Chapter 5

Relevance Feedback and Query Expansion

Introduction

- Most users find it difficult to formulate queries that are well designed for retrieval purposes
- Yet, most users often need to reformulate their queries to obtain the results of their interest
  - Thus, the first query formulation should be treated as an initial attempt to retrieve relevant information
  - Documents initially retrieved could be analyzed for relevance and used to improve initial query

Introduction

- The process of query modification is commonly referred as
  - **relevance feedback**, when the user provides information on relevant documents to a query, or
  - **query expansion**, when information related to the query is used to expand it
- We refer to both of them as feedback methods
- Two basic approaches of feedback methods:
  - **explicit feedback**, in which the information for query reformulation is provided directly by the users, and
  - **implicit feedback**, in which the information for query reformulation is implicitly derived by the system

A Framework for Feedback Methods

A Framework

- Consider a set of documents D_r that are known to be relevant to the current query q
- In relevance feedback, the documents in D_r are used to transform q into a modified query q_m
- However, obtaining information on documents relevant to a query requires the direct interference of the user
  - Most users are unwilling to provide this information, particularly in the Web

A Framework

- Because of this high cost, the idea of relevance feedback has been relaxed over the years
- Instead of asking the users for the relevant documents, we could:
  - Look at documents they have clicked on; or
  - Look at terms belonging to the top documents in the result set
- In both cases, it is expect that the feedback cycle will produce results of higher quality

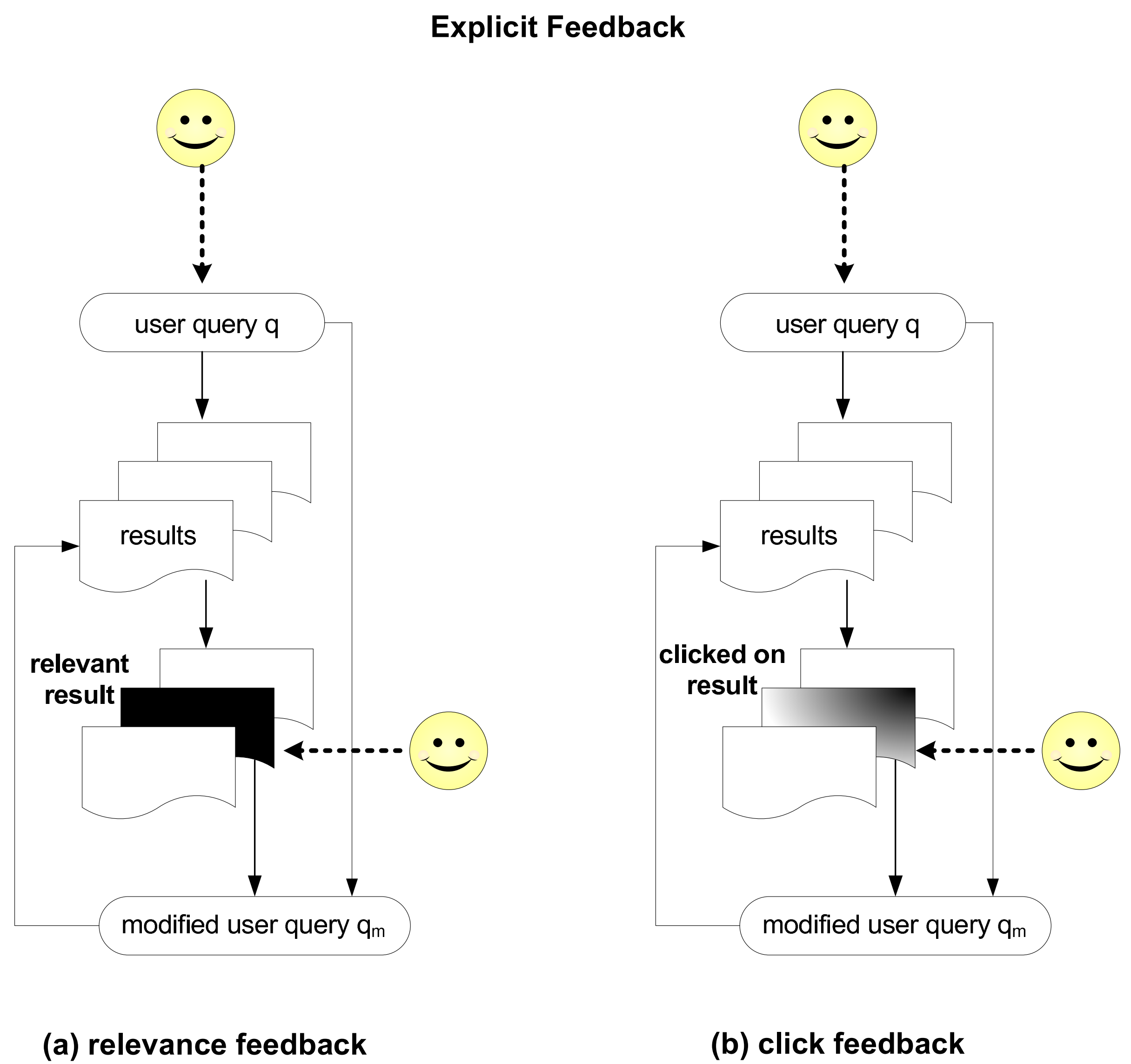
A Framework

- A **feedback cycle** is composed of two basic steps:
  - Determine feedback information that is either related or expected to be related to the original query q and
  - Determine how to transform query q to take this information effectively into account
- The first step can be accomplished in two distinct ways:
  - Obtain the feedback information explicitly from the users
  - Obtain the feedback information implicitly from the query results or from external sources such as a **thesaurus**

A Framework

- In an **explicit relevance feedback** cycle, the feedback information is provided directly by the users
- However, collecting feedback information is expensive and time consuming
- In the Web, **user clicks** on search results constitute a new source of feedback information
- A click indicate a document that is of interest to the user in the context of the current query
  - Notice that a click does not necessarily indicate a document that is relevant to the query

