# 1.2.5 Probabilistic Information Retrieval

The notion of similarity in the vector space model does not have an explanation how it relates to relevance

- The similarity values are just used to rank

An information retrieval model deals with uncertainty on the user's information needs

- Probability theory provides a principled approach to reason about this uncertainty

Probabilistic IR models attempt to provide an explainable model of relevance




One of the key drawbacks of the vector space retrieval model is the lack of interpretability of the similarity values. This gave rise to the development of probabilistic retrieval models, that attempt to determine relevance as a probabilistic concept with an explainable probabilistic model.

# Query Likelihood Model

Given query $q$, determine the probability $P(d|q)$ that document $d$ is relevant to query $q$

Bayes Rule $P(d|q) = \frac{P(q|d)P(d)}{P(q)}$

Assumptions

- $P(d)$, the probability of a document occurring is uniform across a collection
- $P(q)$ is the same for all queries

Thus: $P(d|q)$ can be derived from $P(q|d)$, the query likelyhood





In probabilistic retrieval the objective is to introduce a notion of probability that relates to the relevance of a document d with respect to a query q.

Since we can assume that the probability of a document to occur in a collection is constant (which is correct assuming all documents are different), and the probability of a query to occur is the same for all documents, using Bayes rule, the problem of determining whether a document is relevant for a query is equivalent to the problem of determining whether a query is relevant to a document. The latter probability $P(q|d)$ is also called the query likelihood.

So probabilistic IR can be framed as the question of what is the probability for a specific query to occur, given a specific document. One way to understand why the problem is formulated this way, is that it is more feasible to derive a rich model from a document, than from a query, which contains little information.

## Language Modeling

Query likelihood: determine $P(q|d)$

Assume each document $d$ is generated by a Language Model $M_d$

- a language model is a mechanism that generates the words of the language

Then $P(q|d)$ can be interpreted as the probability that the query $q$ was generated by the language model $M_d$



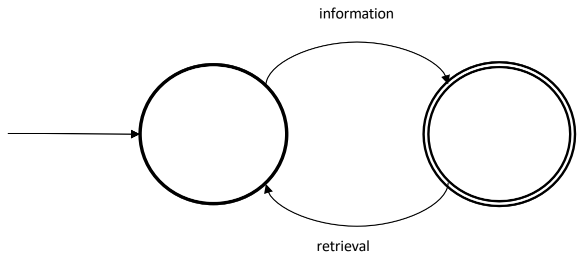


The concept of query likelihood gives now rise to the following approach to capture relevance of a document to a query. We assume that documents are the result of language model. A language model is a (in general probabilistic) process that produces text, and a given document d is assumed to be produced by its specific language model $M_d$. Then the problem of retrieval can be viewed in the following way: if a query is relevant to a document, it should have been produced by the same language model as the document. Using this argument, the query likelihood corresponds to the probability that the query has been produced by the same language model as the document.

Let's have now a more detailed look in what a language model is and how we use it implement this approach practically.

# What is a Language Model?

Deterministic language model = automaton = grammar




This model can produce:

"information retrieval"

"information retrieval information retrieval"

It cannot produce:

"retrieval information"

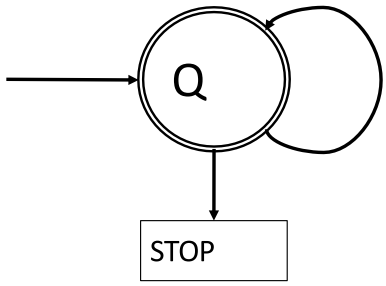



A very simple example of a generative language model is a deterministic automaton. Deterministic automatons are used to recognize or produce regular languages.

# Probabilistic Language Model

Unigram model: assign a probability to each term to appear

- More complex models can be used, e.g., bigrams




| Model $M_1$ | | Model $M_2$ | |
|---|---|---|---|
| STOP | 0.2 | STOP | 0.2 |
| the | 0.2 | the | 0.15 |
| a | 0.1 | a | 0.12 |
| frog | 0.03 | frog | 0.0002 |
| toad | 0.03 | toad | 0.0001 |
| said | 0.02 | said | 0.01 |
| likes | 0.015 | likes | 0.01 |
| dog | 0.01 | dog | 0.04 |

Two different language models derived from 2 documents




Instead of using a deterministic automaton, we can also use a probabilistic state automaton, in other words, a Markov process. In the simplest case the automaton has a single state, and every state transition emits with a certain probability one term out of a vocabulary. In addition, the automaton can stop with a certain probability. The table captures the transition probabilities of two possible models M1 and M2. In the two models, the probability to stop is given as P(STOP|Q) = 0.2.

