Query Operations

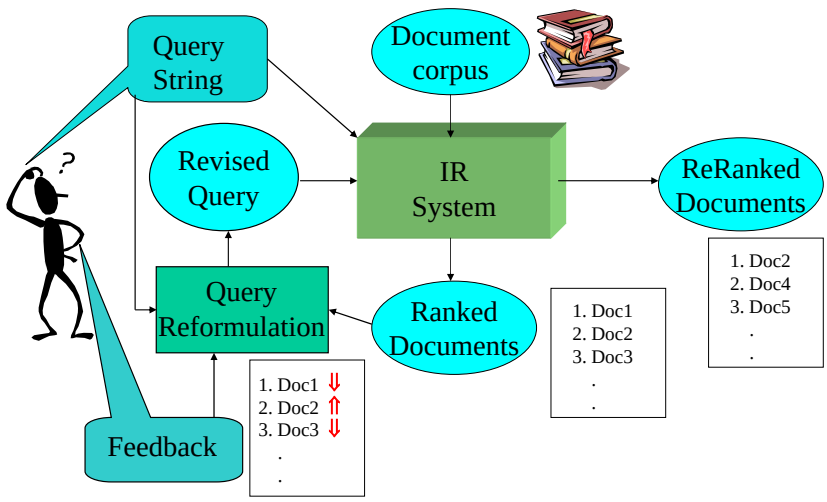
Query Operations

- The main goal: refine a query and improve the usefulness of the obtained answer
- Starting with a “naive” query, the IR system should be able to observe the user’s judgment on relevance and use it to improve retrieval effectiveness
- Note that this involves possible addition of new query terms as well as reweighting of terms in the original query
- Approaches for improving the initial query
  - Feedback information from the user (relevance feedback methods)
  - Information derived from the set of initially retrieved documents (local analysis)
  - Information derived from the document collection (global analysis)

Relevance Feedback

- The user issues a query, and marks some documents as relevant. The query engine then uses the terms indexing those documents (a vector model is assumed) to expand and reweight the original query terms
- The whole idea is move the query away from the set of non-relevant documents (or closer to the relevant ones)
- Basic assumptions:
  - Relevant documents resemble each other
  - The term-weight vectors of non-relevant documents are dissimilar from the ones for the relevant documents
- Advantages:
  - All users need to do is to judge relevance
  - Allows incremental searching (“browsing” like)
  - Relevance definition is based on interaction

Relevance Feedback Architecture



Term Reweighting in the Vector Model

- The vector model measures similarity between documents by using vectors for the documents and for the queries
- By declaring a set of document relevant, we may move the query closer to them
- Let us use the following notation:
  - Dr, set of relevant docs, identified by the user, among the ones retrieved
  - Dn, set of non-relevant docs among the ones retrieved
  - Cr, set of relevant docs in the collection

Term Reweighting in the Vector Model

- If Cr is known in advance (that's not a realistic assumption):

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

  where N is the number of documents

- Realistic possibilities:

$$\vec{q}_{new} = \alpha \vec{q}_{old} + \frac{\beta}{|D_r|} \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \frac{\gamma}{|D_n|} \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j \quad \text{Standard\_Rochio}$$

$$\vec{q}_{new} = \alpha \vec{q}_{old} + \beta \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \gamma \max_{non-relevant}(\vec{d}_j) \quad \text{Ide\_Dec\_Hi}$$

$$\vec{q}_{new} = \alpha \vec{q}_{old} + \beta \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \gamma \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j \quad \text{Ide\_Regular}$$

- One can use $\alpha = \beta = \gamma = 1$, or $\gamma < \beta$ (meaning that the relevant docs are more important than the non-relevant); $\gamma = 0$ is even more strict (positive feedback)

Example Rocchio Calculation

$R_1 = (.030,0.00,0.00,.025,.025,.050,0.00,0.00,.120)$ Relevant docs

$R_2 = (.020,.009,.020,.002,.050,.025,.100,.100,.120)$

$S_1 = (.030,.010,.020,0.00,.005,.025,0.00,.020,0.00)$ Non-rel doc

$Q = (0.00,0.00,0.00,0.00,.500,0.00,.450,0.00,.950)$ Original Query

$\alpha = 1$

$\beta = 0.75$ Constants

$\gamma = 0.25$

$$Q_{new} = \alpha \times Q + \left(\frac{\beta}{2} \times (R_1 + R_2)\right) - \left(\frac{\gamma}{1} \times S_1\right)$$ Rocchio Calculation

