Relevance Feedback and Query Expansion

Reference: Introduction to Information Retrieval
by C. Manning, P. Raghavan, H. Schutze
Relevance Feedback

• It may be difficult to formulate a good query when you don’t know the collection well.
• 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 String
- Document corpus
- Revised Query
- IR System
- Ranked Documents
- ReRanked Documents
- Feedback

1. Doc1
2. Doc2
3. Doc3

1. Doc2
2. Doc4
3. Doc5

Query Reformulation

1. Doc1
2. Doc2
3. Doc3
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

- If $C_r$ is the relevant set of documents and $C_n$ is the set of irrelevant documents.

$$\hat{q}_{opt} = \arg\max_{\hat{q}} [\text{sim}(\hat{q}, C_r) - \text{sim}(\hat{q}, C_n)]$$

- Under cosine similarity, the optimal query is:

$$\hat{q}_{opt} = \frac{1}{|C_r|} \sum_{\hat{d}_j \in C_r} \hat{d}_j - \frac{1}{|C_n|} \sum_{\hat{d}_j \in C_n} \hat{d}_j$$

- The optimal query is the vector difference between the centroids of the relevant and irrelevant documents.
Optimal Query

- X: non-relevant documents
- O: relevant documents
Rocchio’s Algorithm

- Initial query: Red triangle
- Revised query: Green triangle
- Known non-relevant documents: X
- Known relevant documents: O
Rocchio’s Algorithm

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

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

\( \alpha \): a weight for initial query.
\( \beta \): a weight for relevant documents.
\( \gamma \): a weight for irrelevant documents.
Ide Regular Method

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

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

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

- **Query String**
- **Document corpus**
- **IR System**
- **ReRanked Documents**

### Query Reformulation
- Pseudo Feedback
  - 1. Doc1
  - 2. Doc2
  - 3. Doc3

### Revised Query
- 1. Doc2
- 2. Doc4
- 3. Doc5

### Ranked Documents
- 1. Doc1
- 2. Doc2
- 3. Doc3
- .
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.
Query Expansion

• In relevance feedback, users give additional input (relevant/non-relevant) on documents, which is used to reweight terms in the documents

• In query expansion, users give additional input (good/bad search term) on words or phrases
How do we augment the user query?

- **Manual thesaurus**
  - E.g. MedLine: physician, syn: doc, doctor, MD, medico
  - Can be query rather than just synonyms

- **Global Analysis**: (static; of all documents in collection)
  - Automatically derived thesaurus
    - (co-occurrence statistics)
  - Refinements based on query log mining
    - Common on the web

- **Local Analysis**: (dynamic)
  - Analysis of documents in result set
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,
Example of manual thesaurus
Thesaurus-based query expansion

• For each term, $t$, in a query, expand the query with synonyms and related words of $t$ from the thesaurus
  – feline $\rightarrow$ feline cat

• May weight added terms less than original query terms.

• Generally increases recall

• Widely used in many science/engineering fields

• May significantly decrease precision, particularly with ambiguous terms.
  – “interest rate” $\rightarrow$ “interest rate fascinate evaluate”

• There is a high cost of manually producing a thesaurus
  – And for updating it for scientific changes
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 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></th>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_3$</th>
<th>$\ldots$</th>
<th>$w_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_1$</td>
<td>$c_{11}$</td>
<td>$c_{12}$</td>
<td>$c_{13}$</td>
<td>$\ldots$</td>
<td>$c_{1n}$</td>
</tr>
<tr>
<td>$w_2$</td>
<td>$c_{21}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w_3$</td>
<td>$c_{31}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\ddots$</td>
<td>$\ddots$</td>
<td>$\ddots$</td>
<td>$\ddots$</td>
<td>$\ddots$</td>
</tr>
<tr>
<td>$w_n$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$c_{n1}$</td>
</tr>
</tbody>
</table>

$c_{ij}$: 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 \(k_v\)
  \((\infty\) 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”
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.
Query assist

- Generally done by query log mining
- Recommend frequent recent queries that contain partial string typed by user
- A ranking problem! View each prior query as a doc – Rank-order those matching partial string …