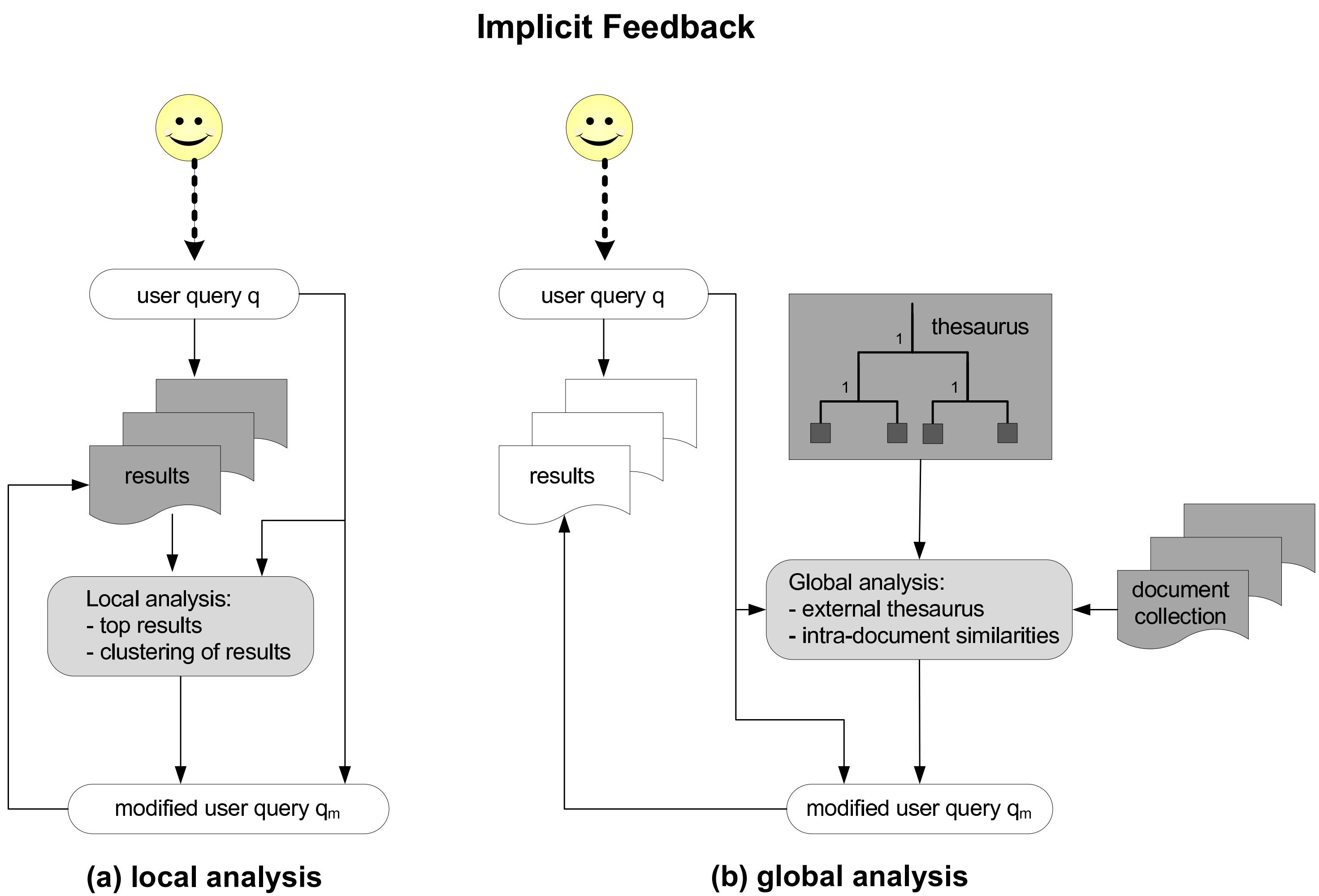
Explicit Feedback Information



A Framework

- In an **implicit relevance feedback** cycle, the feedback information is derived implicitly by the system
- There are two basic approaches for compiling implicit feedback information:
  - **local analysis**, which derives the feedback information from the top ranked documents in the result set
  - **global analysis**, which derives the feedback information from external sources such as a thesaurus

Implicit Feedback Information



Explicit Relevance Feedback

Explicit Relevance Feedback

- In a classic relevance feedback cycle, the user is presented with a list of the retrieved documents
- Then, the user examines them and marks those that are relevant
- In practice, only the top 10 (or 20) ranked documents need to be examined
- The main idea consists of
  - selecting important terms from the documents that have been identified as relevant, and
  - enhancing the importance of these terms in a new query formulation

Explicit Relevance Feedback

- **Expected effect**: the new query will be moved towards the relevant docs and away from the non-relevant ones
- Early experiments have shown good improvements in precision for small test collections
- Relevance feedback presents the following characteristics:
  - it shields the user from the details of the query reformulation process (all the user has to provide is a relevance judgement)
  - it breaks down the whole searching task into a sequence of small steps which are easier to grasp

The Rocchio Method

The Rocchio Method

- Documents identified as relevant (to a given query) have similarities among themselves
- Further, non-relevant docs have term-weight vectors which are dissimilar from the relevant documents
- The basic idea of the Rocchio Method is to reformulate the query such that it gets:
  - closer to the neighborhood of the relevant documents in the vector space, and
  - away from the neighborhood of the non-relevant documents

The Rocchio Method

- Let us define terminology regarding the processing of a given query q, as follows:
  - D_r: set of *relevant* documents among the documents retrieved
  - N_r: number of documents in set D_r
  - D_n: set of *non-relevant* docs among the documents retrieved
  - N_n: number of documents in set D_n
  - C_r: set of relevant docs among all documents in the collection
  - N: number of documents in the collection
  - α, β, γ: tuning constants

The Rocchio Method

- Consider that the set C_r is known in advance
- Then, the best query vector for distinguishing the relevant from the non-relevant docs is given by

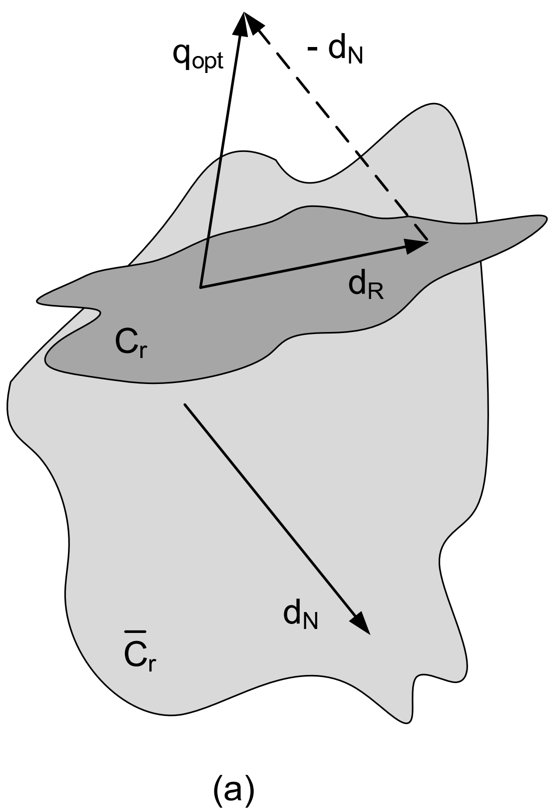
$$\vec{q}_{opt} = \frac{1}{|C_r|} \sum_{\forall \vec{d}_j \in C_r} \vec{d}_j - \frac{1}{N - |C_r|} \sum_{\forall \vec{d}_j \notin C_r} \vec{d}_j$$

where

- $|C_r|$ refers to the cardinality of the set C_r
- $\vec{d}_j$ is a weighted term vector associated with document d_j, and
- $\vec{q}_{opt}$ is the optimal weighted term vector for query q