$Q_{new} = (0.011,0.000875,0.002,0.01,0.527,0.022,0.488,0.033,1.04)$

Resulting feedback query

Term Reweighting in the Vector Model

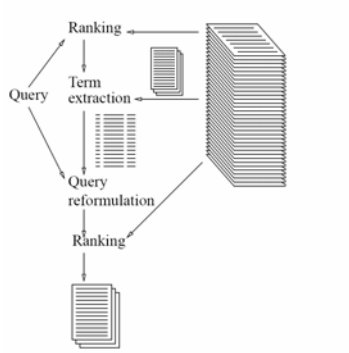
- How good is the modified query ?
- If one measures precision-recall figures using the old and the new queries he/she is likely to find great improvement
- By construction, reformulated query will rank explicitly-marked relevant documents higher and explicitly-marked irrelevant documents lower.
- This should be considered with care: the term reweighting will further increase the precision-recall figures for the documents used in the previous steps
- A better idea is to use the residual collection: remove from the corpus any documents flagged as relevant before.
- 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.

Non-user relevance feedback

- User relevance feedback
  - Clustering hypothesis
    - *known relevant documents contain terms which can be used to describe a larger cluster of relevant documents*
  - Description of cluster built interactively with user assistance
- Other approach
  - Obtain cluster description automatically
  - Identify terms related to query terms
    - Synonyms, stemming variations
    - Terms close to query terms in text
- Local strategies
- Global strategies

Automatic Local Analysis

- Pose initial query
- Rank documents from collection
- Extract candidate expansion terms from supposedly relevant documents
- Select terms and reformulate query
- Do final ranking of documents
- Present results



Local Clustering

- We use the idea of stemming to characterize the similarity between terms
- Let us define, for a query Q:
  - S(s) as the variants of a stem s
    - S(read) = {reading, reads, readable, …}
  - D_l as the local set of returned docs
  - V_l as the local set of all distinct words in D_l
  - S_l as the set of stems of the terms in V_l

Local Clustering

- Local analysis of docs may be too expensive, specially in a WWW context:
  - Retrieving the text of 100 Web documents for local analysis would take too long
- Note that V_l and S_l can be obtained at indexing time (with a somewhat small overhead)
- How to form clusters of terms ?
- We'll see three ideas: association clusters, metric clusters, scalar clusters

Association Clusters

	s_1	s_2	s_3	……..	s_n
s_1	c_{11}	c_{12}	c_{13}	……	c_{1n}
s_2	c_{21}				
s_3	c_{31}				
.	.				
.	.				
s_n	c_{n1}				

c_{uv}: Correlation factor between term u and term v

$$c_{uv} = \sum_{d_j \in D_l} f_{s_u,j} \times f_{s_v,j}$$

f_{ik} : Frequency of stem i in document k

Association matrix S:

$$s_{u,v} = c_{u,v} \quad \text{(Unnormalized) or}$$

$$s_{u,v} = \frac{c_{u,v}}{c_{u,u} + c_{u,u} - c_{u,v}} \quad \text{(Normalized)}$$

- Unnormalized correlation factor favors more frequent terms.
- Normalized correlation factor is 1 if two terms have the same frequency in all documents.

Association Clusters (cont.)

- Given S, how can we build the association clusters ?
- Assume a function $S_u(n)$ which returns the n largest values of the u-th row of S
- $S_u(n)$ returns the values associated with the stems which yielded higher association with stem u across all documents
- Then the set of stems associated with $S_u(n)$ is a cluster of terms around su with no user intervention !

Example: n=6, $|D_l|$=4

F	d_1	d_2	d_3	d_4
k_1	0.8	0.5	0.2	0
k_2	0.5	0	0	0.7
k_3	0.9	0.4	0.8	0
k_4	0	0	0.9	0.7
k_5	0	0	0	0.9
k_6	0.7	0	0.3	0

C	k_1	k_2	k_3	k_4	k_5	k_6
k_1	1.18	0.4	1.08	0.18	0	0.62
k_2	0.4	0.74	0.45	0.49	0.63	0.35
k_3	1.08	0.45	1.61	0.72	0	0.87
k_4	0.18	0.49	0.72	1.3	0.63	0.27
k_5	0	0.63	0	0.63	0.81	0
k_6	0.62	0.35	0.87	0.27	0	0.58

C is left unnormalized.

the closest neighbor of…

- k1 – k3
- k2 – k5
- k3 – k1
- k4 – k3
- k5 – k2 and k4
- k6 – k3

Metric Clusters

- Association correlation does not account for the proximity of terms in documents, just co-occurrence frequencies within documents.
- Metric correlations account for term proximity.

$$c_{u,v} = \sum_{k_i \in V(s_u)} \sum_{k_j \in V(s_v)} \frac{1}{r(k_i, k_j)}$$

$V(s_u)$, $V(s_v)$: sets of keywords which have s_u and s_v as their respective stems.

