Advances in IR

Burloiu Calin-Andrei
Banu Valerian
Dita Adrian
Dan Cioiu
Harutyunyan Paruyr

ISI / 2011

Overview

- Query expansion
- Query reformulation
- Relevance feedback
- Results ranking
- Conclusions and directions

Query Expansion

What is Query Expansion?

- The process of **reformulating** a seed query to improve retrieval performance
- Evaluating a user's input and expanding the search query to match additional documents
- *How to automatically expand the query?*
  - Global techniques
  - Local feedback
- QE can be used for cross-language information retrieval
- In interactive systems, terms must be ranked

QE Using Global Analysis

- **Terms** are added from the global corpus of data based on word relationships
- *Concepts* = nouns or groups of nouns
- *Context of a concept* = all the words that co-occur in documents with that word
- **The global context** of a concept can be used to determine similarities between concepts
- QE: the initial concept is run against the concept database => the global context (a ranked list of concepts)
- The top ranking concepts are added to the query, but still weight less than the initial words
- Global analysis is robust and provides a thesaurus-like model
- Expensive in terms of disk space and computing time

QE Using Local Feedback

- Basic idea: **analyze top documents retrieved by initial query**
- Terms from these documents are clustered and treated as quasi-synonyms
- After matching them against the original query, term occurrence probabilities are recalculated
- Query weights distribution is altered, but terms remain the same
- Alternative: combine with global analysis – build a thesaurus using only best passages from top ranked docs
- Efficient because it’s based on high ranked documents
- Slower at run-time, need extra search

Local vs. Global Evaluation

- Tests were performed by Xu and Croft [1996] on two 2GB datasets and 50+ queries each
- Local feedback performed 15% better than global analysis
- Possible reasons:
  - Frequent concepts are ignored
  - Terms with many co-occurrences are added to the query by global-based algorithms
- Combination of local and global analysis performs ~4% better than classic local feedback

QE in Cross-language IR

- Performing query translations:
  - Machine translation techniques
  - Parallel corpora
  - **Bilingual dictionaries**
- First two are very labor intensive
- Third is doable, but the system must select the proper sense from the dictionary
- This is done by computing the *term proximity score*
  - Collective similarity between each candidate term and all terms from the query

QE in Cross-language IR (cont.)

- The proximity score for a term x depends on:
  - The similarity* between x and each member of the query
  - The inverse document frequency (idf) of x
  - The idf of each term from the query
  - The total number of terms from the query
  - The number of documents containing x
- The query is expanded with terms having a proximity score above a certain threshold
- Terms with non-zero similarity to each member of the query taken individually are taken into consideration

* computed by counting co-occurrences in corpus docs

QE in Cross-language Evaluation

- Tests were run by Adriani and van Rijsbergen (1999) on a 748 MB corpus of English data, and using three dictionaries: Spanish, German and Indonesian
- First stage:
  - Foreign language queries were plainly translated by the software into English
- Proximity score is added (without QE):
  - Precision was improved by 48%, 40%, 28%
- **QE is added:**
  - **Precision was improved by 0.3%, 4%, 6%**

Retrieval Improved by Relevance Feedback

- **Relevance Feedback**
  - results marked as relevant improve retrieval
  - methods:
    - **query reformulation / expansion**
    - reranking

- **Feedback procedures**
  1. **Vector space** model
  2. **Probabilistic** model
     - less effective
     - computationally more demanding
- **Query Expansion**
  - terms from relevant items are added to the query

Vector Space Model for Feedback

- Both the **document index** (*D*) and the **query** (*Q*) are represented as **vectors of terms**
  - $D = (d_1, d_2, \ldots, d_t)$ $Q = (q_1, q_2, \ldots, q_t)$
  - each term has a **weight** in the vectors

- **Relevance score** of document *D* for query *Q*
  - **scalar product** between *D* and *Q*

- Vectors of documents marked as relevant or not modify the initial query vector
  - => **feedback**