The Rocchio Method

- However, the set C_r is not known a priori
- To solve this problem, we can formulate an initial query and to incrementally change the initial query vector



The Rocchio Method

- There are three classic and similar ways to calculate the modified query $\vec{q}_m$ as follows,

$$
\begin{aligned}
Standard\_Rocchio: \quad \vec{q}_m &= \alpha\,\vec{q} + \frac{\beta}{N_r}\sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \frac{\gamma}{N_n}\sum_{\forall \vec{d}_j \in D_n} \vec{d}_j \\
Ide\_Regular: \quad \vec{q}_m &= \alpha\,\vec{q} + \beta\sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \gamma\sum_{\forall \vec{d}_j \in D_n} \vec{d}_j \\
Ide\_Dec\_Hi: \quad \vec{q}_m &= \alpha\,\vec{q} + \beta\sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \gamma\; max\_rank(D_n)
\end{aligned}
$$

where $max\_rank(D_n)$ is the highest ranked non-relevant doc

The Rocchio Method

- Three different setups of the parameters in the Rocchio formula are as follows:
  - $\alpha = 1$, proposed by Rocchio
  - $\alpha = \beta = \gamma = 1$, proposed by Ide
  - $\gamma = 0$, which yields a *positive* feedback strategy
  - The current understanding is that the three techniques yield similar results
- The main advantages of the above relevance feedback techniques are simplicity and good results
  - **Simplicity**: modified term weights are computed directly from the set of retrieved documents
  - **Good results**: the modified query vector does reflect a portion of the intended query semantics (observed experimentally)

Relevance Feedback for the Probabilistic Model

A Probabilistic Method

- The probabilistic model ranks documents for a query q according to the probabilistic ranking principle
- The similarity of a document d_j to a query q in the probabilistic model can be expressed as

$$sim(d_j, q) \; \alpha \sum_{k_i \in q \wedge k_i \in d_j} \left(\log \frac{P(k_i|R)}{1 - P(k_i|R)} + \log \frac{1 - P(k_i|\overline{R})}{P(k_i|\overline{R})} \right)$$

  where

  - $P(k_i|R)$ stands for the probability of observing the term k_i in the set R of relevant documents
  - $P(k_i|\overline{R})$ stands for the probability of observing the term k_i in the set $\overline{R}$ of non-relevant docs

A Probabilistic Method

- Initially, the equation above cannot be used because $P(k_i|R)$ and $P(k_i|\overline{R})$ are unknown
- Different methods for estimating these probabilities automatically were discussed in Chapter 3
- With user feedback information, these probabilities are estimated in a slightly different way
- For the initial search (when there are no retrieved documents yet), assumptions often made include:
  - $P(k_i|R)$ is constant for all terms k_i (typically 0.5)
  - the term probability distribution $P(k_i|\overline{R})$ can be approximated by the distribution in the whole collection

A Probabilistic Method

- These two assumptions yield:

$$P(k_i|R) = 0.5 \qquad P(k_i|\overline{R}) = \frac{n_i}{N}$$

  where n_i stands for the number of documents in the collection that contain the term k_i

- Substituting into similarity equation, we obtain

$$sim_{initial}(d_j, q) = \sum_{k_i \in q \wedge k_i \in d_j} \log \frac{N - n_i}{n_i}$$

- For the feedback searches, the accumulated statistics on relevance are used to evaluate $P(k_i|R)$ and $P(k_i|\overline{R})$

A Probabilistic Method

- Let $n_{r,i}$ be the number of documents in set D_r that contain the term k_i
- Then, the probabilities $P(k_i|R)$ and $P(k_i|\overline{R})$ can be approximated by

$$P(k_i|R) = \frac{n_{r,i}}{N_r} \qquad P(k_i|\overline{R}) = \frac{n_i - n_{r,i}}{N - N_r}$$

- Using these approximations, the similarity equation can rewritten as

$$sim(d_j, q) = \sum_{k_i \in q \wedge k_i \in d_j} \left(\log \frac{n_{r,i}}{N_r - n_{r,i}} + \log \frac{N - N_r - (n_i - n_{r,i})}{n_i - n_{r,i}} \right)$$

A Probabilistic Method

- Notice that here, contrary to the Rocchio Method, no query expansion occurs
- The same query terms are reweighted using feedback information provided by the user
- The formula above poses problems for certain small values of N_r and $n_{r,i}$
- For this reason, a 0.5 adjustment factor is often added to the estimation of $P(k_i|R)$ and $P(k_i|\overline{R})$:

$$P(k_i|R) = \frac{n_{r,i} + 0.5}{N_r + 1} \qquad P(k_i|\overline{R}) = \frac{n_i - n_{r,i} + 0.5}{N - N_r + 1}$$

A Probabilistic Method

- The main advantage of this feedback method is the derivation of new weights for the query terms
- The disadvantages include:
  - document term weights are **not** taken into account during the feedback loop;
  - weights of terms in the previous query formulations are disregarded; and
  - no query expansion is used (the same set of index terms in the original query is reweighted over and over again)
- Thus, this method does **not** in general operate as effectively as the vector modification methods

Evaluation of Relevance Feedback

Evaluation of Relevance Feedback

- Consider the modified query vector $\vec{q}_m$ produced by expanding $\vec{q}$ with relevant documents, according to the Rocchio formula
- Evaluation of $\vec{q}_m$:
  - Compare the documents retrieved by $\vec{q}_m$ with the set of relevant documents for $\vec{q}$
  - In general, the results show spectacular improvements
  - However, a part of this improvement results from the higher ranks assigned to the relevant docs used to expand $\vec{q}$ into $\vec{q_m}$
  - Since the user has seen these docs already, such evaluation is unrealistic

The Residual Collection

- A more realistic approach is to evaluate $\vec{q}_m$ considering only the **residual collection**
  - We call residual collection the set of all docs minus the set of feedback docs provided by the user
- Then, the recall-precision figures for $\vec{q}_m$ tend to be lower than the figures for the original query vector $\vec{q}$
- This is not a limitation because the main purpose of the process is to compare distinct relevance feedback strategies

