Relevance Feedback
Information Systems M

Prof. Paolo Ciaccia

http://www-db.deis.unibo.it/courses/SI-M/


How can a user effectively search?

- All the tools and techniques we have detailed so far assume that

    the user "knows" how to formulate her queries/preferences

- Although with traditional DB's and a few attributes this might be a reasonable assumption, when we consider many attributes/features it is not clear how a user might guess the right combination of weights
    - How can one define the 64 weights for a color-based search using the weighted Euclidean distance?

- Possible solutions:
    - Qualitative preferences (e.g., Skyline): This model is likely to return too many results in high-dimensional spaces, thus one should consider, say, *ranked skyline queries*
    - Browsing: Strictly speaking, there is no search activity anymore, since the user "explores" the DB. To this end, one should devise effective metodhs for presenting objects and for supporting navigation
    - User feedback

The idea of relevance feedback

- The basic idea of relevance feedback is to shift the burden of finding the "right query formulation" from the user to the system
- For this being possible, the user has to provide the system with some information about "how well" the system has performed in answering the original query
- This user feedback typically takes the form of *relevance judgements* expressed over the answer set
- The "feedback loop" can then be iterated multiple times, until the user gets satisfied with the answers
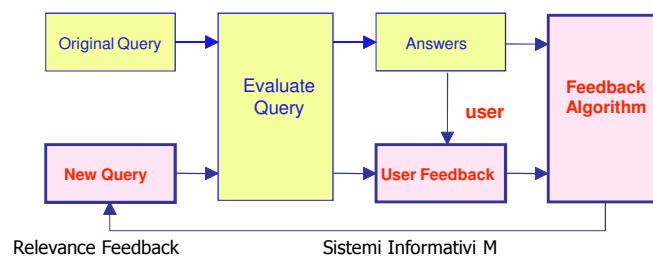
Relevance judgments

- The most common way to evelute the results is based on a 3-valued assessment:

Relevant: the object is relevant to the user

Non-relevant: the object is definitely not relevant (false drop)

Don't care: the user does not say anything about the object

- Information provided by the relevant objects constitutes the so-called "positive feedback", whereas non-relevant objects provide the so-called "negative feedback"
    - It's common the case of systems that only allow for positive feedback

- "Don't care" is needed also to avoid the user the task of assessing the relevance of **all** the results
- Models that allow a finer assessment of results (e.g., relevant, very relevant, etc.) have also been developed

A practical example (1)

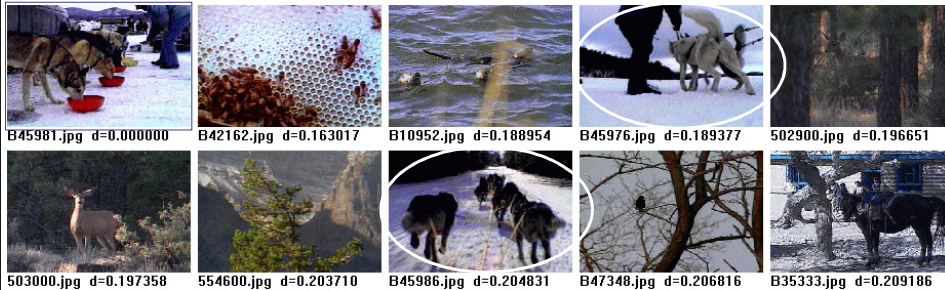Euclidean distance — **32-D HSV histograms**

QueryImage

B45981.jpg d=0.000000   B42162.jpg d=0.163017   B10952.jpg d=0.188954   B45976.jpg d=0.189377   502900.jpg d=0.196651

503000.jpg d=0.197358   554600.jpg d=0.203710   B45986.jpg d=0.204831   B47348.jpg d=0.206816   B35333.jpg d=0.209186

This is the initial query, for which 2 object are assessed as relevant by the user
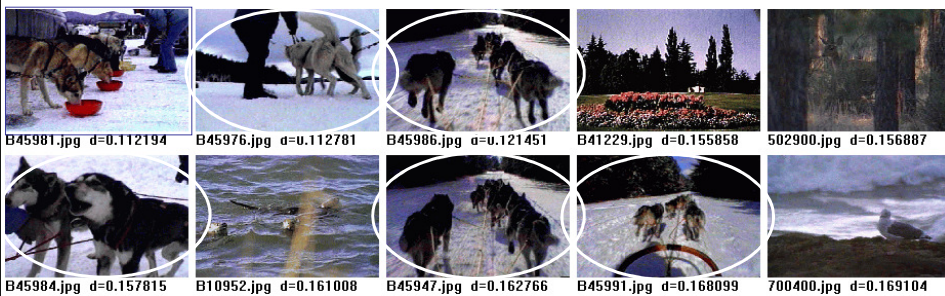
Precision = 0.3 (including the query image)