Probabilistic Model for Feedback

- **Relevance score** depends on:
  - p_i
    - Probability that a term exists in a **relevant** document
  - u_i
    - Probability that a term exists in a **non-relevant** document

$$\text{sim}(D, Q) = \sum_{i=1}^{t} d_i \log \frac{p_i(1 - u_i)}{u_i(1 - p_i)} + \text{constants}$$

$$p_i = Pr(x_i = 1 \mid \text{relevant})$$

$$u_i = Pr(x_i = 1 \mid \text{nonrelevant})$$

Interactive Query Expansion

- New terms are presented to the user in some reasonable order
- It involves relevance weighing of terms
- Relevance is computed based on the quantitative and qualitative components of the results
- Relationship between term frequency and term value:
  - very frequent terms are not very useful,
  - middle frequency terms are quite useful,
  - infrequent terms are likely to be useful but not as much as the middle frequency terms,
  - very infrequent terms are useful, but absent most of the time.
- Several **algorithms** for term ranking exist

Term Ranking Algorithms

- **F4 algorithm** (and all variations)
- **Porter**'s algorithm
- **EMIM** states that:
  - terms may not be distributed independently of each other
  - probability is computed per pair
- ***wt(pt – qt)***
  - w_t is calculated using a F4 variant
  - p_t - q_t is used to quantify the relationship between term frequency and term value
- **ZOOM** term frequency ranking is based on automatic frequency of concepts

Term Ranking Algorithms (cont.)

- The evaluation of these ranking systems was performed by Efthimiadis [1993]
- *w(p - q)* and EMIM outperformed all other algorithms with over 60%
- After analyzing the results, the conclusion is that a simple ranking system could replace all these algorithms, by:
  - Ranking items according to their frequency of occurrence in the relevant documents set
  - Resolving ties according to their term frequency from low to high frequency

Query Reformulation

Query Reformulation (QR)

- The process of iteratively modifying a query to improve the quality of a search engine's results.
- Search engines support users in this task:
  - **explicitly**: by suggesting related queries or query completions.
  - **implicitly**: by expanding the query.
- Successful refinements are closely related to the original query:
  - **syntactically**
  - **semantically**
- Several tactics: generalization, replacement with synonyms, parallel movement (~50% of the time), specification (~30% of the time), correction.

How can we find the best QRs?

Semantic Similarity

- Similar words (queries) occur in similar contexts.
- Pointwise Mutual Information (PMI): a measure of the association between two terms or queries.
- Manipulations of PMI:
  - Joint normalization
  - Specialization normalization
  - Generalization normalization

$$PMI(x,y) = log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$

$$PMI(J)(x,y) = \frac{PMI(x,y)}{-log(p(x,y))}$$

$$PMI(S)(x,y) = \frac{PMI(x,y)}{-log(p(x))}$$

$$PMI(G)(x,y) = \frac{PMI(x,y)}{-log(p(y))}$$

Example

- “apple" -> ”mac os": PMI(G)=0.2917, PMI(S)=0.3686;
  - more evidence of a specialization.
- “ferrari models" -> “ferrari": PMI(G)=1, PMI(S)=0.5558;
  - the target is a “perfect" generalization of the source.

How can we find the best QRs? (cont.)

Syntactic Similarity: **Levenshtein distance** (edit distance)

- Models the similarity between two queries as a function of the edit operations (insertion, deletion, substitution).
- Measures well the user's conservative disposition.
- Different edit distance models, with different cost functions.
- Order-free versions: the terms are sorted.
  - Eg. "pizza Brooklyn" vs. "Brooklyn pizza"

Other:

- Generalized Levenshtein distance: takes into account semantic similarities between terms.
- Generative Model (Oommen and Kashyap).
- QRT, Neural Networks etc.

Evaluating verbose query processing techniques

- Verbose or long queries are a small but significant part of the query stream in web search
- Average query length is 2.4 words
- 10% of all queries have > 5 words
- Search engines perform well on short (keyword) queries => need for query pre-processing

Techniques:

Stopword Removal

Eg. "Can i work while study in Europe" => ""work study Europe"

- Used lists
  - Standard INQUERY stopword list
  - constructed lists from Yahoo! Answers by applying IDF weighting to the set of verbose queries => Top100 and Top200
- Problem with stopwords*:*

  "i would like to know the origin of the phrase to be or not to be" becomes *"know origin phrase"* □

Techniques

Noun phrase detection

- Verbose query ~ sentence or phrase
- MontyLingua used to extract nouns:
- Eg. “give me a name for my next horror script”. Two methods:
  - *give “me” “a name” for “my next horror script”*
  - *me a name my next horror script*