Explicit Feedback Through Clicks

Explicit Feedback Through Clicks

- Web search engine users not only inspect the answers to their queries, they also click on them
- The clicks reflect preferences for particular answers in the context of a given query
- They can be collected in large numbers without interfering with the user actions
- The immediate question is whether they also reflect relevance judgements on the answers
- Under certain restrictions, the answer is affirmative as we now discuss

Eye Tracking

- Clickthrough data provides limited information on the user behavior
- One approach to complement information on user behavior is to use **eye tracking devices**
- Such commercially available devices can be used to determine the area of the screen the user is focussed in
- The approach allows correctly detecting the area of the screen of interest to the user in 60-90% of the cases
- Further, the cases for which the method does not work can be determined

Eye Tracking

- Eye movements can be classified in four types: fixations, saccades, pupil dilation, and scan paths
- **Fixations** are a gaze at a particular area of the screen lasting for 200-300 milliseconds
- This time interval is large enough to allow effective brain capture and interpretation of the image displayed
- Fixations are the ocular activity normally associated with visual information acquisition and processing
- That is, fixations are key to interpreting user behavior

Relevance Judgements

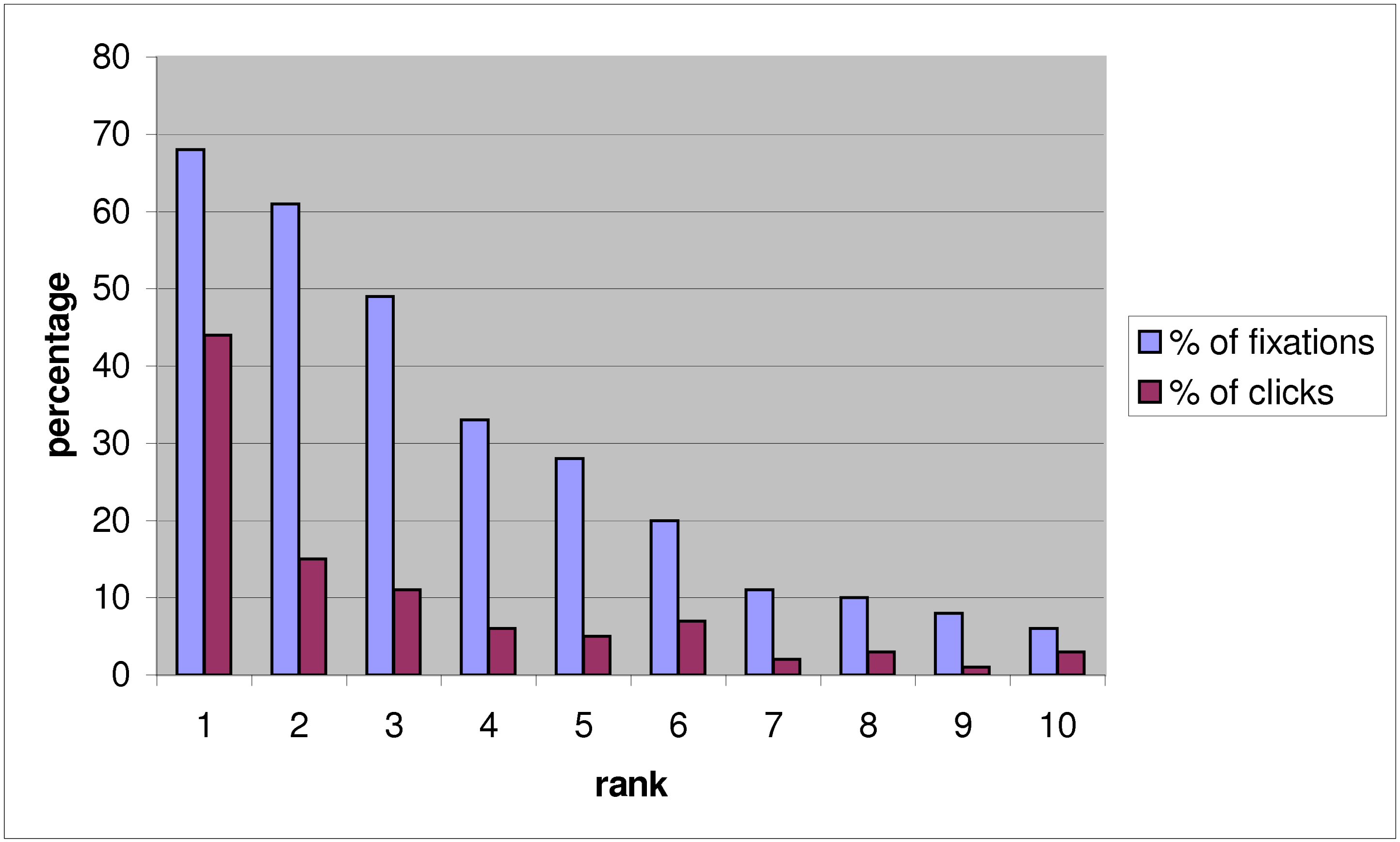
- To evaluate the **quality** of the results, eye tracking is not appropriate
- This evaluation requires selecting a set of test queries and determining relevance judgements for them
- This is also the case if we intend to evaluate the quality of the signal produced by clicks

User Behavior

- Eye tracking experiments have shown that users scan the query results from top to bottom
- The users inspect the first and second results right away, within the second or third fixation
- Further, they tend to scan the top 5 or top 6 answers thoroughly, before scrolling down to see other answers

User Behavior

- Percentage of times each one of the top results was viewed and clicked on by a user, for 10 test tasks and 29 subjects (Thorsten Joachims *et al*)



User Behavior

- We notice that the users inspect the top 2 answers almost equally, but they click three times more in the first
- This might be indicative of a user bias towards the search engine
  - That is, that the users tend to trust the search engine in recommending a top result that is relevant

User Behavior

- This can be better understood by presenting test subjects with two distinct result sets:
  - the normal ranking returned by the search engine and
  - a modified ranking in which the top 2 results have their positions swapped
- Analysis suggest that the user displays a *trust bias* in the search engine that favors the top result
- That is, the position of the result has great influence on the user's decision to click on it

Clicks as a Metric of Preferences

- Thus, it is clear that interpreting clicks as a direct indicative of relevance is not the best approach
- More promising is to interpret clicks as a metric of **user preferences**
- For instance, a user can look at a result and decide to skip it to click on a result that appears lower
- In this case, we say that the user prefers the result clicked on to the result shown upper in the ranking
- This type of preference relation takes into account:
  - the results clicked on by the user
  - the results that were inspected and not clicked on