A practical example (2)

QueryImage

B45981.jpg d=0.112194   B45976.jpg d=0.112781   B45986.jpg d=0.121451   B41229.jpg d=0.155858   502900.jpg d=0.156887

B45984.jpg d=0.157815   B10952.jpg d=0.161008   B45947.jpg d=0.162766   B45991.jpg d=0.168099   700400.jpg d=0.169104

These are the results of the "refined" (new) query, generated using the 1st strategy we will see

Precision = 0.6 (including the query image)

3

A practical example (3)

QueryImage

B45981.jpg d=0.000000   B45984.jpg d=0.000578   B45976.jpg d=0.000660   B45986.jpg d=0.000696   B45985.jpg d=0.000713

B45947.jpg d=0.000739   B45974.jpg d=0.000787   B45962.jpg d=0.000809   B9836.jpg d=0.000913   B15920.jpg d=0.000943

These are the results of the "refined" (new) query, generated using the 2nd strategy we will see

Precision = 0.8 (including the query image)

Relevance Feedback   Sistemi Informativi M   7

A practical example (4)

QueryImage

B45981.jpg d=0.000385   B45976.jpg d=0.000394   B45986.jpg d=0.000414   B45947.jpg d=0.000508   B45984.jpg d=0.000509

B45985.jpg d=0.000602   B45974.jpg d=0.000702   B45962.jpg d=0.000786   B9836.jpg d=0.000892   B45963.jpg d=0.000894

And these are the results obtained by combining the 2 strategies…

Precision = 0.9 (including the query image)

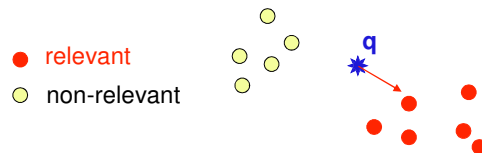Relevance Feedback   Sistemi Informativi M   8

4

Basic query refinement strategies

- When the feature values are vectors, two basic strategies for obtaining a refined query from the previous one and from the user feedback are:

Query point movement:

the idea is simply to move the query point so as to get closer to relevant objects

● relevant

○ non-relevant

Re-weighting:

the idea is to change the weights of the features so as to give more importance to those features that better capture, for the given query at hand, the notion of relevance

---

Query point movement

- The 1st formulation of the query point movement (QPM) strategy dates back to 70's, when it was proposed by J.J. Rocchio in the context of text retrieval systems based on the Vector Space model

- Rocchio's formula is:

$$q_{new} = q_{old} + \beta \times \frac{\sum_{p_j \in Rel}(p_j - q_{old})}{|Rel|} - \gamma \times \frac{\sum_{p_j \in NonRel}(p_j - q_{old})}{|NonRel|}$$

where:

- q_{old} is the previous query point
- Rel is the set of relevant objects that have been retrieved by q_{old},
- NonRel is the set of non-relevant objects that have been retrieved by q_{old},
- β and γ are non-negative parameters that control at which speed the query point moves towards relevant objects and far from non-relevant objects

QPM: geometric view

- Basically, Rocchio's formula adds to the (scaled) old query point the (scaled) centroid, g, of relevant ("good") obejcts, and subtracts the (scaled) centroid, b, of non-relevant ("bad") objects:
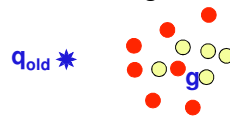
$$q_{new} = q_{old} + \beta \times (g - q_{old}) - \gamma \times (b - q_{old})$$
$$= (1 - \beta + \gamma) \times q_{old} + \beta \times g - \gamma \times b$$

$\beta = 0.6$
$\gamma = 0.4$
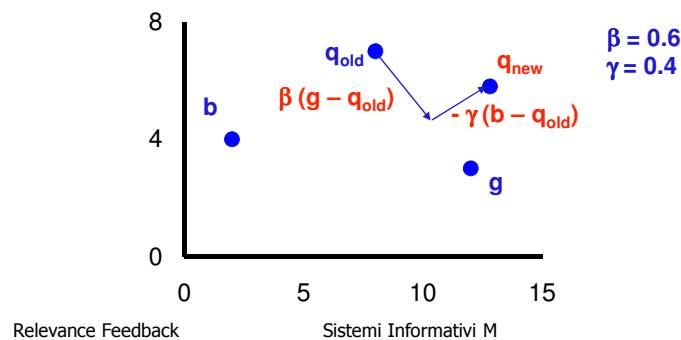
QPM: some observations

