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:

\[
\tilde{q}_{opt} = \frac{1}{|C_r|} \sum_{\forall d \in C_r} \tilde{d}_j - \frac{1}{N - |C_r|} \sum_{\forall d \notin C_r} \tilde{d}_j
\]

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 $q$.

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

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

Ide Regular Method

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

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

$\alpha$: 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:

$$
\tilde{q}_m = \alpha \tilde{q} + \beta \sum_{d_j \in D_n} \tilde{d}_j - \gamma \max_{\text{non-relevant}}(\tilde{d}_j)
$$

$\alpha$: 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.

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 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 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>w_1</th>
<th>w_2</th>
<th>w_3</th>
<th>\ldots</th>
<th>w_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>c_{i1}</td>
<td>c_{i2}</td>
<td>c_{i3}</td>
<td>\ldots</td>
<td>c_{in}</td>
</tr>
<tr>
<td>c_{21}</td>
<td>c_{22}</td>
<td>\ldots</td>
<td>\ldots</td>
<td>c_{2n}</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\ddots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>c_{n1}</td>
<td>c_{n2}</td>
<td>\ldots</td>
<td>\ldots</td>
<td>c_{nn}</td>
</tr>
</tbody>
</table>

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

c_{ij} = \sum_{d \in D} f_i^d \times f_j^d

f_k^i \text{ Frequency of term } i \text{ 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.

\[
c_{ij} = \sum_{k \in V} \sum_{k' \in V} \frac{1}{r(k_i, k_j)}
\]

V: Set of all occurrences of term i in any document.
r(k_i, k_j): Distance in words between word occurrences k_i and k_j.
(= if k_i and k_j are occurrences in different documents.)
Normalized Metric Correlation Matrix

- Normalize scores to account for term frequencies:
\[ s_j = \frac{c_{ij}}{|V|} \]

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.

\[ \text{sim}(k, Q) = \sum_{i \in \mathcal{Q}} \text{c}_i \]

  – “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.