Information Retrieval and Organisation

Dell Zhang

Birkbeck, University of London

2014/15
Relevance Feedback and Query Expansion
Motivation

- How to improve the recall of a search (without compromising precision too much)?
  - “aircraft” in query doesn’t match with “plane” in document
  - “heat” in query doesn’t match with “thermodynamics” in document
- Two general approaches for increasing recall through query reformulation:
  - Local methods (query-dependent):
    e.g., relevance feedback
  - Global methods (query-independent):
    e.g., query expansion
Relevance Feedback

- Basic idea:
  - The user issues a (short, simple) query
  - The system returns an initial set of retrieval results
  - The user marks some returned documents as relevant or not relevant
  - The system computes a better representation of the information need based on the user feedback
  - The system displays a revised set of retrieval results
  - This can go through one or more iterations
  - We will use the term *ad hoc retrieval* to refer to regular retrieval without relevance feedback.
Relevance Feedback

- Example: Content based Image Retrieval (CBIR)
Relevance Feedback

- Results for Initial Query
Relevance Feedback

- User Feedback: Select What is Relevant
Relevance Feedback

- Results After Relevance Feedback
The Rocchio Algorithm

- The classic algorithm for implementing relevance feedback
- Incorporates relevance feedback information into the Vector Space Model
- It does so by “fiddling around” with the query vector $\mathbf{q}$: given a set of relevant documents and a set of non-relevant documents
  - It tries to maximize the similarity of $\mathbf{q}$ with the relevant documents
  - It tries to minimize the similarity of $\mathbf{q}$ with the non-relevant documents
The Rocchio algorithm tries to find the optimal position of the query vector:
Formal Definition

- Given a set $D_r$ of relevant docs and a set $D_{nr}$ of non-relevant docs, Rocchio chooses the query $\vec{q}_{opt}$ that satisfies

$$\vec{q}_{opt} = \max_{\vec{q}} [\text{sim}(\vec{q}, D_r) - \text{sim}(\vec{q}, D_{nr})]$$

where $\text{sim}(\vec{q}, D)$ is the (avg) cosine measure

- Closely related to maximum separation between relevant and nonrelevant docs

- This optimal query vector is:

$$\vec{q}_{opt} = \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j$$
Centroids

- \( \frac{1}{|D|} \sum_{d \in D} \vec{v}(d) \) is called a centroid
- The centroid is the centre of mass of a set of points
- Recall that we represent documents as points in a high-dimensional space.
Any Problem?

- So now we can do a perfect modification of the query vector?
- Unfortunately, that’s not quite true . . .
- This would work if we had the full sets of relevant and non-relevant documents
  - However, the full set of relevant documents is not known
  - Actually, that’s what we want to find . . .
- So, how’s Rocchio’s algorithm used in practice?
Rocchio 1971 Algorithm (SMART)

- Used in practice:

\[
\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j
\]

- \(q_m\): modified query vector;
- \(q_0\): original query vector;
- \(D_r\) and \(D_{nr}\): sets of known relevant and nonrelevant documents respectively;
- \(\alpha, \beta,\) and \(\gamma\): weights attached to each term.
- Negative term weights are ignored.
Rocchio 1971 Algorithm (SMART)

- New query (slowly) moves
  - towards relevant documents
  - away from nonrelevant documents

- Tradeoff $\alpha$ vs. $\beta/\gamma$:
  - If we have a lot of judged documents, we want a higher $\beta/\gamma$. 
Probabilistic Relevance Feedback

- Rather than rewriting the query in a vector space, we could build a classifier.
- A classifier determines which classification an entity belongs to (e.g., classifying a document as relevant or non-relevant).
  - One way of doing this is with a Naive Bayes probabilistic model.
  - We can estimate the probability of a term $t$ appearing in a document, depending on whether it is relevant or not.
  - We’ll come back to this when discussing the probabilistic approach to IR.
When Does Relevance Feedback Work?

- User has to have sufficient knowledge to be able to make an initial query, otherwise we’ll be way off target.

- There can be various reasons why initial query may fail (leading to the result that no relevant documents are found):
  - Misspellings
  - Queries and documents are in different languages
  - Mismatch of user’s and system’s vocabulary: e.g. astronaut vs. cosmonaut
When Does Relevance Feedback Work?

- Relevance prototypes are well-behaved, i.e.
  - Term distribution in relevant documents will be similar to that in the documents marked by the users (relevant documents in one cluster)
  - Term distribution in all non-relevant documents will be different

- Problematic cases:
  - Subsets of the documents using different vocabulary, e.g., Burma vs Myanmar
  - Answer set is inherently disjunctive: e.g. irrational prime numbers
  - Instances of a general concept, which often are a disjunction of more specific concepts, e.g. felines (cat, tiger, etc.)
Relevance Feedback: Evaluation

- Relevance feedback can give very substantial gains in retrieval performance
- Empirically, one round of relevance feedback is often very useful, while two (or more) rounds are marginally useful
- At least five judged documents are recommended (otherwise process is unstable)
Relevance Feedback: Evaluation

- Straightforward evaluation strategy:
  - Start with an initial query $q_0$ and compute a precision-recall graph
  - After getting feedback, compute the modified query $q_m$, again compute a precision-recall graph
- This results in spectacular gains: on the order of 50% in Mean Average Precision (MAP)
  - Unfortunately, this is cheating . . .
  - Gains are partly due to known relevant documents (judged by the user) now ranked higher
Relevance Feedback: Evaluation