## Probability to Create a Query

What is the probability that a query $q$ has been generated by model $M$?

*Example*: $q$ = the frog said dog STOP

$P(q|M_1)$ = 0.2 * 0.03 * 0.02 * 0.01 * 0.2 = 0.000 000 24

$P(q|M_2)$ = 0.15 * 0.0002 * 0.01 * 0.04 * 0.25 = 0.000 000 003

Retrieval becomes the problem of computing for a query $q$ the probability $P(q|M_d)$ for all the documents $d$




Given a language model for generation of documents, we can compute using that model the probability that a given query q has been generated by the model of a document d. We give an example showing such a computation. With this approach we are now ready to compute query likelihood for all documents of a document collection.

# Learning the Model

Learning the model means we must estimate the probability of a query to occur

First step: estimate how likely a single term $t$ occurs

Maximum Likelihood Estimation (MLE) of probabilities under Unigram Model

$$\hat{P}_{mle}(t|M_d) = \frac{tf_{t,d}}{L_d}$$

where

- $tf_{t,d}$ is the number of occurrences of $t$ in $d$ (term frequency)
- $L_d$ is the number of terms in the document (document length)




For applying the probabilistic retrieval method described before, we need to learn a language model of each document. The learning is performed using Maximum Likelihood Estimation (MLE). In the case of the unigram model, this is a straightforward task. We need to estimate the probabilities of terms to occur in a document. This is done by counting the number of occurrences of terms and normalizing by document length.

## Using the Model

Independence assumption: different terms in a query are assumed to occur independently

$$\hat{P}(q|M_d) = \prod_{t \epsilon q} \hat{P}_{mle}(t|M_d)$$




Based on the estimation of term probabilities we can compute a query probability, by making an independence assumptions on terms. This results in an estimation of the probability of a query to occur under a given document model.

**Consider the document:**
**“Information retrieval is the task of finding the documents satisfying the information needs of the user”**

Using MLE to estimate the unigram probability model, what is P(the|$M_d$) and P(information|$M_d$)?

1. 1/16 and 1/16
2. 1/12 and 1/12
3. 1/4 and 1/8
4. 1/3 and 1/6

## Consider the following document

d = “information retrieval and search”

1. P(information search | $M_d$) > P(information | $M_d$)
2. P(information search | $M_d$) = P(information | $M_d$)
3. P(information search | $M_d$) < P(information | $M_d$)

## Issues with MLE

Problem 1: if the query contains a term not occurring in the document, then $\hat{P}(q|M_d) = 0$ !

Problem 2: this is an estimation! A term that occurs once, might have been "lucky", whereas another one with same probability to occur is not contained in the document

- ➢ need to give non-zero probability to unseen terms!




Applying the described approach to estimate relevance of a document to a query has a practical problem: if the query contains a term not occurring in the document the estimated probability will be unavoidably zero, since one of the factors of the product computing that probability will be zero. In other words, the query cannot be generated by the document model, thus the document is not relevant to the query. This is not only impractical, but also not meaningful from a more another perspective. Since we used MLE to generate the model, we were using the statistics of one specific document, that has been generated by a potentially complex document model. It might be the case that the specific generated document by chance does not contain certain terms that are part of the possible terms in the document according to the document model.

## Smoothing

Idea: add a small weight for terms not occurring in a document

- the weight should be smaller than the normalized collection frequency

$$\hat{P}(t|M_c) \leq cf_t/T$$

where

- $cf_t$ = number of times term t occurs in collection
- $T$ = total number of terms in collection

Smoothed estimate

$$\hat{P}(t|d) = \lambda \hat{P}_{mle}(t|M_d) + (1-\lambda)\hat{P}_{mle}(t|M_c)$$

$M_c$ = language model of the whole collection

$\lambda$ = tuning parameter





To address these problems an approach called smoothing is applied. The idea is to assume that in fact every term potentially could occur in the document generated by its document model, including those that are not part of the actual document; only that the probability of terms not seen in the document is presumably smaller than the probability of the term to occur in the overall document collection. The smoothed estimate then combines the estimated likelihood to occur in the document according to the document model, with the estimated likelihood of a term occurring in the general document collection, modeled as a generic language model using the statistics from the whole document collection.

