Modern Information Retrieval

Chapter 5

Relevance Feedback and Query Expansion

Introduction

- Most users find it difficult to formulate queries that are well designed for retrieval purposes
- Search engine 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
- In here we examine various approaches for improving the initial query formulation

Introduction

- This 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
- In here we will refer to both of them as feedback methods
- We distinguish two basic approaches:
  - **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
- The expectation is that the query q_m will return a higher number of documents relevant to q
- 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
- However, 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

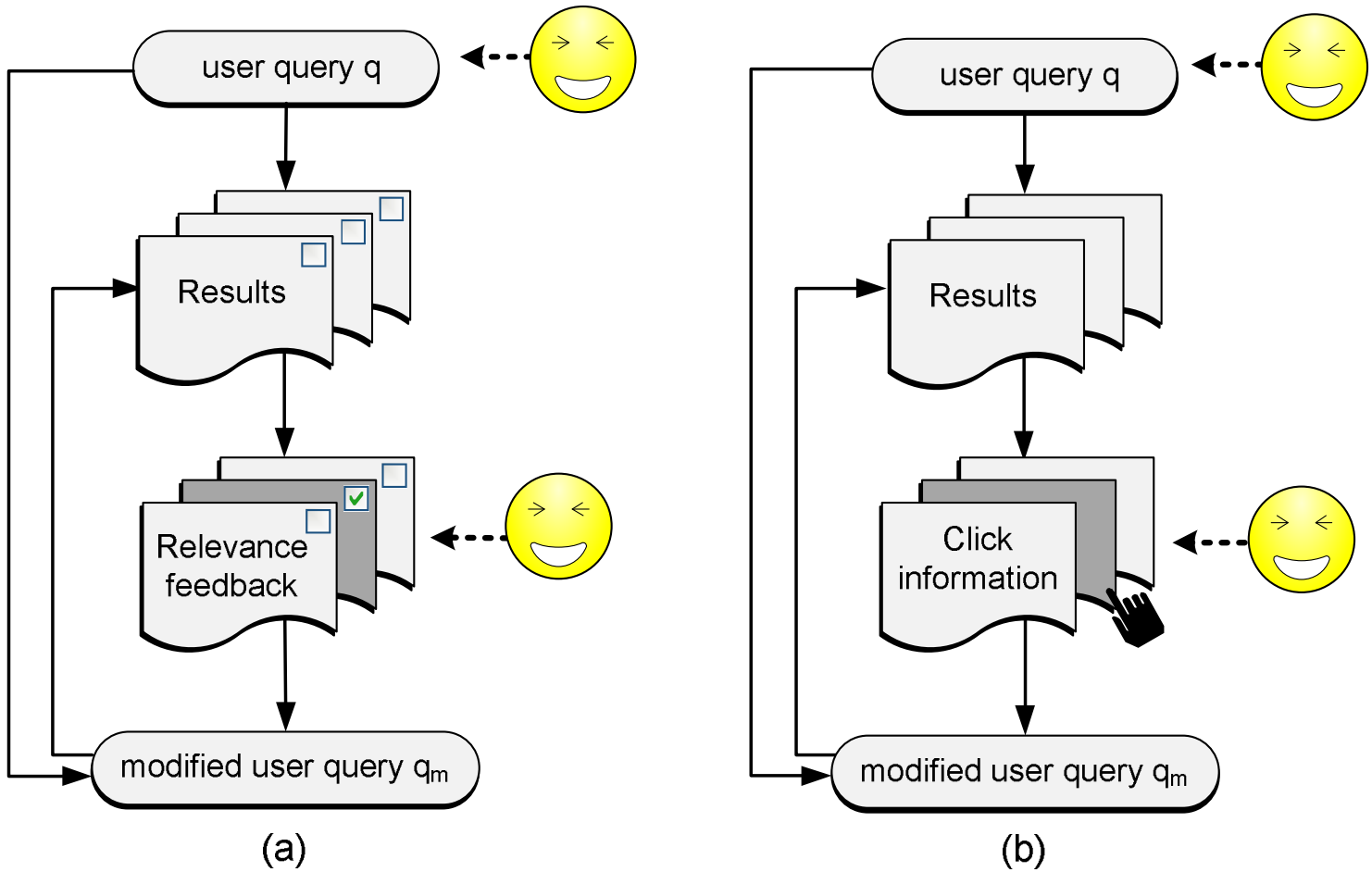
- A **feedback cycle** is composed of two basic steps:
  - Determine feedback information that is either related or expected to be related to the original query q and
  - Determine how to transform query q to take this information effectively into account
- The first step can be accomplished in two distinct 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
- In this case, the users inspect the top ranked docs and indicate those indeed relevant to the query
- 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

