Chapter 05: Query Reformulation
1. Basic concept
2. Query refinement through relevance feedback
3. Query expansion through term clustering
   1. Query-specific clustering
   2. Global clustering (statistical thesauri)
      1. Clustering techniques
      2. Document clustering
      3. Cluster Hierarchies
   3. Semantic thesaurus
Basic concept

• Often, initial queries are not optimal. Possibly because
  • User's lack of knowledge of the domain.
  • User's vocabulary does not match authors' vocabulary.
• Normally, users follow-up initial queries with reformulations that take advantage of experience gained in the initial attempt.
• *Automatic query reformulation*: Rewrite the initial query automatically (or with some assistance from the user).
• Process could be a *single* (final) reformulation, or an *iterative refinement process* that (hopefully) converges to the optimal query.
• We examine two approaches:
  1. Query *refinement* through user relevance feedback.
  2. Query *expansion* through word clusters.
Query refinement by Relevance Feedback

• User examines the set of documents retrieved by the query and marks those that are relevant to his search.
• In practice, only the top (10-20) documents retrieved need to be examined.
• The system then reformulates the query to emphasize the concepts (e.g., terms) expressed in the documents judged to be relevant, and de-emphasize the concepts in the documents judged to be non-relevant.
• Process may be repeated iteratively.
• Advantage: User is only required to mark documents, and does not get involved in the technicalities of reformulation.
• Next, we examine this technique in the vector model.
Refinement by relevance feedback (cont.)

• In the vector model, a query is a vector of term weights; hence reformulation involves *reassigning term weights*.
• If a document is known to be relevant, the query can be improved by *increasing* its similarity to that document.
• If a document is known to be non-relevant, the query can be improved by *decreasing* its similarity to that document.
• Problems:
  • What if the query has to increase its similarity to two very non-similar documents (each “pulls” the query in an entirely different direction)?
  • What if the query has to be decrease its similarity to two very non-similar documents (each “pushes” the query in an entirely different direction)?
• Critical assumptions that must be made:
  • Relevant documents resemble each other (are clustered).
  • Non-relevant documents resemble each other (are clustered).
  • Non-relevant documents differ from the relevant documents.
Refinement by relevance feedback (cont.)

• For a query \( q \), denote
  • \( D_R \): The set of relevant documents in the answer (as identified by the user)
  • \( D_N \): The set of non-relevant documents in the answer.
  • \( C_R \): The set of relevant documents in the collection (the ideal answer).

• Assume (unrealistic!) that \( C_R \) is known in advance.
• It can then be shown that the best query vector for distinguishing the relevant documents from the non-relevant documents is:

\[
\frac{1}{|C_R|} \sum_{d \in C_R} d_j - \frac{1}{|n - C_R|} \sum_{d \notin C_R} d_j
\]

• The left expression is the centroid of the relevant documents, the right expression is the centroid of the non-relevant documents.
• Note: This expression is vector arithmetic! \( d_j \) are vectors, whereas \( |C_R| \) and \( |n - C_R| \) are scalars.
Refinement by relevance feedback (cont.)

- Since we don’t know \( CR \), we shall substitute it by \( DR \) in each of the two expressions, and then use them to modify the initial query \( q \):

\[
q_{\text{new}} = \alpha \cdot q + \frac{\beta}{|DR|} \sum_{d \in DR} d_j - \frac{\gamma}{|DN|} \sum_{d \in DN} d_j
\]

- \( \alpha, \beta, \) and \( \gamma \) are tuning constants; for example, 1.0, 0.5, 0.25.
- **Note:** This expression is *vector arithmetic*! \( q \) and \( d_j \) are vectors, whereas \( |DR|, |DN|, \alpha, \beta, \) and \( \gamma \) are scalars.
Refinement by relevance feedback (cont.)

- \( \frac{\beta}{|D_R|} \sum_{d \in D_R} d_j \) Positive feedback factor. Uses the user's judgments on relevant documents to increase the values of terms. Moves the query to retrieve documents similar to relevant documents retrieved (in the direction of more relevant documents).

- \( \frac{y}{|D_N|} \sum_{d \in D_N} d_j \) Negative feedback factor. Uses the user's judgments on non-relevant documents to decrease the values of terms. Moves the query away from non-relevant documents.

- Positive feedback often weighted significantly more than negative feedback; often, only positive feedback is used.
Refinement by relevance feedback (cont.)