## Probabilistic Retrieval

With smoothing the relevance is computed as

$$P(d|q) \propto P(d) \prod_{t \epsilon q} ((1-\lambda)P(t|M_c) + \lambda P(t|M_d))$$

From a technical perspective the probabilities are computed using term and document frequencies

- the same data is used as in vector space retrieval

Probabilistically motivated models show generally better performance

- But parameter tuning ($\lambda$) is critical
- $\lambda$ can be query-dependent, e.g., query size




Here we summarize the approach for probabilistic retrieval. From a technical perspective computational cost of probabilistic retrieval is comparable to that vector space retrieval. The computation of the likelihoods for the document models requires to determine term frequencies, so in that sense it is equivalent. For deriving the collection model, the global term frequencies need to be computed, which again is like computing inverse document frequencies in a document collection.

In practice, the fine tuning of the model parameters (in that case $\lambda$) is essential for the model to perform well. It is also possible to make the parameter dependent on the query, in particular on the query size.

# Example

Collection consisting of $d_1$ and $d_2$

$d_1$: Einstein was one of the greatest scientists

$d_2$: Albert Einstein received the Nobel prize

Query q: Albert Einstein

Using $\lambda$=1/2:

$P(q|d_1) = ½ * (0/7 + 1/13) * ½ * (1/7 + 2/13) \approx 0.0057$

$P(q|d_2) = ½ * (1/6 + 1/13) * ½ * (1/6 + 2/13) \approx 0.0195$

Albert Einstein





This is a simple example illustrating the use of probabilistic retrieval. Note that the document lengths of d1 and d2 are 7 and 6, and that the collection length is 13.

## Example: Comparing VS and PR

| | Precision | | | |
|---|---|---|---|---|
| Rec. | tf-idf | LM | %chg | |
| 0.0 | 0.7439 | 0.7590 | +2.0 | |
| 0.1 | 0.4521 | 0.4910 | +8.6 | |
| 0.2 | 0.3514 | 0.4045 | +15.1 | * |
| 0.3 | 0.2761 | 0.3342 | +21.0 | * |
| 0.4 | 0.2093 | 0.2572 | +22.9 | * |
| 0.5 | 0.1558 | 0.2061 | +32.3 | * |
| 0.6 | 0.1024 | 0.1405 | +37.1 | * |
| 0.7 | 0.0451 | 0.0760 | +68.7 | * |
| 0.8 | 0.0160 | 0.0432 | +169.6 | * |
| 0.9 | 0.0033 | 0.0063 | +89.3 | |
| 1.0 | 0.0028 | 0.0050 | +76.9 | |
| Ave | 0.1868 | 0.2233 | +19.55 | * |

Ponte & Croft, 1998





This is a result reported from comparing vector space retrieval with probabilistic retrieval. In this experiment probabilistic retrieval improves precision significantly, in particular for higher values of recall. (LM = language model).

## Properties of Retrieval Models

| | Vector Space Model | Language Model | BM25 (another prob. Model) |
|---|---|---|---|
| Model | geometric | probabilistic | probabilistic |
| Length normalization | Requires extensions (pivot normalization) | Inherent to model | Tuning parameters |
| Inverse document frequency | Used directly | Smoothing and collection frequency has similar effect | Used directly |
| Multiple term occurrences | Taken into account | Taken into account | Ignored |
| Simplicity | No tuning required | Tuning essential | Tuning essential |




Here we compare the characteristics of the vector space model with the probabilistic retrieval model based on language models that we have introduced. BM25 is another probabilistic model, that is today considered as one of the most performant retrieval models.

One aspect that is taken implicitly care off in the probabilistic retrieval model based on language models is normalization for document length. For vector space retrieval, specific modifications of the model are used, as we have seen earlier. For collections with widely varying document lengths this proved to be a useful improvement.

In general, the vector space model is preferred, when a quick and simple solution is sought. For probabilistic models, better performance can be achieved, but this depends on careful parameter tuning.

## 1.2.6 Query Expansion

If the user query does not contain any relevant term, a corresponding relevant document will not show up in the result

Example: query “car” will not return “automobile”

How to add such documents (increase recall)?

**Idea**: System adds query terms to user query!



Users cannot predict or imagine all possible ways of how the concepts they are interested to find in their search can be expressed in natural language. Consequently, even under the vector space retrieval model, relevant results are missed, for the example, when the user does not provide different synonyms (different terms with the same meaning) in a query.

