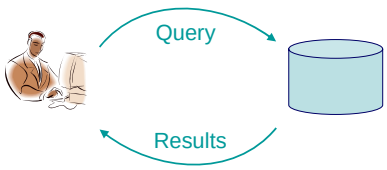
Relevance Feedback, Query Expansion, and Inputs to Ranking Beyond Similarity

Lecture 10
CS 410/510
Information Retrieval on the Internet

Outline

- Query reformulation
- Sources of relevance for feedback
- Using relevance feedback
  - Vector model
  - Probabilistic model
- Clustering approaches
  - Local analysis
  - Global analysis
- Examples of relevance feedback and query expansion
- Inputs to document ranking other than similarity

The basic problem

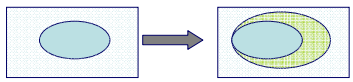


Queries are not always successful

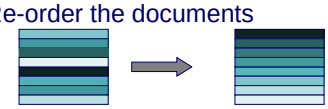
New queries do not learn from previous failure

Possible solutions

- Retrieve new documents



  – Add new terms to the query (*query expansion*)
- Re-order the documents



  – Re-weight terms

How?

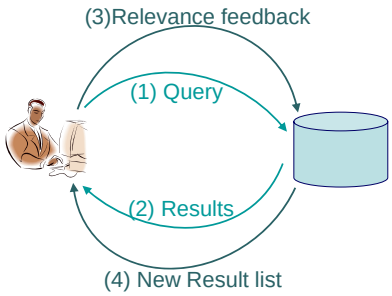
- Where do we get information for adding query terms or re-weighting documents?
- Use relevant documents to help find more relevant documents
- Assumption:
  – Relevant documents are more similar to each other than they are to non-relevant documents

How?

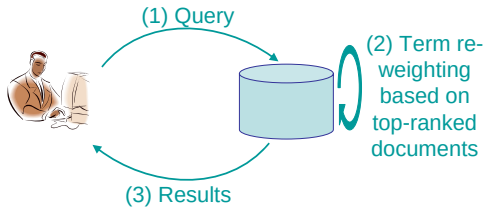
How do we get relevant documents?

1. Ask the user
   – Systems returns documents for initial query
   – User marks which ones are relevant
   – System uses as input to new query
   – Assumption: user will recognize documents that are useful, or close to what he wants
2. Assume top-ranked documents are relevant; automatically reformulate query

Relevance feedback: user



Blind feedback



Using relevance feedback

- Vector model: use terms and term weights in relevant documents to reformulate the query
  - Add new terms (query expansion)
  - Re-weight existing terms
- Basic approach:

  $Vec_{newQ} = Vec_{oldQ} + Vec_{posExamples} - Vec_{negExamples}$

Relevance feedback: Vector model

Three classic query reformulations: (α, β, γ originally set to 1)
D_r is the set of relevant docs; D_n is the set of non-relevant docs

1. Rocchio $\vec{q_m} = \alpha\vec{q} + \frac{\beta}{|D_r|}\sum_{\forall \vec{d_j} \in D_r} \vec{d_j} - \frac{\gamma}{|D_n|}\sum_{\forall d_j \in D_n} \vec{d_j}$

   Divide positive & negative vectors by size of example sets
2. Ide $\vec{q_m} = \alpha\vec{q} + \beta\sum_{\forall \vec{d_j} \in D_r} \vec{d_j} - \gamma\sum_{\forall d_j \in D_n} \vec{d_j}$
3. Ide Dec-Hi $\vec{q_m} = \alpha\vec{q} + \beta\sum_{\forall \vec{d_j} \in D_r} \vec{d_j} - \gamma \max_{non-relevant}(\vec{d_j})$

   Use only top-ranked non-relevant document

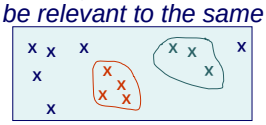
$Vec_{newQ} = Vec_{oldQ} + Vec_{posExamples} - Vec_{negExamples}$

Using relevance feedback

- Probabilistic model: use relevant documents to re-calculate probabilities of relevance
  - Re-weight terms
  - No query expansion