• **Example:**
  • Assume query \( q = (3,0,0,2,0) \) retrieved three documents: \( d1, d2, d3 \).
  • Assume \( d1 \) and \( d2 \) are judged relevant and \( d3 \) is judged non-relevant.
  • Assume the tuning constants used are 1.0, 0.5, 0.25.

\[
\begin{array}{c|ccccc}
  & k1 & k2 & k3 & k4 & k5 \\
  \hline
  q & 3 & 0 & 0 & 2 & 0 \\
  d1 & 2 & 4 & 0 & 0 & 2 \\
  d2 & 1 & 3 & 0 & 0 & 0 \\
  d3 & 0 & 0 & 4 & 3 & 3 \\
  qnew & 3.75 & 1.75 & 0 & 1.25 & 0 \\
\end{array}
\]

The revised query is:

\[
q_{new} = (3, 0, 0, 2, 0) + 0.5 \times (\frac{2+1}{2}, \frac{4+3}{2}, \frac{0+0}{2}, \frac{0+0}{2}, \frac{2+0}{2}) - 0.25 \times (0, 0, 4, 3, 3)
\]

\[
= (3.75, 1.75, -1, 1.25, 0)
\]

\[
= (3.75, 1.75, 0, 1.25, 0)
\]
Refinement by relevance feedback (cont.)

- Using a simplified similarity formula (the nominator only of the cosine):

\[
similarity(d_j, q) = \sum_{i=1}^{t} W_{i,j} \times W_{i,q}
\]

we can compare the similarity of \(q\) and \(q_{new}\) to the three documents:

<table>
<thead>
<tr>
<th></th>
<th>(d_1)</th>
<th>(d_2)</th>
<th>(d_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(q)</td>
<td>6</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>(q_{new})</td>
<td>14.5</td>
<td>9</td>
<td>3.75</td>
</tr>
</tbody>
</table>

- Compared to the original query, the new query is indeed more similar to \(d_1\), \(d_2\). (which were judged relevant), and less similar to \(d_3\) (which was judged non-relevant).
- Notice how the new query added \(k_2\), which was not in the original query.
  - For example, a user may be searching for *software* for a PC, may (inadvertently) mark as relevant documents about software which is actually for a Mac, and the revised query may introduce the term *Mac*. 
Problem: Relevance feedback may not operate satisfactorily, if the identified relevant documents do not form a tight cluster.

Possible solution: Cluster the identified relevant documents, then split the original query into several, by constructing a new query for each cluster.

Problem: Some of the query terms might not be found in any of the retrieved documents. This will lead to reduction of their relative weight in the modified query (or even elimination). Undesirable, because these terms might still be found in future iterations.

Possible solutions: Ensure that the original terms are kept; or present all modified queries to the user for review.

Problem: New query terms might be introduced that conflict with the intention of the user (as in the previous example).

Possible solutions: Present all modified queries to the user for review.
Refinement by relevance feedback (cont.)

• **Conclusion**: Experimentation showed that user relevance feedback in the vector model gives good results.

• However:
  • Users are sometimes reluctant to provide explicit feedback
  • Results in long queries that needs more computation to retrieve, which is a lot of time for search engines.
  • Makes it harder to understand why a particular document was retrieved.

• “**Fully automatic” relevance feedback**: The rank values for the documents in the first answer are used as relevance feedback to automatically generate the second query (no human judgment).
  • The highest ranking documents are assumed to be relevant (positive feedback only).
Query expansion by term clustering

• Expand the query with terms that are similar to the terms mentioned in the query.
  • For each term, create a cluster of similar terms.
• The relevance feedback method shifts the query:
  • Some terms are emphasized, others de-emphasized.
  • Consequently, new documents are discovered and some old documents are abandoned.
• The word-clustering method expands the query:
  • Terms are added.
  • New documents are discovered (but none discarded).
  • Recall increases (and precision decreases).
  • Compensates for the different vocabularies of author and user.
  • Requires no user involvement.
Local vs. Global Automatic Analysis

- Local – Documents retrieved are examined to automatically determine query expansion. No relevance feedback needed.
- Global – Thesaurus used to help select terms for expansion.
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”
Automatic Local Analysis

- Expand query with terms found in local clusters.
- \( D_l \) – set of documents retrieved for query \( q \).
- \( V_l \) – Set of words used in \( D_l \).
- \( S_l \) – Set of distinct stems in \( V_l \).
- \( f_{s_{i,j}} \) – Frequency of stem \( s_i \) in document \( d_j \) found in \( D_l \).
- Construct stem-stem association matrix.
### Association Matrix