In information retrieval we are most of the time concerned about precision, since the assumption is that there exist many relevant documents. But in certain applications recall is more important. Good examples for this is searching for scientific publications, where only a few papers might be of interest or security-relevant applications where also rare risks need to be detected.

In the following we will see one possible approach to deal with this problem, namely extending the user query automatically by the system with additional query terms.

## Two Methods for Extending Queries

1. Local Approach:

- Use information from **current query results**: *user relevance feedback*

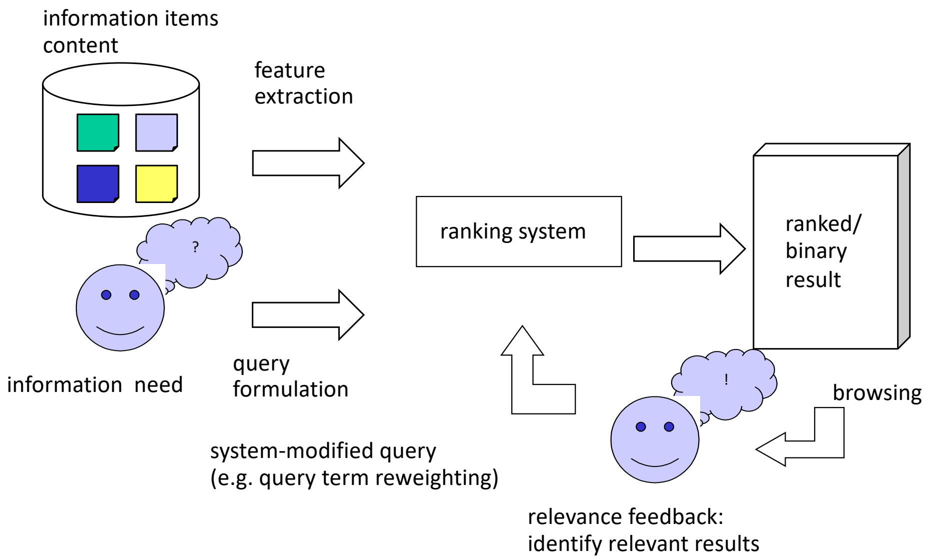
2. Global Approach:

- Use information from a **document collection**: *query expansion*




In the following we will present two types of approaches to query extension. They differ in the way of how new additional query terms are obtained. In the local approach the source of information is the current user query, respectively results produced by answering the user query. In the global approach, the source of information is an existing document collection, either the documents corpus that is queried by the user, or another, external collection of documents.

## 1.2.6.1 User Relevance Feedback



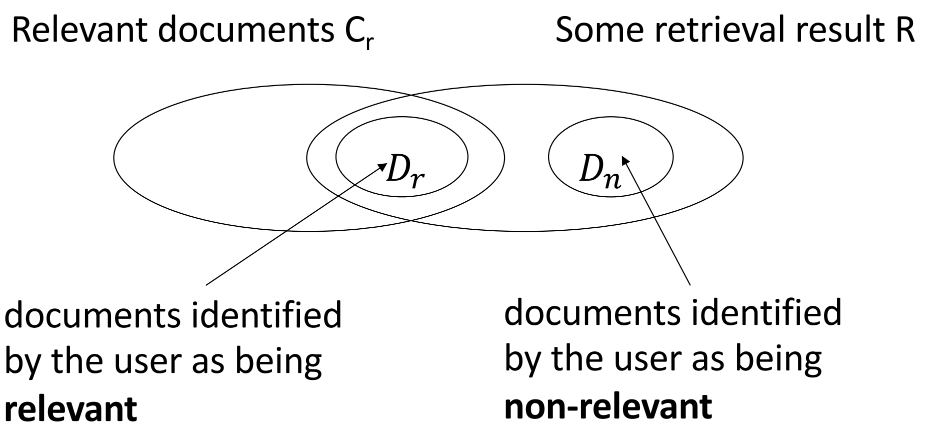



In general, a user does not necessarily know what is his information need and how to appropriately formulate a query. But usually, a user can well identify relevant documents. Therefore, the idea of user relevance feedback is to reformulate a query by taking into account feedback of the user on the relevance of already retrieved documents.

The advantages of such an approach are the following:

- The user is not involved in query formulation, but just points to interesting data items.
- The search task can be split up in smaller steps.
- The search task becomes a process converging to the desired result.