- Alternatives:
  - Evaluate performance on *residual collection*, that is the collection without documents judged by user. However, now modified query may often seem to perform worse, as many relevant documents found by IR system don’t count . . .
  - Use two collections: one for initial query, and the other for comparative evaluation.
  - Do user studies: probably the best (and fairest) evaluation method.
Relevance Feedback on the Web

- Relevance feedback has been little used in web search
  - Exception: Excite web search engine
    - Initially provided full relevance feedback
    - However, the feature was in time dropped, due to lack of use

- What are the reasons for this?
  - Most users would like to complete their search in a single interaction
  - Relevance feedback is hard to explain to the average user (no incentive to give feedback)
  - Web search users are rarely concerned with increasing recall
Pseudo Relevance Feedback

- The technique of *pseudo relevance feedback* (aka *blind relevance feedback*), automates the manual part of relevance feedback
  - Use normal retrieval to find an initial set of most relevant documents
  - Assume that the top-\(k\) ranked documents are relevant, use these as relevance feedback

- This automatic technique mostly works

- However, it can lead to *query drift*:
  - Example: query is about copper mines and the top documents are mostly about mines in Chile, then pseudo relevance feedback may retrieve mainly documents on Chile.
Indirect Relevance Feedback

- The technique of *indirect relevance feedback* (aka *implicit relevance feedback*) uses indirect sources of evidence.
- Usually less reliable than explicit feedback, but more useful than pseudo relevance feedback.
- Ideal for high volume systems like web search engines:
  - Clicks on links are assumed to indicate that the page is more likely to be relevant.
  - Click-rates can be gathered globally for *clickstream mining*.
Query Expansion

- In (global) query expansion, the query is modified based on some global resource, i.e. a resource that is not query-dependent.
- Main information we use: (near-)synonymy
  - A publication or database that collects (near-)synonyms is called a *thesaurus*.
  - We will look at two types of thesauri: manually created and automatically created.
Query Expansion: Example
Thesaurus-based Query Expansion

- For each term \( t \) in the query, expand the query with the words semantically related with \( t \) in the thesaurus.
  - Example: hospital → medical
- Generally increases recall
- But can decrease precision, particularly with ambiguous terms:
  - interest rate → interest rate fascinate evaluate
Manual Thesaurus

- Maintained by publishers (e.g. PubMed)
- Widely used in specialized search engines for science and engineering
- It’s very expensive to create a manual thesaurus and maintain it over time
- Roughly equivalent to annotation with a controlled vocabulary.
Manual Thesaurus: Example

PubMed Query:

("neoplasms"[MeSH Terms] OR cancer[Text Word])
Automatic Thesaurus

- It is possible to generate a thesaurus automatically by analysing the distribution of words in documents or by mining query logs
  - Fundamental notion: similarity between two words
  - Definition 1: Two words are similar if they co-occur with similar words.
  - Definition 2: Two words are similar if they occur in a given grammatical relation with the same words.
    - You can harvest, peel, eat, prepare, etc. apples and oranges, so apples and oranges must be similar.
  - The former is more robust, while the latter is more accurate.
## Automatic Thesaurus: Example

<table>
<thead>
<tr>
<th>Word</th>
<th>Nearest Neighbours</th>
</tr>
</thead>
<tbody>
<tr>
<td>absolutely</td>
<td>absurd, whatsoever, totally, exactly, nothing</td>
</tr>
<tr>
<td>bottomed</td>
<td>dip, copper, drops, topped, slide, trimmed</td>
</tr>
<tr>
<td>captivating</td>
<td>shimmer, stunningly, superbly, plucky, witty</td>
</tr>
<tr>
<td>doghouse</td>
<td>dog, porch, crawling, beside, downstairs</td>
</tr>
<tr>
<td>makeup</td>
<td>repellent, lotion, glossy, sunscreen, skin, gel</td>
</tr>
<tr>
<td>mediating</td>
<td>reconciliation, negotiate, case, conciliation</td>
</tr>
<tr>
<td>keeping</td>
<td>hoping, bring, wiping, could, some, would</td>
</tr>
<tr>
<td>lithographs</td>
<td>drawings, Picasso, Dali, sculptures, Gauguin</td>
</tr>
<tr>
<td>pathogens</td>
<td>toxins, bacteria, organisms, bacterial, parasite</td>
</tr>
<tr>
<td>senses</td>
<td>grasp, psyche, truly, clumsy, naive, innate</td>
</tr>
</tbody>
</table>
Summary

- Users can give feedback
  - on documents: more common in relevance feedback
  - on words or phrases: more common in query expansion
- Relevance feedback can also be thought of as a type of query expansion, as we add terms to the query
  - The terms added in relevance feedback are based on “local” information in the result list.
  - The terms added in query expansion are based on “global” information that is not query-specific.
Summary

- Relevance feedback has been shown to be very effective at improving relevance of results
  - Its successful use requires queries for which the set of relevant documents is medium to large
  - Full relevance feedback often onerous for users; its implementation not very efficient in most IR systems
- Query expansion is often used in web-based or highly specialized IR systems
  - Less successful than relevance feedback, though may be as good as pseudo-relevance feedback
  - Easier to understand for users