Key Concept detection

- The key concept is a short set of sequential words that express an important idea contained within the query.
- For extracting a key concept a classifier based on AdaBoost.M1 meta-classifier using C4.5 Decision Trees
- Eg. “*i read that ions cant have net dipole moments why not*” => *“i ions net dipole moments”*

Techniques & Evaluation

Stop Structure

- stop phrase = a phrase which provides no information about the information need
- *stop structure* = stop phrase which begins at the first word in the query.
- Eg. "*if i am having a lipid test can i drink black coffee*" =>

  "lipid test can i drink black coffee"
- Automatic stop structure classification -> A sequential binary class tagging problem. (Yamcha classifier)

Evaluation

- The use of quotations in formulating queries is clearly not effective.
- Both noun phrase and key concept identification produce significant improvements, when combined with stopword removal.
- The most effective technique = the removal of stop structure.
- Stop structure can be effectively automatically classified.
- Future work may include investigating the automatic removal of stop phrases that may occur at any position within a verbose query

More on Relevance Feedback

Improve Relevance by Implicit Feedback

- Goal: adapting the retrieval system to particular groups of users.
- Implicit measures: user clicks, time spent on page, session duration, pupil dilatation etc.
- The users' clicking decisions are influenced by the relevance of the results -> clicked documents are relevant (but not necessarily accurate).
- Users' clicking decisions are biased:
  - by the order in which the results are presented: users trust in the search engine's ability to estimate the relevance of a page.
  - by the overall quality of the result set.
- Mission: use machine learning to automatically tune retrieval functions (use implicit feedback measures as training data).