Clustering approaches

- Cluster hypothesis: "*closely associated documents tend to be relevant to the same requests*"[1]



- Instead of using weights of individual terms in relevant documents, use relevant documents to describe of cluster of desirable documents

[1] *C.J vanRijsbergen. Information Retrieval, Chapter 3. 1979.*

Clustering for query reformulation

- Local analysis
  - Operate only on documents returned for the current query – the local set
  - Use term co-occurrences to select terms for query expansion
- Global analysis
  - Use information from all the documents in the collection

Local Clustering

- Use term co-occurrence data to calculate correlation between terms *u*, *v*: $C_{u,v}$
- Add the *n* query terms with the highest $C_{u,v}$ where *u* is a term in the original query
- Reflects term clustering

Local Clustering

- Association clusters:
  - $C_{u,v}$ reflects frequency of co-occurrence in the same document
- Metric clusters:
  - $C_{u,v}$ reflects proximity of co-occurrences (number of words between occurrences of *u* and *v*) in documents
- Scalar clusters:
  - Calculate vectors of correlation values for all terms in local documents
  - $C_{u,v}$ reflects similarity of vectors (e.g. cosine of angle between vectors for *u* and *v*)

Local Context Analysis

Use noun groups (one or more adjacent nouns in text) not just keywords: "concepts"

1. Retrieve top *n*-ranked passages with original query
2. Calculate similarity between query and each concept (noun group)
   - Reflects co-occurrence of query terms and concepts in passages
3. Add top *m*-ranked concepts to query, weighted by similarity ranking

Global analysis

- Similarity thesaurus
- Statistical thesaurus

Creating a similarity thesaurus

- Index a term based on documents it appears in
  - each term associated with a vector $\vec{k}_i = (w_{i,1}, w_{i,2}, ..., w_{i,N})$
  - where $w_{i,j}$ is the weight of document$_j$ associated with term$_i$
  - $w_{i,j}$ is calculated similarly to *tf-idf* except it uses inverse term frequency (not inverse document freq.)
  - length of vector is number of documents in collection
- Calculate a matrix of term correlations

$$c_{u,v} = \vec{k}_u \bullet \vec{k}_v = \sum_{\forall d_j} w_{u,j} \times w_{v,j}$$

Using a similarity thesaurus

- Compare candidate terms to the query "concept" or centroid
  - Calculate virtual query vector that combines vectors for each query term, weighted by $w_{i,q}$
  - Previous uses of global analysis added terms based on correlation with **individual** query terms
- Calculate similarity of each term in collection to the query concept
- Add top *r* terms to original query
  - Weighted by similarity to original query

Creating a statistical thesaurus

- Based on clustering documents
  - Algorithm produces hierarchical clusters
  - Uses cosine formula from vector space model
- Classes derived from clusters and 3 parameters
  - Similarity threshold (want tight clusters)
  - Cluster size (want small clusters)
  - Minimum inverse document frequency
    - Only want to use low frequency terms
- Weight the thesaurus classes
  - average term weight/number of terms in class

Using a statistical thesaurus

- Terms within a document class are considered related
- Documents and queries are indexed (augmented) using the thesaurus classes

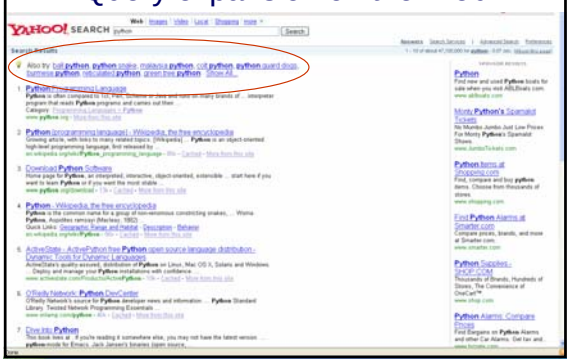
Relevance feedback on the Web

Query expansion on the Web

GoogleSuggest FAQs:

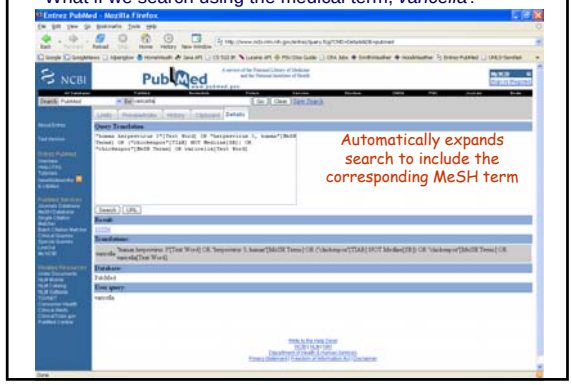
"That's pretty cool. How does it do that?
Our algorithms use a wide range of information to predict the queries users are most likely to want to see. For example, Google Suggest uses data about the overall popularity of various searches to help rank the refinements it offers."

Query expansion on the Web



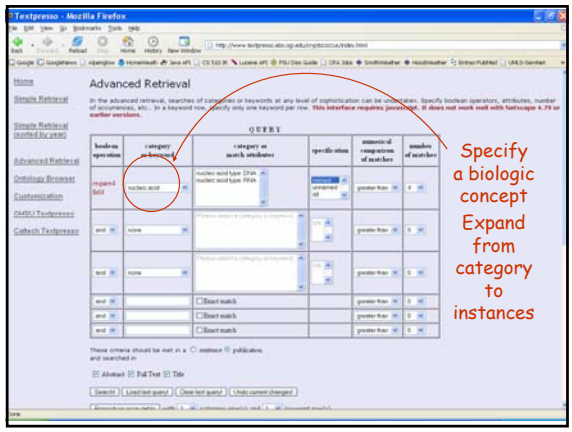
Relevance feedback

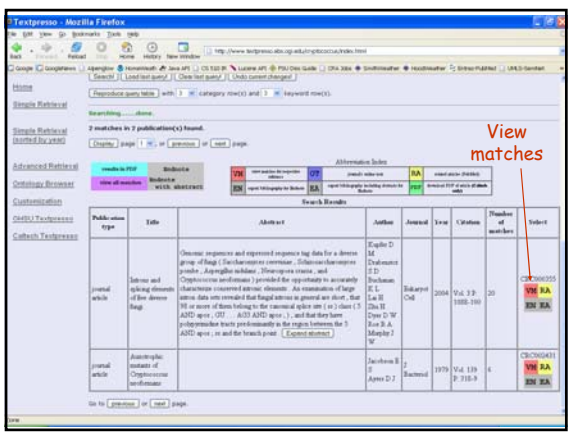
What if we search using the medical term, *varicella*?

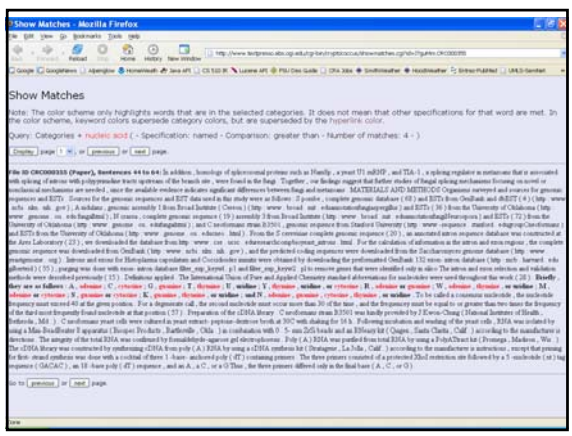


Textpresso

- Text mining application for scientific literature
- Can search terms or categories of terms
  - Biological concepts (gene, cell, nucleic acid)
  - Relationships (association, regulation)
  - Biological processes
- Can search for "gene" or do query expansion to search for any gene name







Document ranking

- Using considerations besides similarity
  - Filters
  - Ranking adjustments

Possible filters

- Language (English, Spanish, etc)
- Source
- Other metadata
- Novelty
  - Suppress documents already seen, duplicates, near-duplicates from same site

Possible ranking inputs

- Inputs other than similarity
  - Importance (page rank)
  - Popularity
  - Reading level
  - Currency
  - Quality
    - What is quality? What are the criteria? Who applies the criteria?

- Next: Classification and clustering