<table>
<thead>
<tr>
<th>w_1</th>
<th>w_2</th>
<th>w_3</th>
<th>\cdots</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>\cdots</td>
</tr>
<tr>
<td>w_2</td>
<td>c_{21}</td>
<td>\cdot</td>
<td>\cdot</td>
<td>\cdots</td>
</tr>
<tr>
<td>w_3</td>
<td>c_{31}</td>
<td>\cdot</td>
<td>\cdot</td>
<td>\cdots</td>
</tr>
<tr>
<td>\cdot</td>
<td>\cdot</td>
<td>\cdot</td>
<td>\cdot</td>
<td>\cdots</td>
</tr>
<tr>
<td>w_n</td>
<td>c_{n1}</td>
<td>\cdot</td>
<td>\cdot</td>
<td>\cdots</td>
</tr>
</tbody>
</table>

- \( c_{ij} \): Correlation factor between stems \( s_i \) and stem \( s_j \)

\[
c_{ij} = \sum_{d_k \in D_l} f_{sik} \times f_{sjk}
\]

- \( f_{ik} \): 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 stems have the same frequency in all documents.
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 Local 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 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.
Automatic Global Analysis

- There are two modern variants based on a thesaurus-like structure built using all documents in collection
  - Query Expansion based on a Similarity Thesaurus
  - Query Expansion based on a Statistical 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” $\rightarrow$ “interest rate fascinate evaluate”
Similarity Thesaurus

- The similarity thesaurus is based on term to term relationships rather than on a matrix of co-occurrence.
- This relationship are not derived directly from co-occurrence of terms inside documents.
- They are obtained by considering that the terms are concepts in a concept space.
- In this concept space, each term is indexed by the documents in which it appears.
- Terms assume the original role of documents while documents are interpreted as indexing elements.
Similarity Thesaurus

- The following definitions establish the proper framework
  - $t$: number of terms in the collection
  - $N$: number of documents in the collection
  - $f_{i,j}$: frequency of occurrence of the term $k_i$ in the document $d_j$
  - $t_j$: vocabulary of document $d_j$
  - $itf_j$: inverse term frequency for document $d_j$
Similarity Thesaurus

- Inverse term frequency for document $d_j$

$$\text{itf}_j = \log \frac{t}{t_j}$$

- To $k_i$ is associated a vector

$$k_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,N})$$
where \( w_{i,j} \) is a weight associated to index-document pair\([k_i,d_j]\). These weights are computed as follows

\[
w_{i,j} = \sqrt{\sum_{l=1}^{N} (0.5 + 0.5 \times \frac{f_{i,l}}{\max_l(f_{i,l})})^2 \times itf_j^2} \]
Similarity Thesaurus

- The relationship between two terms $k_u$ and $k_v$ is computed as a correlation factor $c_{u,v}$ given by

\[ c_{u,v} = \vec{k}_u \cdot \vec{k}_v = \sum_{\forall d_j} w_{u,j} \times w_{v,j} \]

- The global similarity thesaurus is built through the computation of correlation factor $c_{u,v}$ for each pair of indexing terms $[k_u, k_v]$ in the collection.
Similarity Thesaurus

- This computation is expensive
- Global similarity thesaurus has to be computed only once and can be updated incrementally
Query expansion is done in three steps as follows:

1. Represent the query in the concept space used for representation of the index terms.
2. Based on the global similarity thesaurus, compute a similarity $\text{sim}(q,kv)$ between each term $kv$ correlated to the query terms and the whole query $q$.
3. Expand the query with the top $r$ ranked terms according to $\text{sim}(q,kv)$.
Query Expansion - step one

- To the query $q$ is associated a vector $\vec{q}$ in the term-concept space given by

\[
\vec{q} = \sum_{k_i \in q} w_{i,q} \vec{k}_i
\]

where $w_{i,q}$ is a weight associated to the index-query pair $[k_i,q]$
Query Expansion - step two

- Compute a similarity $\text{sim}(q, k_v)$ between each term $k_v$ and the user query $q$

\[
\text{sim}(q, k_v) = \overrightarrow{q} \cdot \overrightarrow{k_v} = \sum_{k_u \in q} w_{u,q} \times c_{u,v}
\]

- where $c_{u,v}$ is the correlation factor
Query Expansion - step three

- Add the top r ranked terms according to \( \text{sim}(q,k_v) \) to the original query \( q \) to form the expanded query \( q' \)
- To each expansion term \( k_v \) in the query \( q' \) is assigned a weight \( w_{v,q'} \) given by

\[
 w_{v,q'} = \frac{\text{sim}(q,k_v)}{\sum_{k_u \in q} w_{u,q}}
\]