A Framework

- Explicit feedback information: (a) relevant results selected by the users and (b) results clicked on by the users

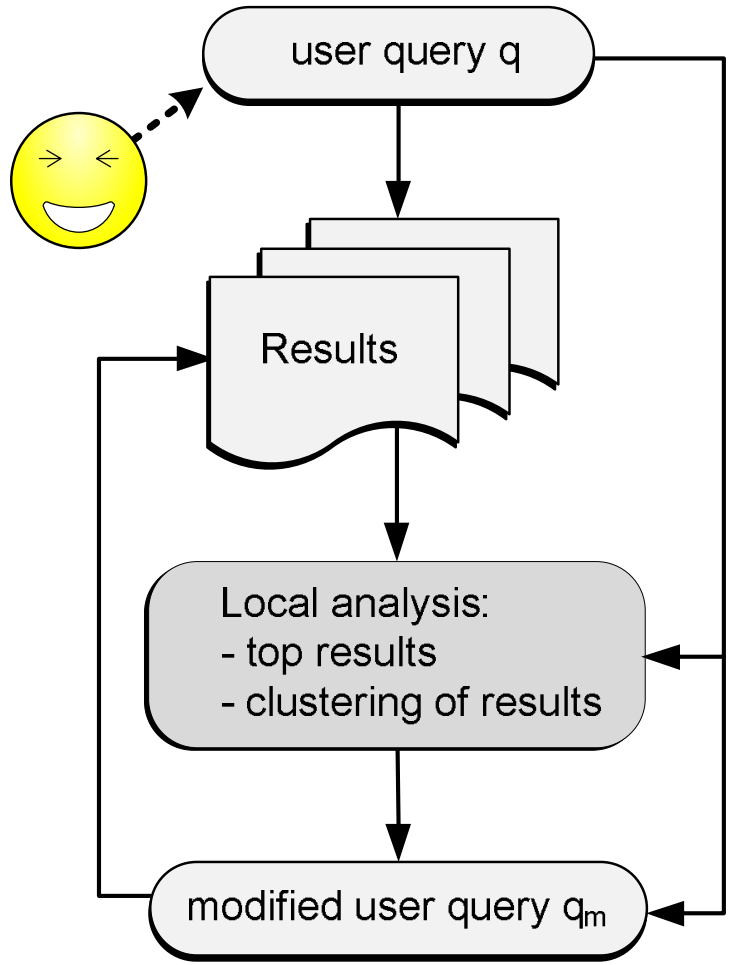


A Framework

- 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 derive the feedback information from the top ranked documents in the result set
  - **global analysis**, which derive the feedback information from external sources such as a thesaurus