Observed User Behavior

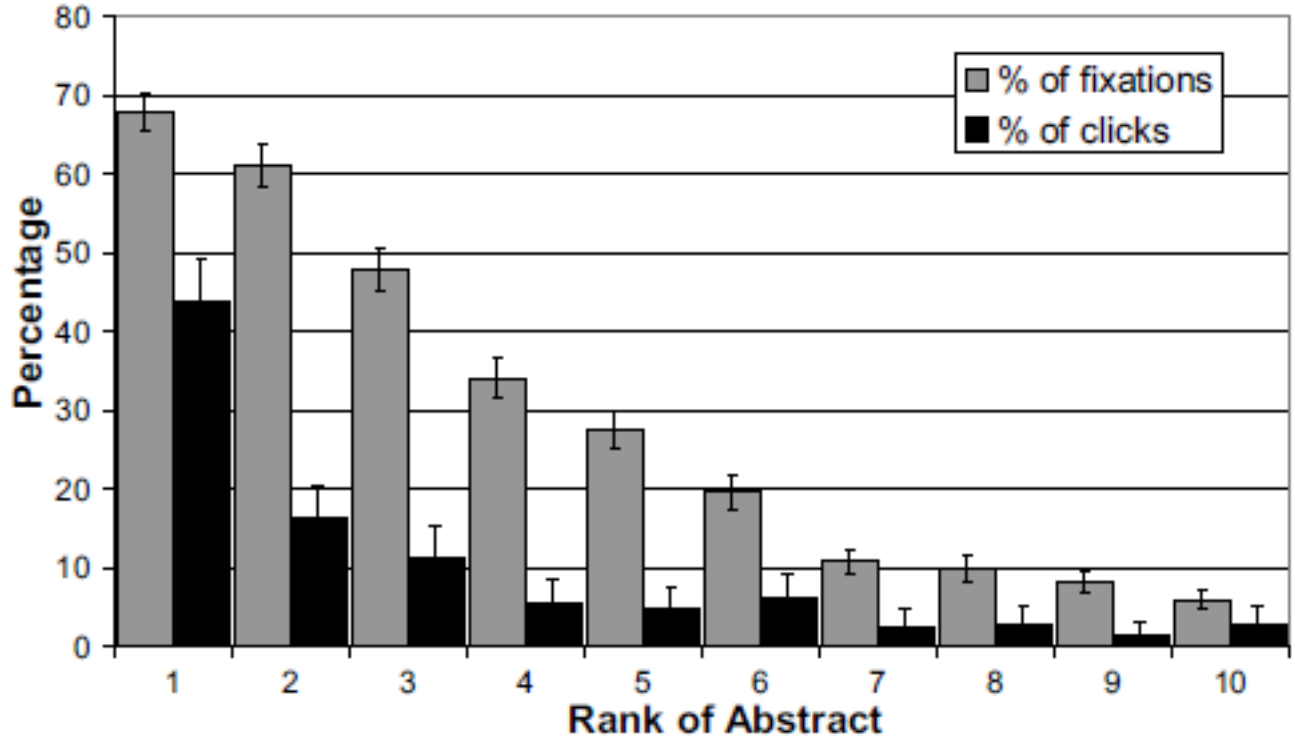

Fig. 1. Percentage of times an abstract was viewed/clicked depending on the rank of the result.

Observed User Behavior

- Users tend to read the results from top to bottom.
- Individuals tend to view the first and second-ranked results right away (within the second or third eye fixation), with a big gap before viewing the third-ranked abstract.
- Most users follow a depth-first search strategy and scan the view-able results quite thoroughly before resorting to scrolling.
- Once the user has started scrolling, rank appears to becomes less of an influence for attention -> results ranked 7 to 10 receive equal attention.
- Users act before they had evaluated all options, trading-off quality against exploration effort.

Relative feedback from clicks

- Mission: use click-through data as training data for better ranking.
- Identify the relevance of a result from the user's choice among the available options (the results from one page).
- Given the results $(l_1^*, l_2, l_3^*, l_4, l_5^*, l_6, l_7)$ where (*) are clicked links, in the following order: l_3, l_1, l_5.
- Strategies:
  - **Click > Skip Above**: r(l3) > r(l2), r(l5) > r(l2), r(l5) > r(l4)
  - **Last Click > Skip Above**: r(l5) > r(l2), r(l5) > r(l4)
  - **Click > Earlier Click**: r(l1) > r(l3), r(l5) > r(l3), r(l5) > r(l1)
  - **Click > Skip Previous**: r(l5) > r(l4), r(l3) > r(l2)
  - **Click > No-Click Next**: r(l1) > r(l2), r(l3) > r(l4), r(l5) > r(l6)
- Best results:
  - **Last Click > Skip Above**
  - **Click > Skip Above**
- They produce preference between the top few results.

Relative feedback from clicks (cont.)

- Users typically reformulate queries to refine the results.
- Given a query chain (QC) of 4 queries:
  - q1: l_{11}, l_{12}, l_{13}, l_{14}, l_{15}, l_{16}, l_{17}
  - q2: l_{21}^{*}, l_{22}, l_{23}^{*}, l_{24}, l_{25}^{*}, l_{26}, l_{27}
  - q3: l_{31}, l_{32}^{*}, l_{33}, l_{34}, l_{35}, l_{36}, l_{37}
  - q4: l_{41}^{*}, l_{42}, l_{43}, l_{44}, l_{45}, l_{46}, l_{47}
- Strategies:
  - Click > Skip Earlier QC:
  - Last Click > Skip Earlier QC
  - Click > Click Earlier QC
  - Click > TopOne NoClickEarlier QC
  - Click > TopTwo NoClickEarlier QC
  - TopOne > TopOne Earlier QC
- Best results: "Click > TopOne NoClickEarlier QC" and "Click > TopTwo NoClickEarlier QC".

Negative Relevance Feedback

- **More problematic** than positive feedback
  - negative documents **do not cluster** together
  - negative documents **may distract in different ways**

- **Problem Formulation**
  - For a **difficult query** all the first results are **non-relevant** => **negative examples**
  - **Rerank** the next unseen results based on this negative feedback

Negative Relevance Feedback

- **Models**
  - Vector Space Models
  - Language Models
    - more effective

- **Strategies**
  - **Query modification**
    - less effective
  - **Score combination**
    - positive and negative query representations maintained **separately**
    - scores are **combined**

Score combination

- **Heuristics**
  - only **penalize** documents which are most similar to the negative query model

- **Computing negative query representation**
  1. negative information combined in a **single query**
  2. negative information captured with **more than one query** and **aggregated**

Relevance feedback in image retrieval

- *“An image is worth a thousand words” and the machine does not know what these words are*
- *Judging a document takes time, while an image reveals its content almost instantly to a human observer*, which makes the feedback process faster and more sensible for the end user.
- Typical scenario for relevance feedback in content-based image retrieval:
  - **Step 1.** Machine provides an initial retrieval results, through query-by-keyword, sketch, or example, etc.
  - **Step 2.** User provides a judgment on the currently displayed images as to whether, and to what degree, they are relevant or irrelevant to her/his request.
  - **Step 3.** Machine learns and tries again. Go to step 2.

