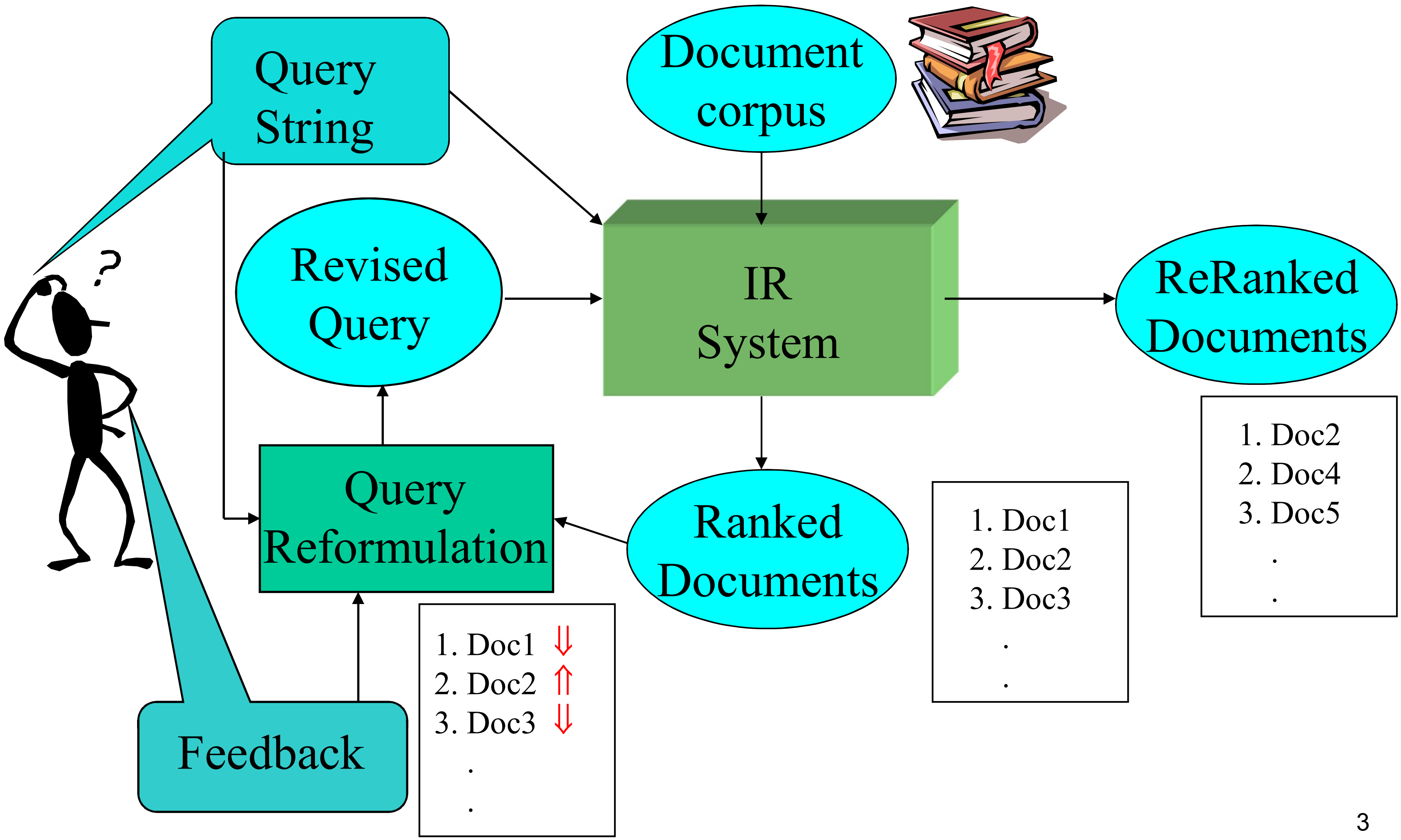
Query Operations

Relevance Feedback &
Query Expansion

Relevance Feedback

- After initial retrieval results are presented, allow the user to provide feedback on the relevance of one or more of the retrieved documents.
- Use this feedback information to reformulate the query.
- Produce new results based on reformulated query.
- Allows more interactive, multi-pass process.

Relevance Feedback Architecture



Query Reformulation

- Revise query to account for feedback:
  - Query Expansion: Add new terms to query from relevant documents.
  - Term Reweighting: Increase weight of terms in relevant documents and decrease weight of terms in irrelevant documents.
- Several algorithms for query reformulation.

Query Reformulation for VSR

- Change query vector using vector algebra.
- **Add** the vectors for the **relevant** documents to the query vector.
- **Subtract** the vectors for the **irrelevant** docs from the query vector.
- This both adds both positive and negatively weighted terms to the query as well as reweighting the initial terms.