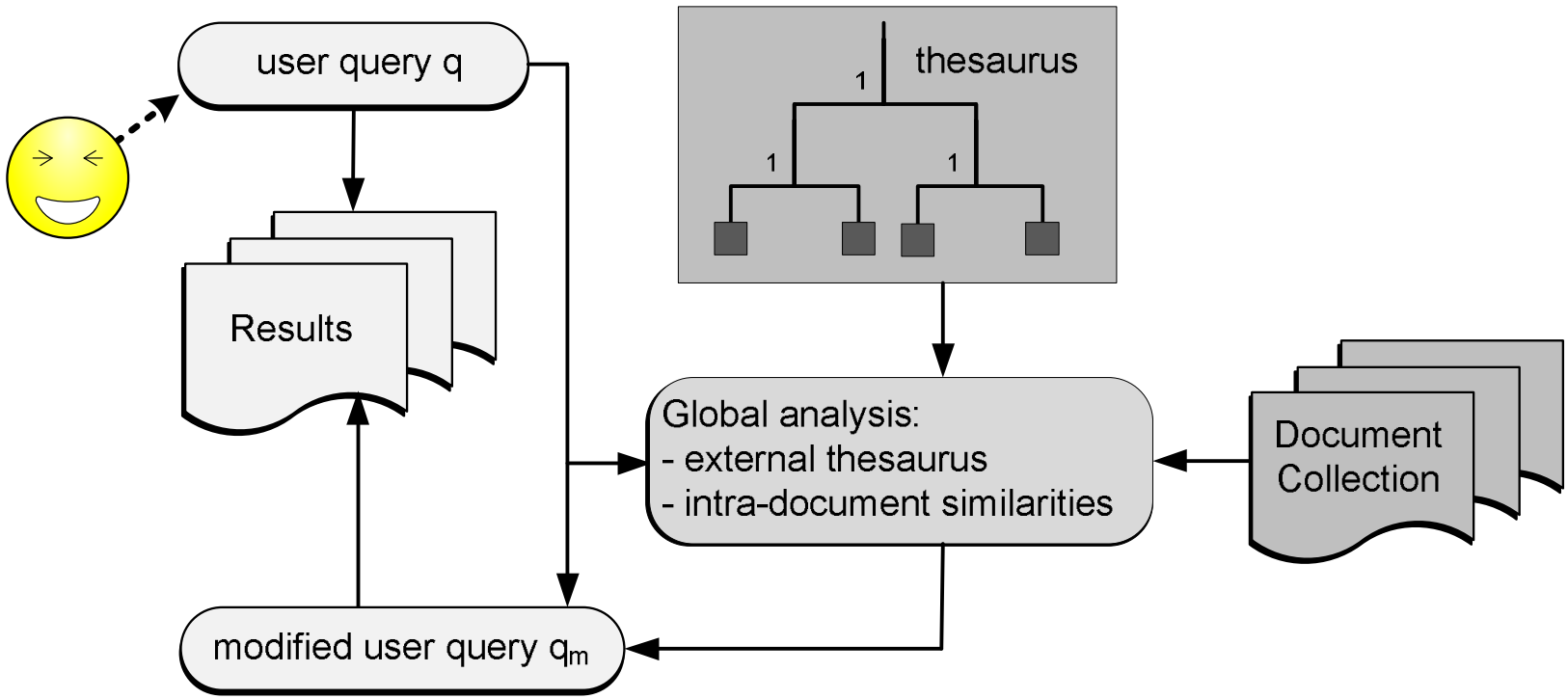
A Framework

- Implicit feedback information: local analysis



A Framework

- Implicit feedback information: global analysis



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
  - of 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

Explicit Relevance Feedback

The Rochio Method

The Rochio 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 Rochio 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 Rochio 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 Rochio Method

- Consider that the complete set C_r of relevant documents to a given query q 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 Rochio 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
- There are three classic and similar ways to calculate the modified query $\vec{q}_m$ as follows,

$$Standard\_Rochio: \quad \vec{q}_m = \alpha\,\vec{q} + \frac{\beta}{N_r}\sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \frac{\gamma}{N_n}\sum_{\forall \vec{d}_j \in D_n} \vec{d}_j$$

$$Ide\_Regular: \quad \vec{q}_m = \alpha\,\vec{q} + \beta\sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \gamma\sum_{\forall \vec{d}_j \in D_n} \vec{d}_j$$

$$Ide\_Dec\_Hi: \quad \vec{q}_m = \alpha\,\vec{q} + \beta\sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \gamma\; max\_rank(D_n)$$