Relevance feedback in image retrieval

A classification problem with a twist:

- Small sample issue (<20 per round of interaction)
- Asymmetry in training sample (too many positives or negatives)
- Real time requirement

Variants of relevance feedback algorithms

- *User model:*
  - *What to look for?*
    - *category or target search*
  - *What to feed back?*
    - *binary or labeled answer*
  - *Greedy vs. cooperative*

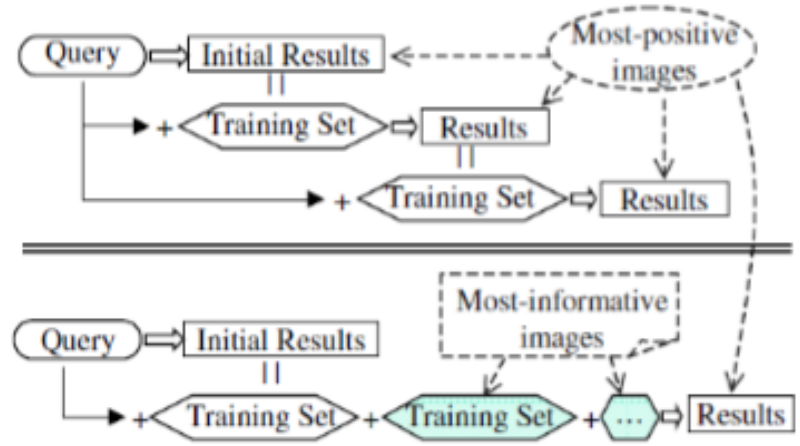

Fig. 1. *Top*: the *show-me-the-results* scenario for a greedy user, where further training, if any, will be performed on the current best results. *Bottom*: the *ask-me-questions* scenario for a co-operative user, where the machine can actively select more than one screen of samples to be added *sequentially* into its training set

Variants of relevance feedback algorithms

- ***Algorithmic assumptions:***
  - *Feature selection and representation:* color histogram or moments, texture, shape, and structure features
  - *Class distribution:* what distribution to impose on the target classes
- *Data structure: for a hierarchical tree structure, learning is difficult, a tradeoff between speed and accuracy*
- ***Objective functions and learning machines***
- ***Negative examples***
- *Singularity issue in sample covariance matrix:* the number of training examples is smaller than the dimensionality of the feature space
- *Pre-clustering and long-term learning:* the rationale of relevance feedback contradicts that of pre-clustering.
- *Global vs. regional query*
- *Incorporating textual annotations:* “Semantic gap” between high-level concepts in the user’s mind and low-level features extracted by the machine is so wide in many cases that the use of keywords

Relevance feedback in image retrieval

- Targeted at a very specific application scenario, namely *the real-time learning from user interactions during information retrieval*, relevance feedback as a classification or learning problem possesses very unique characteristics and difficulties

- Even though labeled the same as “relevance feedback” algorithms, many schemes were developed under quite different application or user assumptions.

- Through the comparison and analysis of existing literature, some common problems across different approaches, as well as some misconceptions have been discovered

Results Ranking

Context-Aware Ranking

- Integrate **context information** into the **ranking model**
- **Context information**
  - previous queries (same session)
  - answers clicked or skipped

- **Personalized search**
  - *individual* users, short / long histories
- Context-aware search
  - *all* users, short histories

User Search Relations

1. **Reformulation**
   - Query 1: "homes for rent in atlanta"
   - Query 2: "houses for rest in atlanta"
2. **Specialization**
   - Query 1: "time life music"
   - Query 2: "time life Christian CDs"
3. **Generatlization**
   - Query 1: "free online Tetris"
   - Query 2: "Tetris game"
4. **General Association**
   - Query 1: "Xbox 360"
   - Query 2: "FIFA 2010"

Context-aware Ranking

- For each search relation pattern some previous *results* are either *demoted* or *promoted* for the next query

- A **learning-to-rank** approach is used
  - an **SVM** model for classification on the preference between a pair of documents

- Offline training
  - search sessions
  - user judges

Ranking using Multiple Document Types in Desktop Search

- A typical desktop environment contains many document types (email, presentations, web pages, pdfs, etc.) each with different metadata.
- Predicting which types of documents a user is looking for in the context of a given query is a crucial part of providing effective desktop search.
- Retrieval effectiveness of a desktop search system can be enhanced by improving type prediction performance.