# Feedback from Users






The situation when receiving feedback from users can be described as follows: the retrieval system returns some result set R that is presented to the user. This result set overlaps with the set of relevant documents ($C_r$). The user can the identify within the result set both documents that are relevant and non-relevant. This gives the two feedback sets $D_r$ and $D_n$.

# Rocchio Algorithm

*Rocchio algorithm*: find a query that optimally separates relevant from non-relevant documents

$$\vec{q}_{opt} = \arg\max_{\vec{q}} \left[ \mathrm{sim}\left(\vec{q}, \mu(D_r)\right) - \mathrm{sim}\left(\vec{q}, \mu(D_n)\right) \right]$$

Centroid of a document set

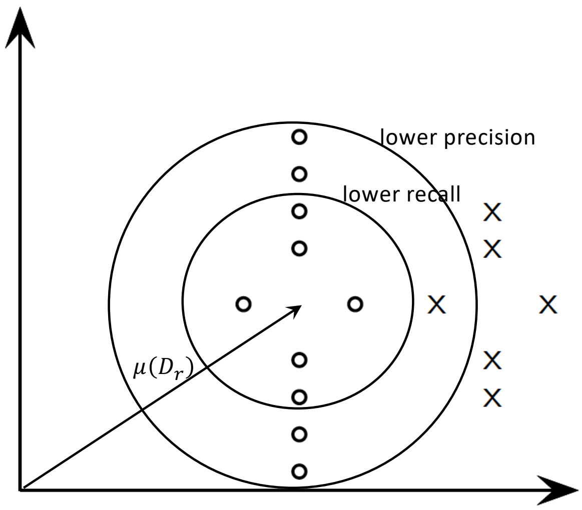
$$\mu(D) = \frac{1}{|D|} \sum_{d \in D} \vec{d}$$





The basic approach for user relevance feedback was introduced by Rocchio. It is based on the observation, that the centroid of all document vectors of a document set D can be considered as the most characteristic representation of the document set. In order to to construct a query $q_{opt}$ that optimally separates relevant from non-relevant documents, such a query has to have maximal similarity with the set of relevant documents, respectively its centroid, and maximal dissimilarity with the set of non-relevant documents, respectively its centroid. This can be achieved by finding a query that maximizes the difference among these two similarity values.

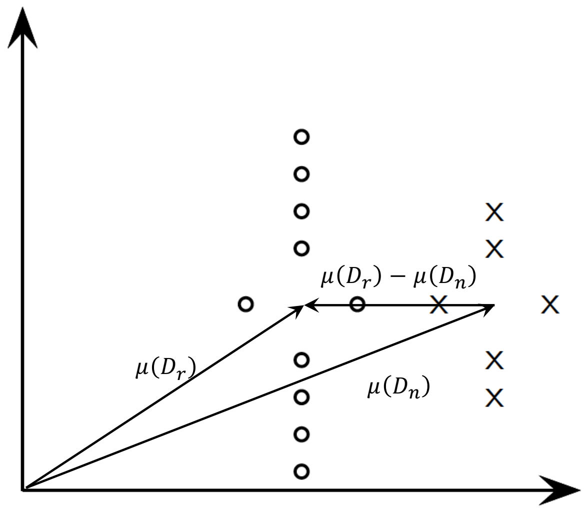
# Illustration of Rocchio Algorithm






We now motivate of how the optimal query vector can be found with an illustration. Assume that the relevant documents are marked by circles, and the non-relevant documents are marked by crosses, and that the vector space has 2 dimensions. When we consider the simply centroid of the relevant documents as a search query, then we see that we cannot achieve optimal precision and recall at the same time.

