms. But they learn a fundamentally similar function, so it is not surprising that they are about equally effective

Evaluating Relevance Feedback

- By construction, reformulated query will rank explicitly-marked relevant documents higher and explicitly-marked irrelevant documents lower.
- Method should not get credit for improvement on *these* documents, since it was told their relevance.
- In machine learning, this error is called "testing on the training data."
- Evaluation should focus on generalizing to **other** un-rated documents.


Graph 1: Expansion vs Recall-Precision at Various Number Relevant
Num Rel (N')
"280.1"
"248.9"
"188.8"
"137.7"
"80.2"
"47.8"
"27.3"
"16.9"
"8.8"
"3.1"
Average Recall-Precision
0.41
0.4
0.39
0.38
0.37
0.36
0.35
0.34
0.33
0.32
10
100
1000
Number Expansion Terms (X') on Logscale

Fair Evaluation of Relevance Feedback

- Remove from the corpus any documents for which feedback was provided.
- Measure recall/precision performance on the remaining *residual collection*.
- Compared to complete corpus, specific recall/precision numbers may decrease since relevant documents were removed.
- However, **relative** performance on the residual collection provides fair data on the effectiveness of relevance feedback.

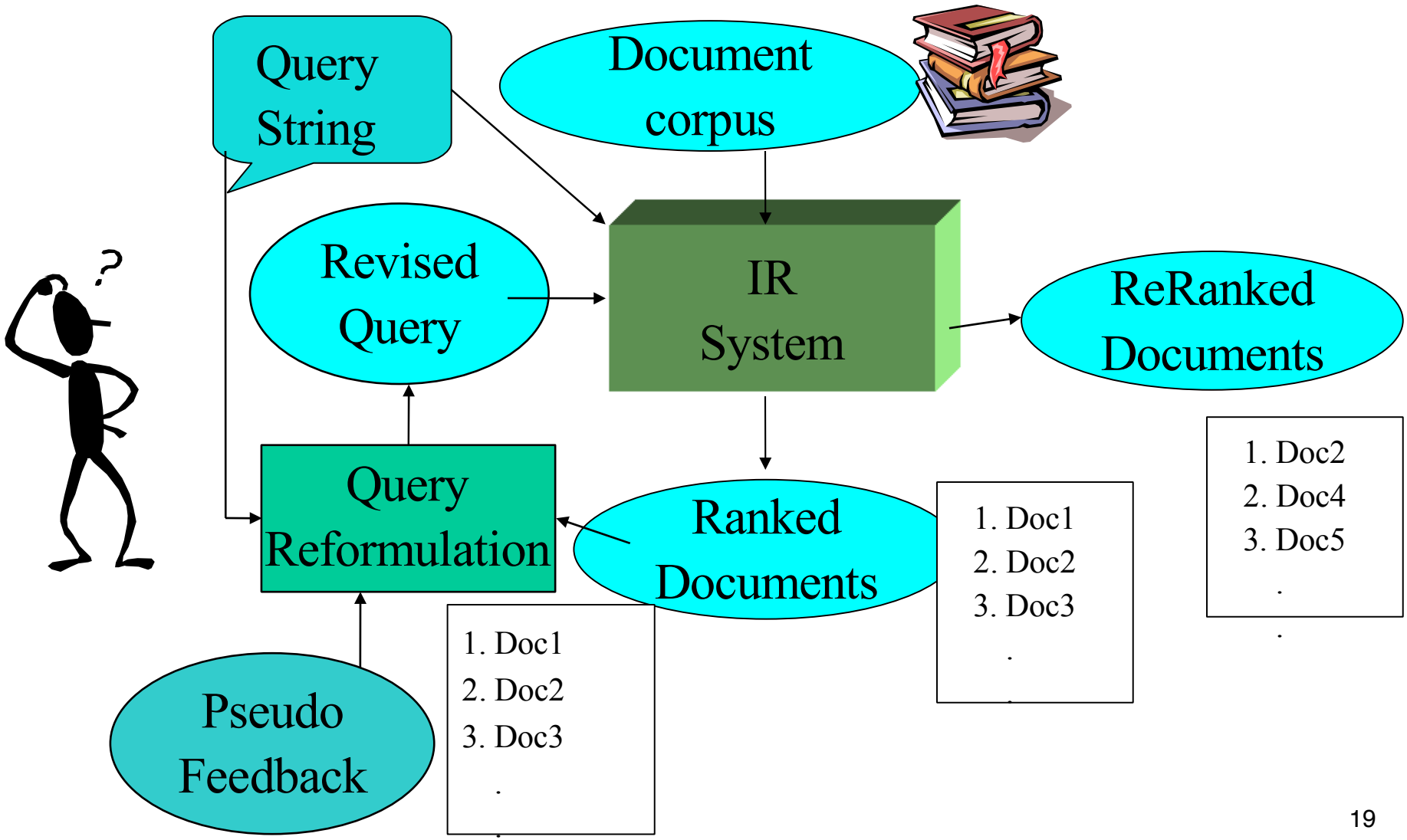
Why is Feedback Not Widely Used

- Users sometimes reluctant to provide explicit feedback.
- Results in long queries that require more computation to retrieve, and search engines process lots of queries and allow little time for each one.
- Makes it harder to understand why a particular document was retrieved.

Pseudo Feedback

- Use relevance feedback methods without explicit user input.
- Just **assume** the top *m* retrieved documents are relevant, and use them to reformulate the query.
- Allows for query expansion that includes terms that are correlated with the query terms.