Clicks within a Same Query

- To interpret clicks as user preferences, we adopt the following definitions
  - Given a ranking function $\mathcal{R}(q_i, d_j)$, let r_k be the kth ranked result
  - That is, r_1, r_2, r_3 stand for the first, the second, and the third top results, respectively
  - Further, let $\surd r_k$ indicate that the user has clicked on the k_{th} result
  - Define a preference function $r_k > r_{k-n}$, $0 < k - n < k$, that states that, according to the click actions of the user, the kth top result is preferrable to the $(k - n)$th result

Clicks within a Same Query

- To illustrate, consider the following example regarding the click behavior of a user:

$$r_1 \quad r_2 \quad \surd r_3 \quad r_4 \quad \surd r_5 \quad r_6 \quad r_7 \quad r_8 \quad r_9 \quad \surd r_{10}$$

- This behavior does not allow us to make definitive statements about the relevance of results r_3, r_5, and r_{10}
- However, it does allow us to make statements on the relative preferences of this user
- Two distinct strategies to capture the preference relations in this case are as follows.
  - Skip-Above: if $\surd r_k$ then $r_k > r_{k-n}$, for all r_{k-n} that was not clicked
  - Skip-Previous: if $\surd r_k$ and r_{k-1} has not been clicked then $r_k > r_{k-1}$

Clicks within a Same Query

- To illustrate, consider again the following example regarding the click behavior of a user:

$$r_1 \quad r_2 \quad \surd r_3 \quad r_4 \quad \surd r_5 \quad r_6 \quad r_7 \quad r_8 \quad r_9 \quad \surd r_{10}$$

- According to the Skip-Above strategy, we have:

$$r_3 > r_2; \qquad r_3 > r_1$$

- And, according to the Skip-Previous strategy, we have:

$$r_3 > r_2$$

- We notice that the Skip-Above strategy produces more preference relations than the Skip-Previous strategy

Clicks within a Same Query

- Empirical results indicate that user clicks are in agreement with judgements on the relevance of results in roughly 80% of the cases
  - Both the Skip-Above and the Skip-Previous strategies produce preference relations
  - If we swap the first and second results, the clicks still reflect preference relations, for both strategies
  - If we reverse the order of the top 10 results, the clicks still reflect preference relations, for both strategies
- Thus, the clicks of the users can be used as a strong indicative of personal preferences
- Further, they also can be used as a strong indicative of the relative relevance of the results for a given query

Clicks within a Query Chain

- The discussion above was restricted to the context of a single query
- However, in practice, users issue more than one query in their search for answers to a same task
- The set of queries associated with a same task can be identified in **live query streams**
  - This set constitute what is referred to as a **query chain**
- The purpose of analysing query chains is to produce new preference relations

Clicks within a Query Chain

- To illustrate, consider that two result sets in a same query chain led to the following click actions:

$$\begin{array}{llllllllll} r_1 & r_2 & r_3 & r_4 & r_5 & r_6 & r_7 & r_8 & r_9 & r_{10} \\ s_1 & \surd s_2 & s_3 & s_4 & \surd s_5 & s_6 & s_7 & s_8 & s_9 & s_{10} \end{array}$$

  where

  - r_j refers to an answer in the first result set
  - s_j refers to an answer in the second result set

- In this case, the user only clicked on the second and fifth answers of the second result set

Clicks within a Query Chain

- Two distinct strategies to capture the preference relations in this case, are as follows
  - Top-One-No-Click-Earlier: if $\exists\ s_k \mid \surd s_k$ then $s_j > r_1$, for $j \leq 10$.
  - Top-Two-No-Click-Earlier: if $\exists\ s_k \mid \surd s_k$ then $s_j > r_1$ and $s_j > r_2$, for $j \leq 10$
- According the first strategy, the following preferences are produced by the click of the user on result s_2:

$$s_1 > r_1; \quad s_2 > r_1; \quad s_3 > r_1; \quad s_4 > r_1; \quad s_5 > r_1; \quad \ldots$$

- According the second strategy, we have:

$$\begin{array}{llllll} s_1 > r_1; & s_2 > r_1; & s_3 > r_1; & s_4 > r_1; & s_5 > r_1; & \ldots \\ s_1 > r_2; & s_2 > r_2; & s_3 > r_2; & s_4 > r_2; & s_5 > r_2; & \ldots \end{array}$$

Clicks within a Query Chain

- We notice that the second strategy produces twice more preference relations than the first
- These preference relations must be compared with the relevance judgements of the human assessors
- The following conclusions were derived:
  - Both strategies produce preference relations in agreement with the relevance judgements in roughly 80% of the cases
  - Similar agreements are observed even if we swap the first and second results
  - Similar agreements are observed even if we reverse the order of the results

Clicks within a Query Chain

- These results suggest:
  - The users provide negative feedback on whole result sets (by not clicking on them)
  - The users learn with the process and reformulate better queries on the subsequent iterations

Click-based Ranking

Click-based Ranking

- Click through information can be used to improve the ranking
- This can be done by **learning** a modified ranking function from click-based preferences
- One approach is to use support vector machines (SVMs) to learn the ranking function

Click-based Ranking

- In this case, preference relations are transformed into inequalities among weighted term vectors representing the ranked documents
- These inequalities are then translated into an SVM optimization problem
- The solution of this optimization problem is the optimal weights for the document terms
- The approach proposes the combination of different retrieval functions with different weights

Implicit Feedback Through Local Analysis

Local analysis

- Local analysis consists in deriving feedback information from the documents retrieved for a given query q
- This is similar to a relevance feedback cycle but done without assistance from the user
- Two local strategies are discussed here: **local clustering** and local **context analysis**

Local Clustering

Local Clustering

- Adoption of clustering techniques for query expansion has been a basic approach in information retrieval
- The standard procedure is to quantify term correlations and then use the correlated terms for query expansion
- Term correlations can be quantified by using global structures, such as association matrices
- However, global structures might not adapt well to the local context defined by the current query
- To deal with this problem, **local clustering** can be used, as we now discuss

Association Clusters

- For a given query q, let
  - D_ℓ: **local document set**, i.e., set of documents retrieved by q
  - N_ℓ: number of documents in D_l
  - V_l: local vocabulary, i.e., set of all distinct words in D_l
  - $f_{i,j}$: frequency of occurrence of a term k_i in a document $d_j \in D_l$
  - $\mathbf{M}_\ell$=$[m_{ij}]$: term-document matrix with V_l rows and N_l columns
  - m_{ij}=$f_{i,j}$: an element of matrix $\mathbf{M}_\ell$
  - $\mathbf{M}_\ell^T$: transpose of $\mathbf{M}_\ell$
- The matrix

$$\mathbf{C}_\ell = \mathbf{M}_\ell \mathbf{M}_\ell^T$$

is a local term-term correlation matrix

Association Clusters

- Each element $c_{u,v} \in \mathbf{C}_\ell$ expresses a correlation between terms k_u and k_v
- This relationship between the terms is based on their joint co-occurrences inside documents of the collection
- Higher the number of documents in which the two terms co-occur, stronger is this correlation
- Correlation strengths can be used to define local clusters of neighbor terms
- Terms in a same cluster can then be used for query expansion
- We consider three types of clusters here: **association clusters**, **metric clusters**, and **scalar clusters**.