Optimal Query

- Assume that the relevant set of documents C_r are known.
- Then the best query that ranks all and only the relevant queries at the top 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.

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.

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

α: Tunable weight for initial query.
β: Tunable weight for relevant documents.
γ: Tunable weight for irrelevant documents.

Ide Regular Method

- Since more feedback should perhaps increase the degree of reformulation, do not normalize for amount of feedback:

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

α: Tunable weight for initial query.
β: Tunable weight for relevant documents.
γ: Tunable weight for irrelevant documents.

Ide “Dec Hi” Method

- Bias towards rejecting **just** the highest ranked of the irrelevant documents:

$$\vec{q}_m = \alpha \vec{q} + \beta \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \gamma \max_{non-relevant}(\vec{d}_j)$$

α: Tunable weight for initial query.
β: Tunable weight for relevant documents.
γ: Tunable weight for irrelevant document.

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.

Relevance Feedback in Java VSR

- Includes “Ide Regular” method.
- Invoke with “-feedback” option, use “r” command to reformulate and redo query.
- See sample feedback trace.
- Since stored frequencies are not normalized (since normalization does not effect cosine similarity), must first divide all vectors by their maximum term frequency.

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.

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.

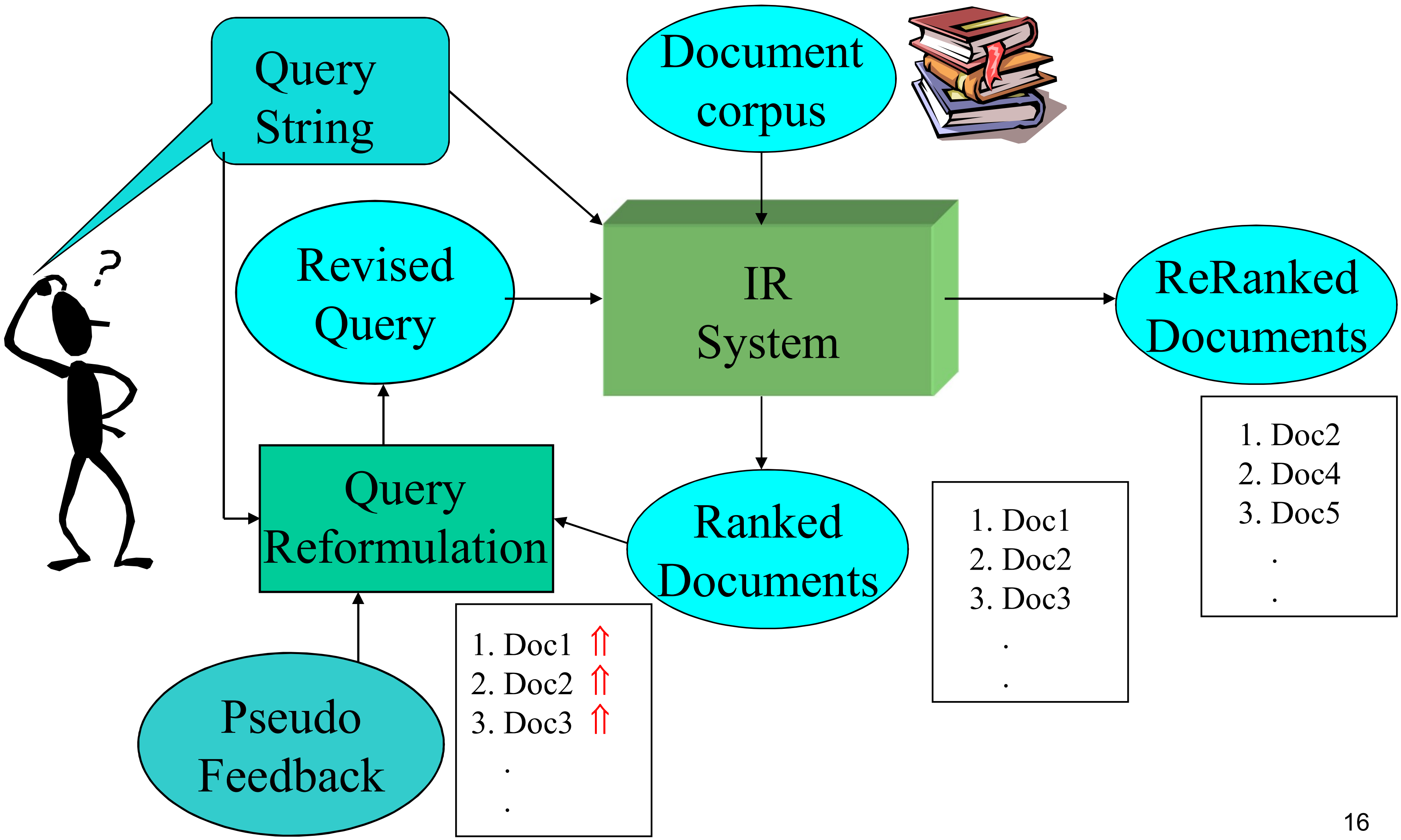
Why is Feedback Not Widely Used

- 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

- A thesaurus provides information on synonyms and semantically related words and phrases.
- Example:

```
physician
  syn: ||croaker, doc, doctor, MD,
medical, mediciner, medico, ||sawbones
  rel: medic, general practitioner,
surgeon,
```

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.
- Developed by famous cognitive psychologist George Miller and a team at Princeton University.
- About 144,000 English words.
- Nouns, adjectives, verbs, and adverbs grouped into about 109,000 synonym sets called *synsets*.

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 to whole)
- Meronym: computer → cpu (whole to part)
- Hyponym: plant → tree (specialization)
- Hypernym: apple → fruit (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.

Statistical Thesaurus

- Existing human-developed thesauri are not easily available in all languages.
- Human thesuari are limited in the type and range of synonymy and semantic relations they represent.
- Semantically related terms can be discovered from statistical analysis of corpora.

Automatic Global Analysis

- Determine term similarity through a pre-computed statistical analysis of the complete corpus.
- Compute association matrices which quantify term correlations in terms of how frequently they co-occur.
