Information Retrieval and Web Search

Relevance Feedback.
Query Expansion

Instructor: Rada Mihalcea
Intelligent Information Retrieval

1. Relevance feedback
   - Direct feedback
   - Pseudo feedback

2. Query expansion
   - With a “natural” thesaurus
   - With an “artificial” thesaurus
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.

• Similar with what IR basic model?
Relevance Feedback Architecture

- Query String
- Document corpus
- IR System
- Ranked Documents
- ReRanked Documents
- Revised Query
- Query Reformulation
- Feedback

1. Doc1
2. Doc2
3. Doc3

1. Doc1
2. Doc2
3. Doc3

1. Doc2
2. Doc4
3. Doc5
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 in Vector-Space Model

- 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:

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

Where $N$ is the total number of documents in the collection.
Standard Rocchio Method

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

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

- $\alpha$: Tunable weight for initial query.
- $\beta$: Tunable weight for relevant documents.
- $\gamma$: Tunable weight for irrelevant documents.

Notice terms are normalized with the “amount” of feedback.
Ide Regular Method

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

\[
\tilde{q}_m = \alpha \tilde{q} + \beta \sum_{\forall d_j \in D_r} \tilde{d}_j - \gamma \sum_{\forall d_j \in D_n} \tilde{d}_j
\]

\(\alpha\): Tunable weight for initial query.
\(\beta\): Tunable weight for relevant documents.
\(\gamma\): Tunable weight for irrelevant documents.
Ide “Dec Hi” Method

• Bias towards rejecting just the highest ranked of the irrelevant documents:

\[
\tilde{q}_m = \alpha \tilde{q} + \beta \sum_{\forall d_j \in D_{r}} \tilde{d}_j - \gamma \max_{\text{non\_relevant}} (\tilde{d}_j)
\]

\(\alpha\): Tunable weight for initial query.
\(\beta\): Tunable weight for relevant documents.
\(\gamma\): 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.

\[ \alpha = \beta = \gamma = 1 \]
Example

• Query: cheap CDs cheap DVDs extremely cheap CDs
• D1 relevant: CDs cheap software cheap CDs
• D2 irrelevant: cheap thrills DVDs

• Assume:
  – Standard Rocchio with $\alpha = 1$, $\beta = 0.75$, $\gamma = 0.25$
  – Weighting only based on TF (no normalization, no IDF)

• What is the query vector after relevance feedback?
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 unrated 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 are reluctant to provide explicit feedback
- 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
- Found to improve performance on TREC ad-hoc retrieval tasks
- Works even better if top documents must also satisfy additional boolean constraints in order to be used in feedback. (not only constraints imposed by the vector-space model)
Relevance Feedback Architecture

Query String

Revised Query

IR System

Ranked Documents

ReRanked Documents

1. Doc1
2. Doc2
3. Doc3

1. Doc1
2. Doc2
3. Doc3

Pseudo Feedback
Relevance Feedback on the Web

- Excite initially had true relevance feedback, but abandoned it due to lack of use

- Some search engines offer a similar/related pages feature (simplest form of relevance feedback)
  - Google

- Implicit relevance feedback is frequently used
  - Click-through data
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,
Query Expansion with a Thesaurus

- 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
  - Why?
WordNet

- A database of semantic relationships between English words
- Developed by Prof. George Miller and a team at Princeton University
- About 150,000 English words
- Nouns, adjectives, verbs, and adverbs grouped into about 110,000 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-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.
Statistical Thesaurus

- Existing human-developed thesauri are not easily available in all languages
- Human thesauri 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

<table>
<thead>
<tr>
<th>W₁</th>
<th>W₂</th>
<th>W₃</th>
<th>...</th>
<th>Wₙ</th>
</tr>
</thead>
<tbody>
<tr>
<td>W₁</td>
<td>C₁₁</td>
<td>C₁₂</td>
<td>C₁₃</td>
<td>...</td>
</tr>
<tr>
<td>W₂</td>
<td>C₂₁</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W₃</td>
<td>C₃₁</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wₙ</td>
<td>Cₙ₁</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ c_{ij} : \text{Correlation factor between term } i \text{ and term } j \]

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

\[ f_{ik} : \text{Frequency of term } i \text{ in document } k \]

Or, we could look for the number of times two words co-occur in a document.

Does this matrix remind you of anything we have seen so far?
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
Query Expansion with Correlation Matrix

- For each term $i$ in query, expand query with the terms with the highest value of $s_{ij}$
- 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” $\rightarrow$ “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 Powerbook 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

\[ \text{sim}(k_i, Q) = \sum_{k_j \in Q} c_{ij} \]

  – “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
Conclusion

• Relevance feedback (manual or automatic) and query expansion are techniques for intelligent information retrieval

• 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

• Attempt to improve a “basic” IR system by learning new terms

• Always improve recall, sometimes improve precision