Association Clusters

- An association cluster is computed from a local correlation matrix $\mathbf{C}_\ell$
- For that, we re-define the correlation factors $c_{u,v}$ between any pair of terms k_u and k_v, as follows:

$$c_{u,v} = \sum_{d_j \in D_l} f_{u,j} \times f_{v,j}$$

- In this case the correlation matrix is referred to as a **local association matrix**
- The motivation is that terms that co-occur frequently inside documents have a synonymity association

Association Clusters

- The correlation factors $c_{u,v}$ and the association matrix $\mathbf{C}_\ell$ are said to be unnormalized
- An alternative is to normalize the correlation factors:

$$c'_{u,v} = \frac{c_{u,v}}{c_{u,u} + c_{v,v} - c_{u,v}}$$

- In this case the association matrix $\mathbf{C}_\ell$ is said to be normalized

Association Clusters

- Given a local association matrix $\mathbf{C}_\ell$, we can use it to build local association clusters as follows
- Let $C_u(n)$ be a function that returns the n largest factors $c_{u,v} \in \mathbf{C}_\ell$, where v varies over the set of local terms and $v \neq u$
- Then, $C_u(n)$ defines a local association cluster, a neighborhood, around the term k_u
- Given a query q, we are normally interested in finding clusters only for the $|q|$ query terms
- This means that such clusters can be computed efficiently at query time

Metric Clusters

- Association clusters do not take into account where the terms occur in a document
- However, two terms that occur in a same sentence tend to be more correlated
- A **metric cluster** re-defines the correlation factors $c_{u,v}$ as a function of their distances in documents

Metric Clusters

- Let $k_u(n, j)$ be a function that returns the nth occurrence of term k_u in document d_j
- Further, let $r(k_u(n, j), k_v(m, j))$ be a function that computes the distance between
  - the nth occurrence of term k_u in document d_j; and
  - the mth occurrence of term k_v in document d_j
- We define,

$$c_{u,v} = \sum_{d_j \in D_l} \sum_n \sum_m \frac{1}{r(k_u(n, j), k_v(m, j))}$$

- In this case the correlation matrix is referred to as a local **metric matrix**

Metric Clusters

- Notice that if k_u and k_v are in distinct documents we take their distance to be infinity
- Variations of the above expression for $c_{u,v}$ have been reported in the literature, such as $1/r^2(k_u(n,j), k_v(m,j))$
- The metric correlation factor $c_{u,v}$ quantifies absolute inverse distances and is said to be unnormalized
- Thus, the local metric matrix $\mathbf{C}_\ell$ is said to be unnormalized

Metric Clusters

- An alternative is to normalize the correlation factor
- For instance,

$$c'_{u,v} = \frac{c_{u,v}}{total\ number\ of\ [k_u, k_v]\ pairs\ considered}$$

- In this case the local metric matrix C_ℓ is said to be normalized

Scalar Clusters

- The correlation between two local terms can also be defined by comparing the neighborhoods of the two terms
- The idea is that two terms with similar *neighborhoods* have some synonymity relationship
  - In this case we say that the relationship is indirect or induced by the neighborhood
  - We can quantify this relationship comparing the neighborhoods of the terms through a scalar measure
  - For instance, the cosine of the angle between the two vectors is a popular scalar similarity measure

Scalar Clusters

- Let
  - $\vec{s}_u = (c_{u,x_1}, s_{u,x_2}, \ldots, s_{u,x_n})$: vector of neighborhood correlation values for the term k_u
  - $\vec{s}_v = (c_{v,y_1}, c_{v,y_2}, \ldots, c_{v,y_m})$: vector of neighborhood correlation values for term k_v
- Define,

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

- In this case the correlation matrix C_ℓ is referred to as a **local scalar matrix**

Scalar Clusters

- The local scalar matrix $\mathbf{C}_\ell$ is said to be induced by the neighborhood
- Let $C_u(n)$ be a function that returns the n largest $c_{u,v}$ values in a local scalar matrix $\mathbf{C}_\ell$, $v \neq u$
- Then, $C_u(n)$ defines a scalar cluster around term k_u

Neighbor Terms

- Terms that belong to clusters associated to the query terms can be used to expand the original query
- Such terms are called neighbors of the query terms and are characterized as follows
- A term k_v that belongs to a cluster $C_u(n)$, associated with another term k_u, is said to be a **neighbor** of k_u
- Often, neighbor terms represent distinct keywords that are correlated by the current query context

Neighbor Terms

- Consider the problem of expanding a given user query q with neighbor terms
- One possibility is to expand the query as follows
- For each term $k_u \in q$, select m neighbor terms from the cluster $C_u(n)$ and add them to the query
- This can be expressed as follows:

$$q_m = q \cup \{k_v | k_v \in C_u(n), k_u \in q\}$$

- Hopefully, the additional neighbor terms k_v will retrieve new relevant documents

Neighbor Terms

- The set $C_u(n)$ might be composed of terms obtained using correlation factors normalized and unnormalized
- Query expansion is important because it tends to improve recall
- However, the larger number of documents to rank also tends to lower precision
- Thus, query expansion needs to be exercised with great care and fine tuned for the collection at hand

Local Context Analysis

Local Context Analysis

- The local clustering techniques are based on the set of documents retrieved for a query
- A distinct approach is to search for term correlations in the whole collection
- Global techniques usually involve the building of a thesaurus that encodes term relationships in the whole collection
- The terms are treated as concepts and the thesaurus is viewed as a concept relationship structure
- The building of a thesaurus usually considers the use of small contexts and phrase structures

Local Context Analysis

- Local context analysis is an approach that combines global and local analysis
- It is based on the use of **noun groups**, i.e., a single noun, two nouns, or three adjacent nouns in the text
- Noun groups selected from the top ranked documents are treated as document concepts
- However, instead of documents, passages are used for determining term co-occurrences
    - Passages are text windows of fixed size