- Let $\gamma = 0$ and $\beta = 1$. Then $q_{new} = g$, thus the new query point coincides with the center of relevant objects
- This strategy (which is the 1st one used in the image retrieval example) can sometimes lead to "overshoot" the region of relevant objects


- Overshooting can also occur with large values of γ. Indeed, it's easy to construct examples where negative feedback will move the query point towards non-relevant objects
    - This is a reason why negative feedback is rarely used, even if some recent proposals [AGG02] present more robust solutions
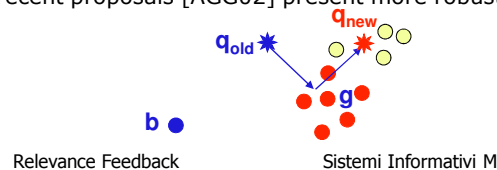
Re-weighting

- The idea of the re-weighting strategy is to analyze the relevant objects in order to understand if some feature (dimension) is more important than others in determining "what makes an object relevant"
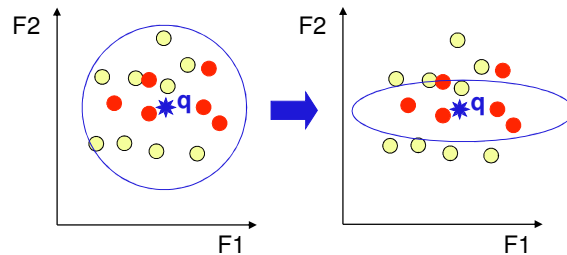
> - The feature F2 allows a better discrimination than F1 of relevant and non-relevant objects


Variance-based re-weighting

- For the relevant case of weighted Euclidean distances, the re-weighting strategy is easily implemented as follows:
    - Let Rel = {p_1,…,$p_{|Rel|}$} be the set of relevant objects retrieved by q_{old}
    - Let $p_{i,j}$ be the feature value of p_j for the i-th feature (i=1,…,D)

- The weight w_i of the i-th feature is estimated as $w_i \propto 1/\sigma_i^2$, that is, the inverse of the variance of feature values along the i-th coordinate
    - In the figure $w_2 > w_1$ since the variance on F2 is less than the variance on F1

- Besides the intuition, this strategy has a theoretical justification, which relies on the minimization of distances from the relevant objects [RH00]
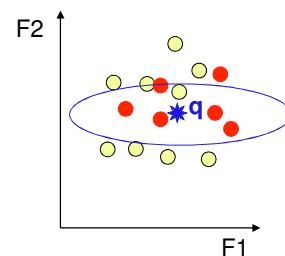
Other approaches

- Several other approaches to implement relevance feedback strategies exist
- In particular:

MindReader [ISF98] solves the problem by looking for the optimal ellipsoid that minimizes the sum of distances from relevant objects

Query expansion techniques replace the original query point with multiple query points
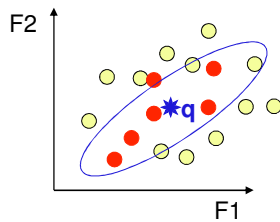
However, when |Rel| < D, the corresponding linear optimization problem is unconstrained, and the approach is not applicable
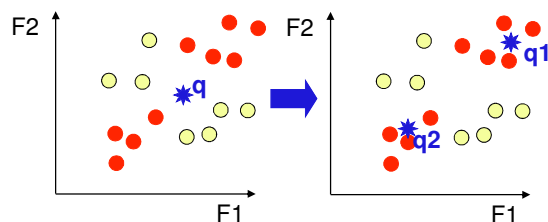
The technique requires smarter execution strategies, so as to avoid deterioration of performance due to the multiple query points [COM+04]

User feedback: other scenarios

- Relevance feedback is the basic mechanism to implement an effective user-system interaction

- User feedback also arises in other contexts:

  - If the systems keeps trace of user feedback through time, this will lead to the formation of "user profiles", which can subsequently be exploited for selectively disseminating new information (*information filtering*)

  - If what is returned to a given user exploits the feedback ("opinions") expressed by other users, we move towards the areas of *collaborative filtering* and *recommender systems*