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

The Rochio Method

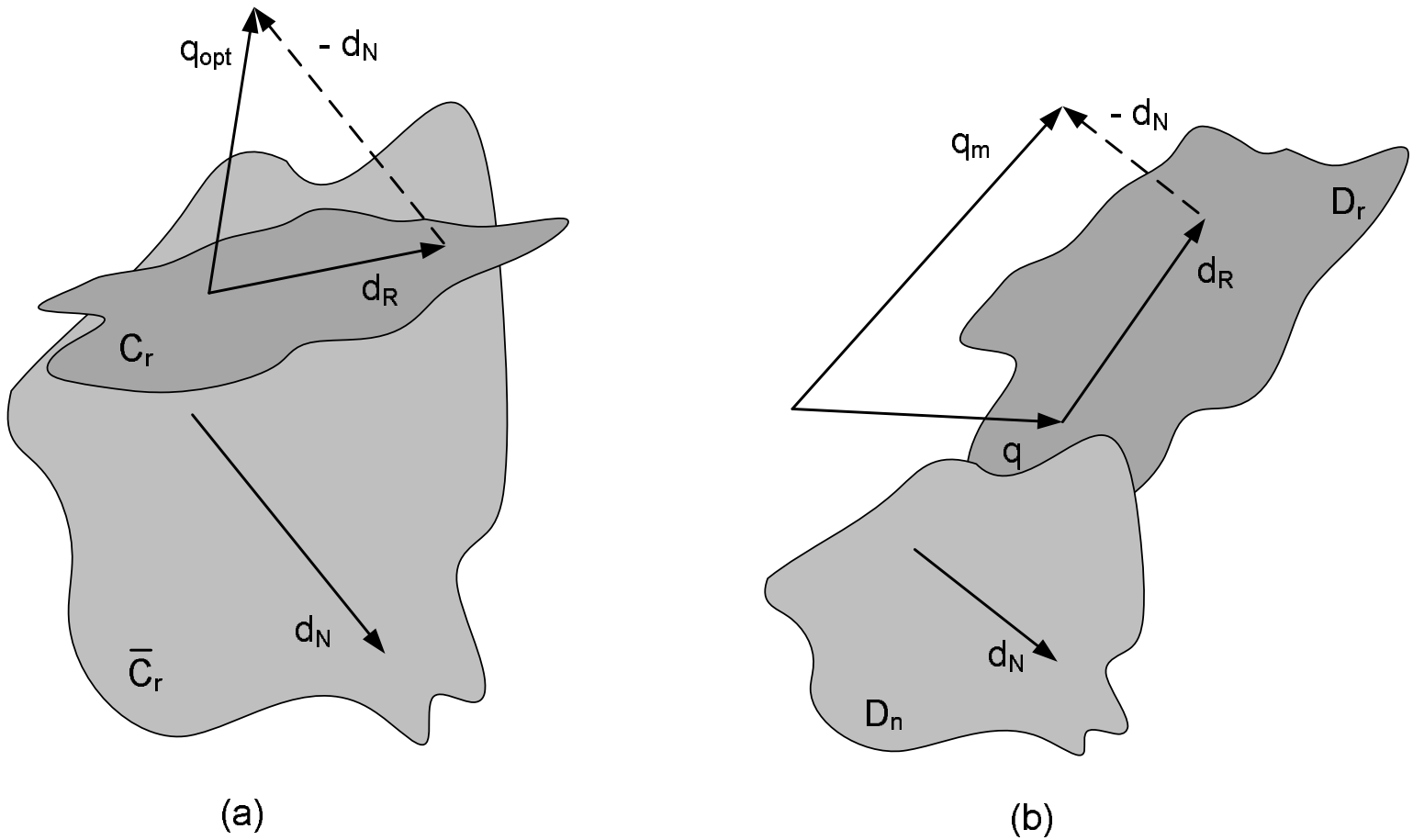
- In the original formulations, Rochio fixed $\alpha = 1$ and Ide fixed $\alpha = \beta = \gamma = 1$
- An alternative approach is to set γ to 0 which yields a *positive* feedback strategy
- The current understanding is that the three techniques yield similar results