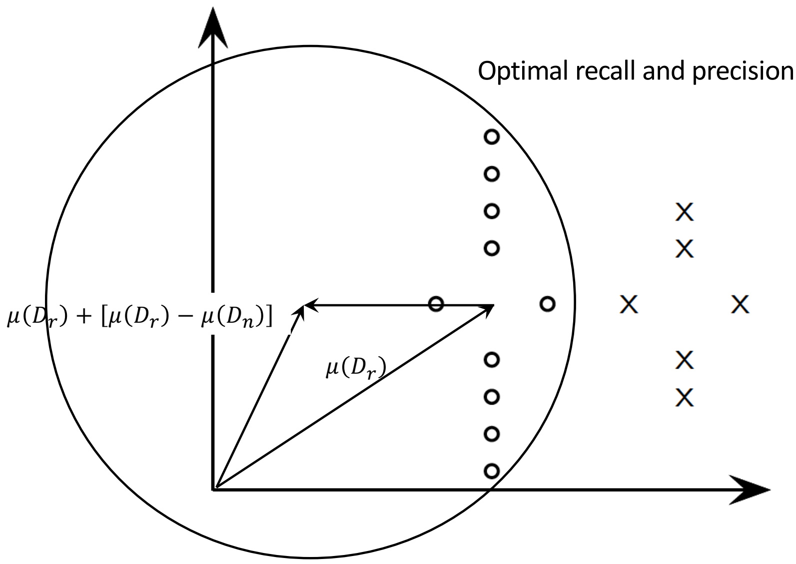
## Illustration of Rocchio Algorithm






We therefore consider also the centroid of the non-relevant documents as part of the user relevance feedback. We compute the difference vector among the two centroids, and we will use this difference vector to "move away" the query from the non-relevant documents.

## Illustration of Rocchio Algorithm








We add the difference vector to the centroid for the relevant documents. The resulting optimal query vector now can include all relevant documents in its result, without including non-relevant ones. In practice, such a clear separation will not always be achieved, but it has been shown that under some additional assumptions, this method is the optimal way to construct a query separating relevant from non-relevant documents.

## Identifying Relevant Documents

Following the previous reasoning the optimal query is

$$\vec{q}_{opt} = \mu(D_r) + [\mu(D_r) - \mu(D_n)]$$

Under cosine similarity

$$\vec{q}_{opt} = [\mu(D_r) - \mu(D_n)]$$

Practical issues

- User relevance feedback is not complete
- Users do not necessarily identify non-relevant documents
- Original query should continue to be considered




We derived in the previous illustration an optimal query vector, under a model that uses Euclidean distance as metrics. This approach is frequently used for illustration of how such a vector can be constructed. Since in the vector space model we use cosine similarity as similarity measure, the optimal vector under this metric is different. It is given as the difference vector of the two centroids.

Constructing an optimal query vector as described is only theoretically possible, since the complete information on relevant and non-relevant documents is missing in practice. Therefore, the theoretical considerations serve as a basis to devise a practical scheme, that is **approximating** the theoretical optimal vector by a vector that can be constructed from available data.

## SMART: Practical Relevance Feedback

Approximation scheme for the theoretically optimal query vector

If users identify some relevant documents $D_r$ from the result set R of a retrieval query q

- Assume all elements in $R \setminus D_r$ are not relevant, i.e., $D_n = R \setminus D_r$
- Modify the query to approximate theoretically optimal query

$$\vec{q}_{approx} = \alpha \vec{q} + \frac{\beta}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \frac{\gamma}{|R \setminus D_r|} \sum_{\vec{d}_j \notin D_r} \vec{d}_j$$

- $\alpha, \beta, \gamma$ are tuning parameters, $\alpha, \beta, \gamma \geq 0$
- Example: $\alpha = 1$, $\beta = 0.75$, $\gamma = 0.25$





The approximation scheme for user relevance feedback is called SMART. It assumes that users have identified some relevant documents. Then the scheme assumes that all other documents should be considered as non-relevant. This results in a modification of the original query that is controlled by 3 tuning parameters.

Since the assumption that all documents that have not been marked relevant are non-relevant is of course not correct, two mechanisms are used to moderate the impact of this wrong assumption:

1. The original query vector is maintained, in order not to drift away too dramatically from the original user query.
2. The weight given for the modification using the centroid of non-relevant documents is generally kept lower than the weight for the centroid of the relevant documents

# Example

Query q= "application theory"

Result

| |
|---|
| 0.77: B17 The Double Mellin-Barnes Type Integrals and Their Applications to Convolution Theory |
| 0.68: B3 Automatic Differentiation of Algorithms: Theory, Implementation, and Application |
| 0.23: B11 Oscillation Theory for Neutral Differential Equations with Delay |
| 0.23: B12 Oscillation Theory of Delay Differential Equations |

Query reformulation

$$\vec{q}_{approx} = \frac{1}{4}\vec{q} + \frac{1}{4}\vec{d_3} - \frac{1}{12}(\vec{d_{17}} + \vec{d_{12}} + \vec{d_{11}}), \alpha = \beta = \gamma = \frac{1}{4}$$

Result for reformulated query

| |
|---|
| 0.87: B3 Automatic Differentiation of Algorithms: Theory, Implementation, and Application |
| 0.61: B17 The Double Mellin-Barnes Type Integrals and Their Applications to Convolution Theory |
| 0.29: B7 Knapsack Problems: Algorithms and Computer Implementations |
| 0.23: B5 Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra |



This example shows how the query reformulation works. By identifying document B3 as being relevant and modifying the query vector it turns out that new documents (B5 and B7) become relevant. The reason is that those new documents share terms with document B3, and these terms are newly considered in the reformulated query.