Local Context Analysis

- More specifically, the local context analysis procedure operates in three steps
  - First, retrieve the top n ranked passages using the original query
  - Second, for each concept c in the passages compute the similarity $sim(q, c)$ between the whole query q and the concept c
  - Third, the top m ranked concepts, according to $sim(q, c)$, are added to the original query q
- A weight computed as $\boxed{1 - 0.9 \times i/m}$ is assigned to each concept c, where
  - i: position of c in the concept ranking
  - m: number of concepts to add to q
- The terms in the original query q might be stressed by assigning a weight equal to 2 to each of them

Local Context Analysis

- Of these three steps, the second one is the most complex and the one which we now discuss
- The similarity $sim(q, c)$ between each concept c and the original query q is computed as follows

$$\boxed{sim(q, c) = \prod_{k_i \in q} \left(\delta + \frac{\log(f(c,k_i) \times idf_c)}{\log n} \right)^{idf_i}}$$

where n is the number of top ranked passages considered

Local Context Analysis

- The function $f(c, k_i)$ quantifies the correlation between the concept c and the query term k_i and is given by

$$f(c, k_i) = \sum_{j=1}^{n} pf_{i,j} \times pf_{c,j}$$

  where

  - $pf_{i,j}$ is the frequency of term k_i in the j-th passage; and
  - $pf_{c,j}$ is the frequency of the concept c in the j-th passage

- Notice that this is the correlation measure defined for association clusters, but adapted for passages

Local Context Analysis

- The inverse document frequency factors are computed as

$$
\begin{aligned}
idf_i &= max(1, \frac{\log_{10} N/np_i}{5}) \\
idf_c &= max(1, \frac{\log_{10} N/np_c}{5})
\end{aligned}
$$

  where

  - N is the number of passages in the collection
  - np_i is the number of passages containing the term k_i; and
  - np_c is the number of passages containing the concept c

- The idf_i factor in the exponent is introduced to emphasize infrequent query terms

Local Context Analysis

- The procedure above for computing $sim(q, c)$ is a non-trivial variant of tf-idf ranking
- It has been adjusted for operation with TREC data and did not work so well with a different collection
- Thus, it is important to have in mind that tuning might be required for operation with a different collection

Implicit Feedback Through Global Analysis

Global Context Analysis

- The methods of local analysis extract information from the local set of documents retrieved to expand the query
- An alternative approach is to expand the query using information from the whole set of documents—a strategy usually referred to as **global analysis procedures**
- We distinguish two global analysis procedures:
  - Query expansion based on a similarity thesaurus
  - Query expansion based on a statistical thesaurus

Query Expansion based on a Similarity Thesaurus

Similarity Thesaurus

- We now discuss a query expansion model based on a global **similarity thesaurus** constructed automatically
- The similarity thesaurus is based on term to term relationships rather than on a matrix of co-occurrence
- Special attention is paid to the selection of terms for expansion and to the reweighting of these terms
- Terms for expansion are selected based on their similarity to the whole query

Similarity Thesaurus

- A similarity thesaurus is built using term to term relationships
- These relationships are derived 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
- Thus, terms assume the original role of documents while documents are interpreted as indexing elements

Similarity Thesaurus

- Let,
  - t: number of terms in the collection
  - N: number of documents in the collection
  - $f_{i,j}$: frequency of term k_i in document d_j
  - t_j: number of distinct index terms in document d_j
- Then,

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

is the inverse term frequency for document d_j (analogous to inverse document frequency)

Similarity Thesaurus

- Within this framework, with each term k_i is associated a vector $\vec{k}_i$ given by

$$\vec{k}_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,N})$$

- These weights are computed as follows

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

where $max_j(f_{i,j})$ computes the maximum of all $f_{i,j}$ factors for the i-th term

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 given by the scalar term-term matrix composed of correlation factors $c_{u,v}$
- This global similarity thesaurus has to be computed only once and can be updated incrementally

Similarity Thesaurus

- Given the global similarity thesaurus, query expansion is done in three steps as follows
  - First, represent the query in the same vector space used for representing the index terms
  - Second, compute a similarity $sim(q, k_v)$ between each term k_v correlated to the query terms and the whole query q
  - Third, expand the query with the top r ranked terms according to $sim(q, k_v)$

Similarity Thesaurus

- For the first step, the query is represented by a vector $\vec{q}$ given by

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

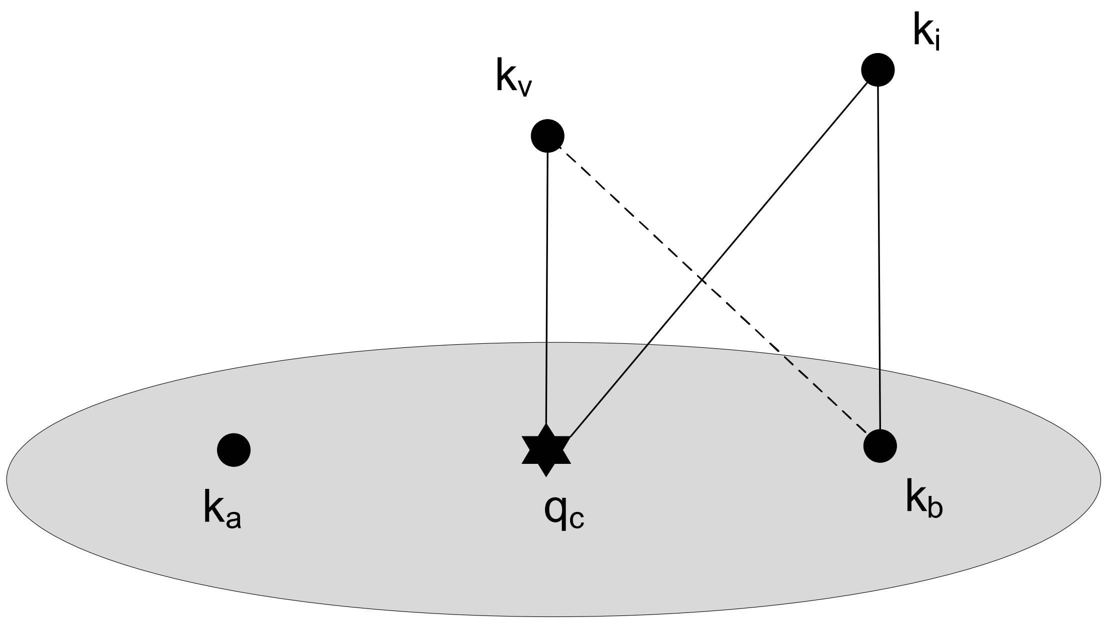
  where $w_{i,q}$ is a term-query weight computed using the equation for $w_{i,j}$, but with $\vec{q}$ in place of $\vec{d_j}$