The Rochio Method

- The Rochio feedback process:

  (a) optimal query formulation, when C_r is known

  (b) Rochio query reformulation process



The Rochio Method

- 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)

Explicit Relevance Feedback

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 on Modeling
- 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 Rochio Method, no query expansion occurs
- The same query terms are reweighted using feedback information provided by the user
- 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

Explicit Relevance Feedback

Evaluation of Relevance Feedback

Evaluation of Relevance Feedback

- Assume that we want to evaluate modified query vector $\vec{q}_m$ generated by the Rochio formula
- A approach is to retrieve a set of docs using $\vec{q}_m$, and evaluate this set by considering the relevant docs for $\vec{q}$
- In general, the results show spectacular improvements
- However, a part of this improvement results from the higher ranks assigned to the docs identified as relevant
- Since the user has seen these docs already, such evaluation is unrealistic
- Further, it masks any real gains in retrieval performance due to documents not seen by the user yet

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 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

Explicit Feedback Through Clicks

Eye Tracking and Relevance Judgements

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

Eye Tracking

- **Saccades** are rapid eye movements between fixations lasting 40-50 milliseconds
- This time interval is too short for effective information acquisition
- **Pupil dilation**, that is an indicator of arousal of interest, and **scan paths** are not discussed here

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

Relevance Judgements

- To illustrate, the study in Thorsten Joachims *et al.* uses a test set composed of 10 queries
- To compile click through data, real users need to be employed
- Each page and results set viewed by a user is evaluated for relevance by independent human assessors
- A human assessor is presented with a query and the top 10 results for that query
- Then, the assessor should decide on a strict preference ordering for all 10 results
- These relevance judgements can then be used to evaluate the quality of the docs clicked on by the users