Retrieval model:

- Type-specific retrieval:
  - Probabilistic retrieval model for semi-structured data
- Type prediction
  - will produce scores for each sub-collection
- Result merging
  - Uses the Cori algorithm for merging based on sub-collection retrieval and collection scores

Figure 1: Suggested retrieval model for desktop search.

Type prediction methods

- Existing methods: Query-likelihood of Collection, Query-likelihood of Query Log, Geometric Average, ReDDE, Query Clarity, Dictionary-based Matching
- **Using Document Metadata Fields for Type Prediction**
  - Field-based collection query likelihood (FQL)
  - extends the collection query likelihood model for collection scoring by combining the query likelihood score for each field of the collection instead of using the score for the whole collection.
  - tries to infer the mapping between a use query and each collection by combining mapping probabilities for the fields of each collection
- **Combining Type Prediction Methods**
  - Grid-search of Parameter Values
  - Multi-class Classification
  - Rank-learning Method

Collections and results

- **Pseudo-desktop** - collected documents with similar characteristics to a typical desktop environment and generated queries by statistically taking terms from each of the target documents.
- **DocTrack Game** – used to collect a large quantity of known-item queries in a reasonably realistic setting
- Increased sub-collection retrieval and type prediction reflects the performance and quality of the final rank list
- A new type prediction method (FQL) that exploits type-specific metadata and show that the new method has better performance than a state-of-the-art collection scoring method
- Results show that a combination of collection scoring methods can improve the performance even further.

Future Directions

- http://www.youtube.com/watch?v=mTBShTwCnD4
  - Ranking results of different type (ex.: images, web pages, videos etc.)
  - Non-text search queries (like images)
  - Semantic Web
- http://insidesearch.blogspot.com/2011/12/search-quality-highlights-new-monthly.html

Fun and useful:

- Tips & Tricks for Students Conducting Online Search: http://www.hackcollege.com/blog/2011/11/23/infographic-get-more-out-of-google.html

Other worth-mentioning techniques

- QE using Apriori algorithm
  - Rungsawang et al.
  - association rule discovery
  - all rules are given confidences, right-hand terms are ranked according to their confidence level
- Probabilistic QE using query logs
  - Cui et al.
  - multiple queries are mapped to same document
  - based on user and search engine history
  - "blind" relevance assumption
- Stemming
  - can be used when expansion is less rewarding
  - works for Arabic queries (Aljlayl et al.)

Cloaking

Search Engine Optimization Spam

Cloaking

- Cloaking
- Is cloaking ethical?
- Google and Cloaking
- Cloaking Software

- What is cloaking?
  http://www.youtube.com/watch?feature=player_embedded&v=QHtnfOgp65Q
- How cloaking works?

- Cloaking types
  - IP based
  - User-agent based

- Is cloaking ethical?
- How do spammers misuse our websites?

Google and Cloaking

- An example how cloaking can actually help Google
- How can we find out if spammers abuse our site?

Cloaking Software

Fantomas is widely recognized as the best cloaking software around.

Image Search Engine

Work on optimizing images for the web is divided into two stages:

- Preparing images in editor
- The compression process

Image searches are most rapidly growing areas of search today. Over 360,000,000 searches per month in all major search engines: Google, Yahoo!, Ask, MSN и AOL according to Hitwise.com

In the leading Search Engines (Google, Yahoo and MSN) images integrated with text search results.

Reference: http://3pu.info/image-optimization-for-search-engines/

Optimizing the image has three options

1. Image file name
2. The image attributes title and alt
3. Name of the Image

How to get at the top of results in image search engine?

1. The large images are indexing better(now the normal size is not less than 300pixels)
2. Don't make the transition from picture preview or icons to a large image with JavaScript. It is bad for indexing.
3. Place the theme unique text closet to the picture.

Questions?