- The expanded query \( q' \) is then used to retrieve new documents to the user
Query Expansion Sample

- Doc1 = D, D, A, B, C, A, B, C
- Doc2 = E, C, E, A, A, D
- Doc3 = D, C, B, B, D, A, B, C, A
- Doc4 = A

- \( c(A,A) = 10.991 \)
- \( c(A,C) = 10.781 \)
- \( c(A,D) = 10.781 \)
- ...
- \( c(D,E) = 10.398 \)
- \( c(B,E) = 10.396 \)
- \( c(E,E) = 10.224 \)
Query Expansion Sample

- Query: $q = A E E$
  - $\text{sim}(q,A) = 24.298$
  - $\text{sim}(q,C) = 23.833$
  - $\text{sim}(q,D) = 23.833$
  - $\text{sim}(q,B) = 23.830$
  - $\text{sim}(q,E) = 23.435$

- New query: $q' = A C D E E$
  - $w(A,q') = 6.88$
  - $w(C,q') = 6.75$
  - $w(D,q') = 6.75$
  - $w(E,q') = 6.64$
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-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.
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.
Query Expansion Based on a Statistical Thesaurus

- Global thesaurus is composed of classes which group correlated terms in the context of the whole collection.
- Such correlated terms can then be used to expand the original user query.
- This terms must be low frequency terms.
- However, it is difficult to cluster low frequency terms.
- To circumvent this problem, we cluster documents into classes instead and use the low frequency terms in these documents to define our thesaurus classes.
- This algorithm must produce small and tight clusters.
Query Expansion based on a Statistical Thesaurus

- Use the thesaurus class to query expansion.
- Compute an average term weight $w_{tc}$ for each thesaurus class $C$

$$w_{tc} = \frac{\sum_{i=1}^{C} w_{i,C}}{|C|}$$
Query Expansion based on a Statistical Thesaurus

- $w_t^c$ can be used to compute a thesaurus class weight $w_c$ as

$$W_c = \frac{w_{tc}}{|C|} \times 0.5$$
Query Expansion Sample

Doc1 = D, D, A, B, C, A, B, C
Doc2 = E, C, E, A, A, D
Doc3 = D, C, B, B, D, A, B, C, A
Doc4 = A

q = A E E

TC = 0.90  NDC = 2.00  MIDF = 0.2

idf A = 0.0
idf B = 0.3
idf C = 0.12
idf D = 0.12
idf E = 0.60
Query Expansion based on a Statistical Thesaurus

Problems with this approach

- initialization of parameters $TC, NDC$ and $MIDF$
- $TC$ depends on the collection
- Inspection of the cluster hierarchy is almost always necessary for assisting with the setting of $TC$.
- A high value of $TC$ might yield classes with too few terms
Complete link algorithm

- This is document clustering algorithm with produces small and tight clusters
  - Place each document in a distinct cluster.
  - Compute the similarity between all pairs of clusters.
  - Determine the pair of clusters \([C_u, C_v]\) with the highest inter-cluster similarity.
  - Merge the clusters \(C_u\) and \(C_v\)
  - Verify a stop criterion. If this criterion is not met then go back to step 2.
  - Return a hierarchy of clusters.

- Similarity between two clusters is defined as the minimum of similarities between all pair of inter-cluster documents
Selecting the terms that compose each class

- Given the document cluster hierarchy for the whole collection, the terms that compose each class of the global thesaurus are selected as follows
  - Obtain from the user three parameters
    - TC: Threshold class
    - NDC: Number of documents in class
    - MIDF: Minimum inverse document frequency
Selecting the terms that compose each class

- Use the parameter TC as threshold value for determining the document clusters that will be used to generate thesaurus classes.
- This threshold has to be surpassed by \( \text{sim} (C_u, C_v) \) if the documents in the clusters \( C_u \) and \( C_v \) are to be selected as sources of terms for a thesaurus class.
Selecting the terms that compose each class

- Use the parameter NDC as a limit on the size of clusters (number of documents) to be considered.
- A low value of NDC might restrict the selection to the smaller cluster Cu+v
Selecting the terms that compose each class

- Consider the set of document in each document cluster pre-selected above.
- Only the lower frequency documents are used as sources of terms for the thesaurus classes
- The parameter MIDF defines the minimum value of inverse document frequency for any term which is selected to participate in a thesaurus class
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.
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.
Conclusion

- Thesaurus is an efficient method to expand queries.
- The computation is expensive but it is executed only once.
- Query expansion based on similarity thesaurus may use high term frequency to expand the query.
- Query expansion based on statistical thesaurus need well defined parameters.