## Discussion

Underlying assumptions of SMART algorithm

1. Original query contains sufficient number of relevant terms
2. Results contain new relevant terms that co-occur with original query terms
3. Relevant documents form a single cluster
4. Users are willing to provide feedback (!)

All assumptions can be violated in practice

Practical considerations

- Modified queries are complex → expensive processing
- Explicit relevance feedback consumes user time → could be used in other ways





Concerning the first assumption, if the initial query of the user does not contain sufficient information to retrieve a sufficiently large number of documents that are relevant to the true interest of the user (i.e., has sufficient recall), the relevance feedback system will not be able to discover additional relevant terms.

Concerning the second assumption, new terms can only be included as part of the modified query, if they co-occur at least in some documents together with original query terms. Otherwise, these terms could never be part of relevant documents in the result of the original query (why?).

Concerning the third assumption, implicitly the SMART algorithm assumes that all relevant documents are part of one cluster in the vector space. If they form multiple clusters, it is not able to correctly produce a query that can retrieve the relevant documents.

**Can documents which do not contain any keywords of the original query receive a positive similarity coefficient after relevance feedback?**

1. No
2. Yes, independent of the values β and γ
3. Yes, but only if β > 0
4. Yes, but only if γ > 0

## Which year Rocchio published his work on relevance feedback?

A. 1965

B. 1975

C. 1985

D. 1995

## Pseudo-Relevance Feedback

If users do not give feedback, automate the process

– Choose the top-k documents as the relevant ones
– Extend the query by selecting from the top-k documents the most relevant terms, according to some weighting scheme
– Alternatively: apply the SMART algorithm

Works often well

– But can fail horribly: query drift




The idea of relevance feedback has also been adopted for an automated extension of queries. Instead of the user selecting relevant documents, the system automatically chooses the top-k results as the set of relevant documents and then extends the query either by selecting some most relevant terms using a weighting scheme or applying the SMART algorithm.

This works well if the original query already separates well the topic of interest from other topics. If this is not the case, the method can fail catastrophically, driving the query towards a topic that is different from the originally intended one, where irrelevant query terms reinforce each other.

# Weighting Schemes

| Term ranking algorithm | Formula |
|---|---|
| Algorithms based on frequency heuristics | |
| total_freq | Score($t$) = $f_{R,t}$, where $f_{R,t}$ is the total frequency of term $t$ within the set of pseudo-relevant documents |
| IDF | $\text{Score}(t) = \log \frac{N}{n_t}$ |
| r_lohi [1] | Score($t$) = $r_t$ for ties, $n_t$ in ascending order, where $r_t$ is the number of pseudo-relevant documents containing term $t$ |
| Algorithm based on vector space model | |
| Rocchio | $\text{Score}(t) = \sum_{\forall d \in R} w_{d,t}$ |
| Algorithms based on distribution analysis | |
| F4MODIFIED [2,3] | $\text{Score}(t) = \log \frac{p_t}{1 - p_t} - \log \frac{q_t}{1 - q_t}$ |
| EMIM [1] | $\text{Score}(t) = \sum_{i \in \{t_k, \bar{t}_k\}, j \in \{R, \bar{R}\}} P(i,j) \log \frac{P(i,j)}{P(i)P(j)}$ |
| RSV [4] | $\text{Score}(t) = w_t (p_t - q_t)$ |
| KLD [5] | $\text{Score}(t) = p_t \cdot \log \frac{p_t}{c_t}$ |
| CHI2 [5] | $\text{Score}(t) = \frac{(p_t - c_t)^2}{c_t}$ |
| CHI1 [5] | $\text{Score}(t) = \frac{(p_t - c_t)}{}$ |

$p_t$: the probability of occurrence of term $t$ in the set of pseudo-relevant documents, $q_t$: the probability of occurrence of term $t$ in the set of non-relevant documents, $c_t$: the probability of occurrence of term $t$ in the whole document collection, $w_t$: the weight to be assigned to term $t$, EMIM: expected mutual information measure, F4: F4MODIFIED, IDF: inverse document frequency, KLD:



These are examples of weighting schemes that have been considered for pseudo-relevance feedback.