$r(k_i,k_j)$: Distance in words between word occurrences k_i and k_j

(∞ if k_i and k_j are occurrences in different documents).

Association matrix S:

$s_{u,v} = c_{u,v}$ (Unnormalized) or

$$s_{u,v} = \frac{c_{u,v}}{|V(s_u)| \times |V(s_v)|} \quad \text{(Normalized)}$$

Metric clusters are in general more effective than associative clusters

Scalar Clusters

- Consider the rows $\vec{S}_u$ and $\vec{S}_v$ of the association matrix S – if they are "similar" then stems u and v are likely to be similar as well.
- For example consider the following documents (overly simplified):
  - Document 1: "the last *reading* was very high"
  - Document 2: "the last *measurement* was very high"
  - Terms "reading" and "measurement" are considered as similar
- To measure such similarity we use the cosine between $\vec{S}_u$,$\vec{S}_v$. The new association matrix S is defined as

$$s_{u,v} = \frac{\vec{s}_u \cdot \vec{s}_v}{|\vec{s}_u| \times |\vec{s}_v|}$$

- We can then obtain a cluster for u by re-using the idea of $S_u(n)$

Global Analysis

- Thus far we've used a local analysis in the sense that only the documents returned were taken into account
- Can we take the whole set of documents into account ?
- Indeed we can build a "thesaurus" using all documents, however, this is an expensive task. Fortunately, can be done once and updated incrementally
- Some experiments have shown this to be a worthwhile approach

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 (number of distinct index terms in the document)
  - itf_j: inverse term frequency for document d_j, namely

$$itf_j = \log \frac{t}{t_j}$$

- We associate to term k_i the vector $\vec{k}_i = (w_{i,1}, w_{i,2}, ..., w_{i,N})$

where $w_{i,j}$ is a weight associated to index-document pair$[k_i, d_j]$. These weights are computed as follows

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

Similarity Thesaurus

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

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

- 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
- This computation is expensive
- Global similarity thesaurus has to be computed only once and can be updated incrementally

Similarity Thesaurus

- Query expansion is done in three steps as follows:
  - Represent the query in the concept space used for representation of the index terms

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

    where $w_{i,q}$ is a weight associated to the index-query pair[k_i ,q]
  - Based on the global similarity thesaurus, compute a similarity sim(q,k_v) between each term k_v correlated to the query terms and the whole query q:

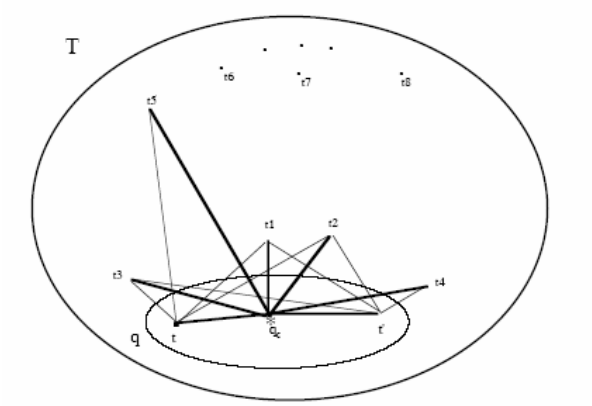
$$sim(q, k_v) = \vec{q} \cdot \overrightarrow{k}_v = \sum_{k_u \in q} w_{u,q} \times c_{u,v}$$

    where $c_{u,v}$ is the correlation factor
  - Expand the query with the top r ranked terms according to sim(q,k_v) to form the expanded query q'
  - We assign a weight w_v to each expansion term k_v in the query q':

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

  - The expanded query q' is then used to retrieve new documents to the user

Similarity Thesaurus



- Query q has two terms t, t'
- q_c the query concept defined by the weighted sum of t, t' (see last overhead)
- Terms t_1, t_2 most similar to query concept: expansion of q using these terms
- Expansion in previous methods has been based on correlation to single query terms: expansion of q with terms t_3 and t_4.

Query Expansion

- Is it economically viable ?
- Local analysis would need to retrieve all documents in the answer (not only “headers”)
- On the one hand, it might be too expensive to be processed on the client due to the network contention
- Further to that, processing it in the server (which has the documents) may be problematic as the system’s throughput is bound to decrease
- Global analysis is “cheaper”, can be done incrementally and seems to be effective