- For the second step, the similarity $sim(q, k_v)$ is computed as

$$sim(q, k_v) = \vec{q} \cdot \vec{k_v} = \sum_{k_i \in q} w_{i,q} \times c_{i,v}$$

Similarity Thesaurus

- A term k_v might be closer to the whole query centroid q_C than to the individual query terms



- Thus, terms selected here might be distinct from those selected by previous global analysis methods

Similarity Thesaurus

- For the third step, the top r ranked terms are added to the query q to form the expanded query q_m
- To each expansion term k_v in query q_m is assigned a weight w_{v,q_m} given by

$$w_{v,q_m} = \frac{sim(q, k_v)}{\sum_{k_i \in q} w_{i,q}}$$

- The expanded query q_m is then used to retrieve new documents
- This technique has yielded improved retrieval performance (in the range of 20%) with three different collections

Similarity Thesaurus

- Consider a document d_j which is represented in the term vector space by $\vec{d_j} = \sum_{k_i \in d_j} w_{i,j} \vec{k_i}$
- Assume that the query q is expanded to include all the t index terms (properly weighted) in the collection
- Then, the similarity $sim(q, d_j)$ between d_j and q can be computed in the term vector space by

$$\boxed{sim(q, d_j) \;\; \alpha \sum_{k_v \in d_j} \sum_{k_u \in q} w_{v,j} \times w_{u,q} \times c_{u,v}}$$

Similarity Thesaurus

- The previous expression is analogous to the similarity formula in the generalized vector space model
- Thus, the generalized vector space model can be interpreted as a query expansion technique
- The two main differences are
  - the weights are computed differently
  - only the top r ranked terms are used

Query Expansion based on a Statistical Thesaurus

Global Statistical Thesaurus

- We now discuss a query expansion technique based on a **global statistical thesaurus**
- The approach is quite distinct from the one based on a similarity thesaurus
- The global thesaurus is composed of classes that group correlated terms in the context of the whole collection
- Such correlated terms can then be used to expand the original user query

Global Statistical Thesaurus

- To be effective, the terms selected for expansion must have high term discrimination values
  - This implies that they must be low frequency terms
- However, it is difficult to cluster low frequency terms due to the small amount of information about them
- To circumvent this problem, documents are clustered into classes
- The low frequency terms in these documents are then used to define thesaurus classes

Global Statistical Thesaurus

- A document clustering algorithm that produces small and tight clusters is the **complete link algorithm**:

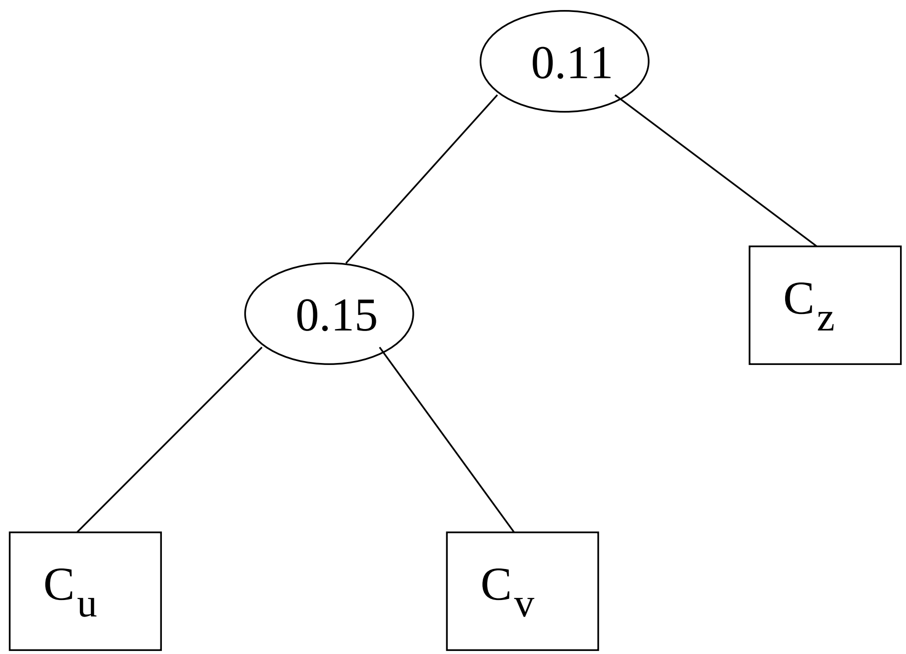
1. Initially, place each document in a distinct cluster
2. Compute the similarity between all pairs of clusters
3. Determine the pair of clusters $[C_u, C_v]$ with the highest inter-cluster similarity
4. Merge the clusters C_u and C_v
5. Verify a stop criterion (if this criterion is not met then go back to step 2)
6. Return a hierarchy of clusters

Global Statistical Thesaurus

- The similarity between two clusters is defined as the minimum of the similarities between two documents not in the same cluster
- To compute the similarity between documents in a pair, the cosine formula of the vector model is used
- As a result of this minimality criterion, the resultant clusters tend to be small and tight

Global Statistical Thesaurus

- Consider that the whole document collection has been clustered using the complete link algorithm
- Figure below illustrates a portion of the whole cluster hierarchy generated by the complete link algorithm



where the inter-cluster similarities are shown in the ovals

Global Statistical Thesaurus

- 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 a class
  - MIDF: minimum inverse document frequency
- Paramenter TC determines the document clusters that will be used to generate thesaurus classes
  - Two clusters C_u and C_v are selected, when TC is surpassed by $sim(C_u, C_v)$

Global Statistical Thesaurus

- Use NDC as a limit on the number of documents of the clusters
  - For instance, if both C_{u+v} and C_{u+v+z} are selected then the parameter NDC might be used to decide between the two
- MIDF defines the minimum value of IDF for any term which is selected to participate in a thesaurus class

Global Statistical Thesaurus

- Given that the thesaurus classes have been built, they can be used for query expansion
- For this, an average term weight wt_C for each thesaurus class C is computed as follows

$$wt_C = \frac{\sum_{i=1}^{|C|} w_{i,C}}{|C|}$$

where

  - $|C|$ is the number of terms in the thesaurus class C, and
  - $w_{i,C}$ is a weight associated with the term-class pair $[k_i, C]$

Global Statistical Thesaurus

- This average term weight can then be used to compute a thesaurus class weight w_C as

$$w_C = \frac{wt_C}{|C|} \times 0.5$$

- The above weight formulations have been verified through experimentation and have yielded good results