## 1.2.6.2 Global Query Expansion

Query is expanded using a global, *query-independent* resource

- Manually edited thesaurus
- Automatically extracted thesaurus
- Query logs





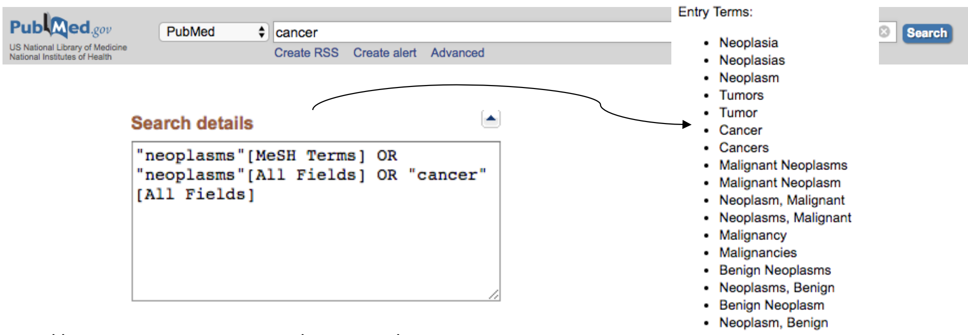
Global methods for expanding user queries can rely on a variety of resources. These may be thesauri that are manually constructed or automatically derived (a thesaurus is a database that contains words and their synonyms), or the automated analysis of query logs.

## Manually Created Thesaurus

Expensive to create and maintain

- Used mainly in science and engineering

Example: Pubmed




https://www.ncbi.nlm.nih.gov/pubmed/?term=cancer





Performing query expansion using a manually thesaurus requires the (expensive) effort of creating and maintaining such a thesaurus. This task is mainly performed in highly specialized technical fields in science and engineering. One prominent example of such a Thesaurus is maintained by Pubmed, the biggest publication database for medical literature maintained by the NIH, the National Institute of Health in the US. When using its search engine, you will find a window ”Search details” that shows how the user query is automatically expanded using the Pubmed thesaurus. In this example we see that the search system identifies that “cancer” is an entry on the concept “neoplasms”, and thus extends the query with all entries that it finds associated in the thesaurus (e.g., it would also search for “tumor”).

## Automatic Thesaurus Generation

Generate a thesaurus automatically by analyzing the distribution of words in documents

- Problem: find words with similar meaning (synonyms)

*Approach 1:* Two words are similar if they **co-occur** with similar words
"switzerland" ≈ "austria" because both occur with words such as "national", "election", "soccer" etc., so they must be similar.

*Approach 2:* Two words are similar if they occur in the same **text pattern**
"live in *", "travel to *", "size of *" are all phrases in which both "switzerland" or "austria" can occur




In order to avoid the effort of manually creating a thesaurus one can find methods to create it automatically by studying large numbers of documents and the distribution of words in those. This leads to the concept of word similarity. There exists two basic methods to study this similarity, either statistically, by observing which words occur together in documents, or in a more accurate way by identifying whether the words occur in the same text patterns.

We will study later in the lecture such methods in more detail. For the first approach, we will study so-called "word embeddings". For the second approach, we will learn about this type of methods in "information extraction".

## Expansion using Query Logs

Query logs are an important resource for query expansion with search engines

- Exploit correlations in user sessions

Example 1: users extend query

- After searching “Obama”, users search “Obama president”
- Therefore, ”president” might be a good expansion

Example 2: users refer to same result

- User A accesses URL epfl.ch after searching “Aebischer”
- User B accesses URL epfl.ch after searching “Vetterli”
- “Vetterli” might be a potential expansion for the query “Aebischer”




Query logs contain potentially rich information for query expansion. There are different ways of how such knowledge can be exploited. We show here two possibilities.

Other methods rely on mining query logs using various techniques, including clustering and association rule mining, that we will introduce later in the lecture in the part on “data mining”.

# References

Papers


- Rocchio, J. (1971). Relevance feedback in information retrieval. The Smart retrieval system-experiments in automatic document processing, 313-323.
- Ponte, Jay Michael, and W. Bruce Croft. "A language modeling approach to information retrieval." PhD diss., University of Massachusetts at Amherst, 1998.
- Yoo, S., & Choi, J. (2011). Evaluation of term ranking algorithms for pseudo-relevance feedback in MEDLINE retrieval. Healthcare informatics research, 17(2), 120-130.