- Expand queries with statistically most similar terms.

Association Matrix

$$
\begin{array}{c|cccc|}
 & w_1 & w_2 & w_3 \ldots\ldots\ldots\ldots & w_n \\
\hline
w_1 & c_{11} & c_{12} & c_{13} \ldots\ldots\ldots\ldots & c_{1n} \\
w_2 & c_{21} & & & \\
w_3 & c_{31} & & & \\
. & . & & & \\
. & . & & & \\
w_n & c_{n1} & & & \\
\hline
\end{array}
$$

c_{ij}: Correlation factor between term *i* and term *j*

$$c_{ij} = \sum_{d_k \in D} f_{ik} \times f_{jk}$$

f_{ik} : Frequency of term *i* in document *k*

Normalized Association Matrix

- Frequency based correlation factor favors more frequent terms.
- Normalize association scores:

$$s_{ij} = \frac{c_{ij}}{c_{ii} + c_{jj} - c_{ij}}$$

- Normalized score is 1 if two terms have the same frequency in all documents.

Metric Correlation Matrix

- Association correlation does not account for the proximity of terms in documents, just co-occurrence frequencies within documents.
- Metric correlations account for term proximity.

$$c_{ij} = \sum_{k_u \in V_i} \sum_{k_v \in V_j} \frac{1}{r(k_u, k_v)}$$

V_i: Set of all occurrences of term i in any document.

$r(k_u,k_v)$: Distance in words between word occurrences k_u and k_v (∞ if k_u and k_v are occurrences in different documents).

Normalized Metric Correlation Matrix

- Normalize scores to account for term frequencies:

$$s_{ij} = \frac{c_{ij}}{|V_i| \times |V_j|}$$

Query Expansion with Correlation Matrix

- For each term i in query, expand query with the n terms, j, with the highest value of c_{ij} (s_{ij}).
- This adds semantically related terms in the "neighborhood" of the query terms.

Problems with Global Analysis

- Term ambiguity may introduce irrelevant statistically correlated terms.
  – "Apple computer" → "Apple red fruit computer"
- Since terms are highly correlated anyway, expansion may not retrieve many additional documents.

Automatic Local Analysis

- At query time, dynamically determine similar terms based on analysis of top-ranked retrieved documents.
- Base correlation analysis on only the "local" set of retrieved documents for a specific query.
- Avoids ambiguity by determining similar (correlated) terms only within relevant documents.
  - "Apple computer" →
    "Apple computer Macbook laptop"

Global vs. Local Analysis

- Global analysis requires intensive term correlation computation only once at system development time.
- Local analysis requires intensive term correlation computation for every query at run time (although number of terms and documents is less than in global analysis).
- But local analysis gives better results.

Global Analysis Refinements

- Only expand query with terms that are similar to *all* terms in the query.

$$sim(k_i, Q) = \sum_{k_j \in Q} c_{ij}$$

  - "fruit" not added to "Apple computer" since it is far from "computer."
  - "fruit" added to "apple pie" since "fruit" close to both "apple" and "pie."
- Use more sophisticated term weights (instead of just frequency) when computing term correlations.

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.