Pseudo Feedback Architecture



PseudoFeedback Results

- Found to improve performance on TREC competition ad-hoc retrieval task.
- Works even better if top documents must also satisfy additional boolean constraints in order to be used in feedback.

Thesaurus-based Query Expansion

- For each term, *t*, in a query, expand the query with synonyms and related words of *t* from the thesaurus.
- May weight added terms less than original query terms.
- Generally increases recall.
- May significantly decrease precision, particularly with ambiguous terms.
  - "interest rate" "interest rate fascinate evaluate"

WordNet

- A more detailed database of semantic relationships between English words.
- About 144,000 English words.
- Nouns, adjectives, verbs, and adverbs grouped into about 109,000 synonym sets called *synsets*.
- http://www.cogsci.princeton.edu/cgi-bin/webwn

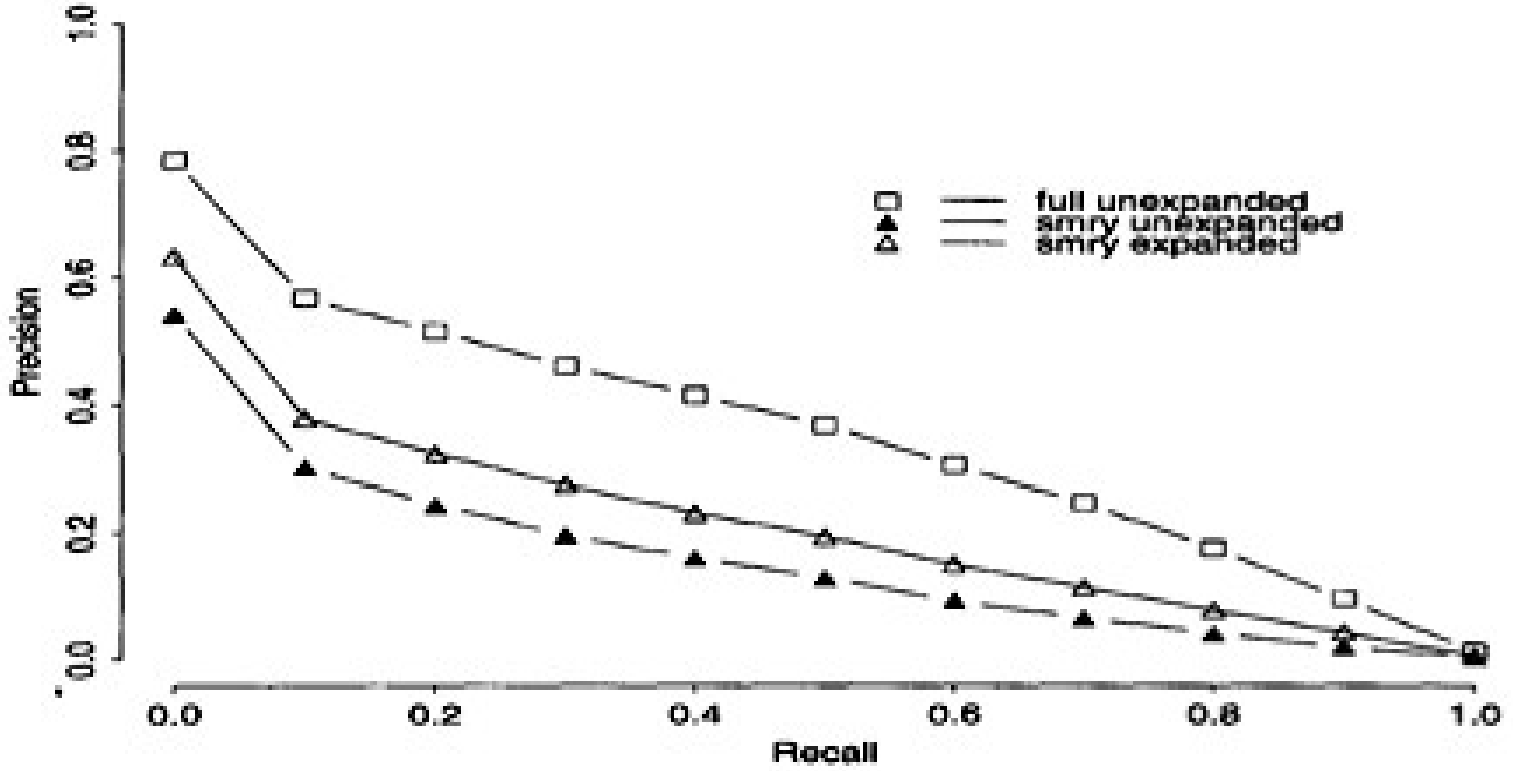
WordNet Synset Relationships

- Antonym: front − back
- Attribute: benevolence − good (noun to adjective)
- Pertainym: alphabetical − alphabet (adjective to noun)
- Similar: unquestioning − absolute
- Cause: kill − die
- Entailment: breathe − inhale
- Holonym: chapter − text (part-of)
- Meronym: computer − cpu (whole-of)
- Hyponym: tree − plant (specialization)
- Hypernym: fruit − apple (generalization)

WordNet Query Expansion

- Add synonyms in the same synset.
- Add hyponyms to add specialized terms.
- Add hypernyms to generalize a query.
- Add other related terms to expand query.

Wordnet Expansion Results



From Voorhees (1994) the effectiveness of hand-selected WordNet synsets on precision and recall of 50 TREC queries

Query Expansion Conclusions

- Expansion of queries with related terms can improve performance, particularly recall.
- However, must select similar terms very carefully to avoid problems, such as loss of precision.