Explicit Feedback Through Clicks

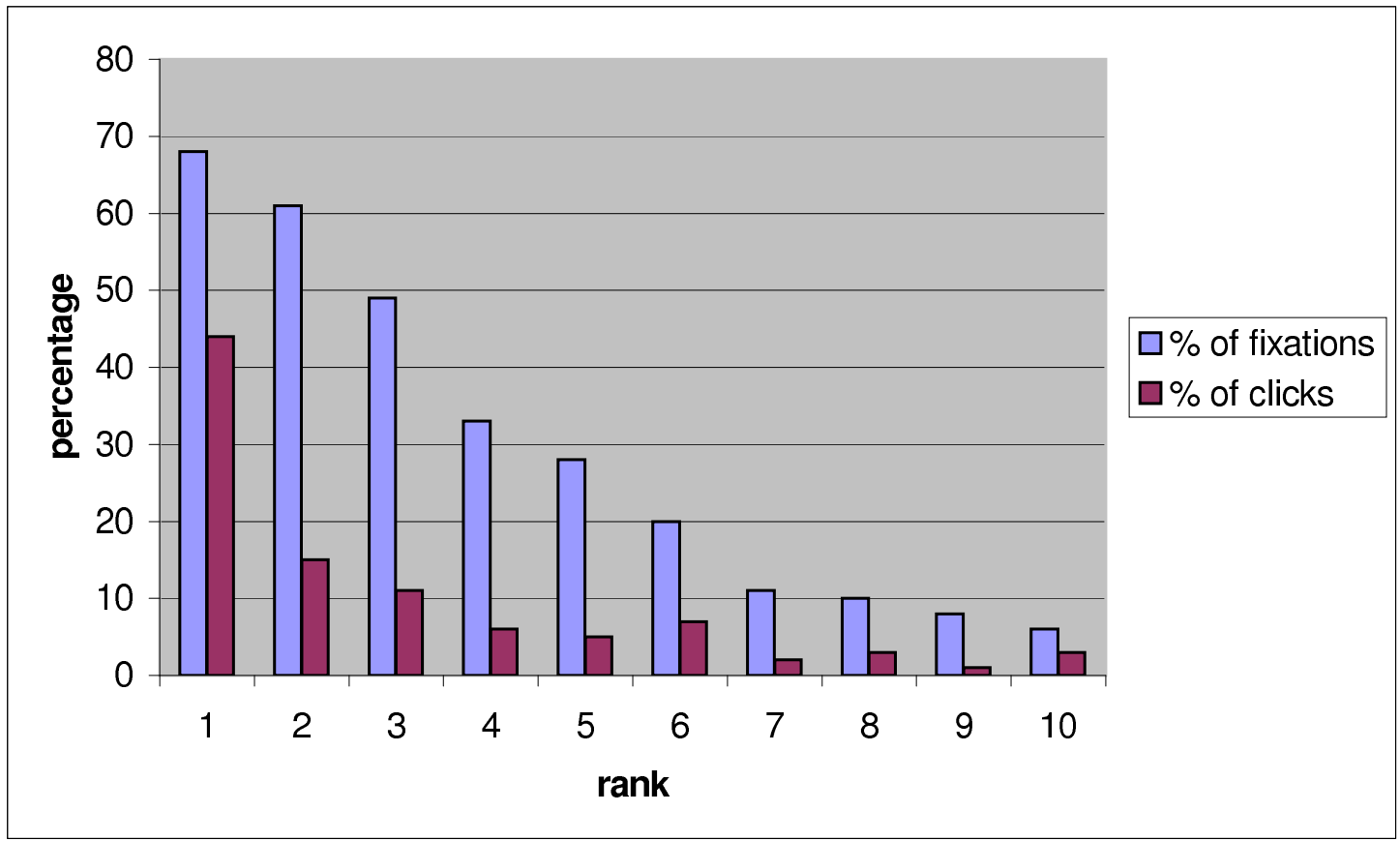
User Behavior

User Behavior

- Eye tracking experiments reported 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 considered in Thorsten Joachims *et al*.



User Behavior

- We notice that in 60-70% of the tasks the users have a fixation in either the first or the second result
- The relative frequency of fixation in the fourth result drops to half
- For the results at the bottom of the page, the relative frequency of fixation tends to be below 10%
- Also, the users inspect the top 2 answers almost equally, but they click three times more often 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

Explicit Feedback Through Clicks

Clicks as a Metric of User Preferences

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 for a result, and decides to skip it to click on a result that appears lower
- In this case, we can 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 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 $\sqrt{r_k}$ indicate that the user has clicked on the kth 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:

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

- This behavior does not allow us to make definitive statements about the relevance 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 $\sqrt{r_k}$ then $r_k > r_{k-n}$, for all r_{k-n} that was not clicked
  - Skip-Previous: if $\sqrt{r_k}$ 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:

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

- 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

- Conclusions derived, in agreement with the relevance judgements 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 Same Query

- 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 Same Query

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

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

  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 Same Query

- 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:

$$s_1 > r_1; \quad s_2 > r_1; \quad s_3 > r_1; \quad s_4 > r_1; \quad s_5 > r_1; \quad \ldots$$
$$s_1 > r_2; \quad s_2 > r_2; \quad s_3 > r_2; \quad s_4 > r_2; \quad s_5 > r_2; \quad \ldots$$

Clicks within a Same Query

- 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 was 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 Same Query

- 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

Explicit Feedback Through Clicks

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