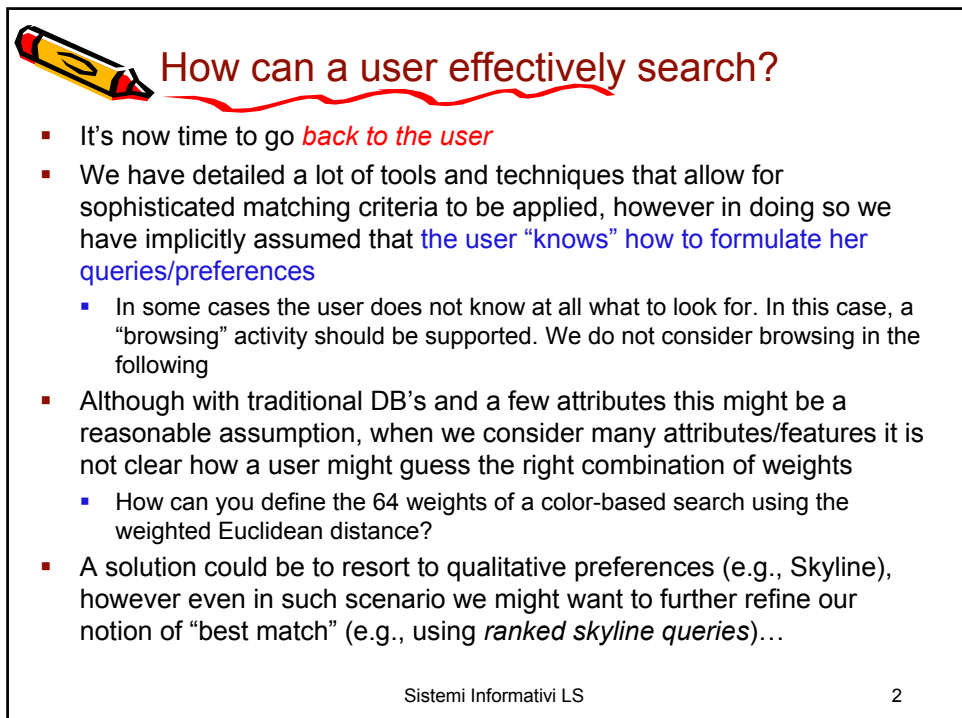


Relevance Feedback

Prof. Paolo Ciaccia
<http://www-db.deis.unibo.it/courses/SI-LS/>
10_RelevanceFeedback.pdf

Sistemi Informativi LS



How can a user effectively search?

- It's now time to go *back to the user*
- We have detailed a lot of tools and techniques that allow for sophisticated matching criteria to be applied, however in doing so we have implicitly assumed that the user "knows" how to formulate her queries/preferences
 - In some cases the user does not know at all what to look for. In this case, a "browsing" activity should be supported. We do not consider browsing in the following
- 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 you define the 64 weights of a color-based search using the weighted Euclidean distance?
- A solution could be to resort to qualitative preferences (e.g., Skyline), however even in such scenario we might want to further refine our notion of "best match" (e.g., using *ranked skyline queries*)...

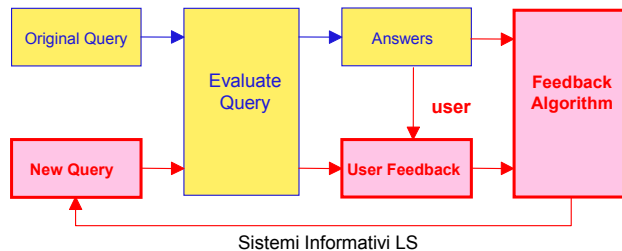
Sistemi Informativi LS

2



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



3



Relevance judgments

- The commonest way to evaluate 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)

QueryImage

Euclidean distance

32-D HSV histograms



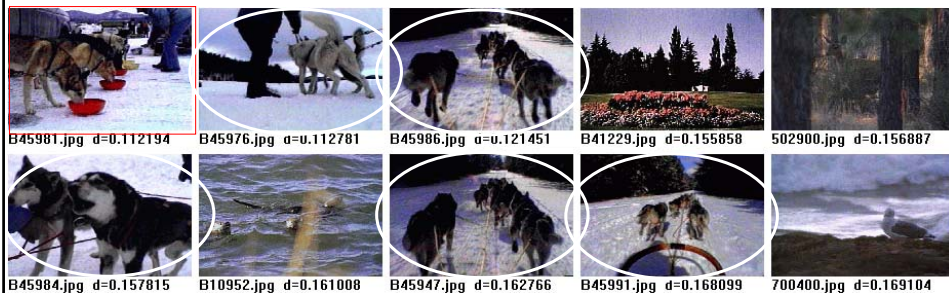
This is the initial query, for which 2 object are assessed as relevant by the user

Precision = 0.3 (including the query image)



A practical example (2)

QueryImage



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

Precision = 0.6 (including the query image)



A practical example (3)

QueryImage



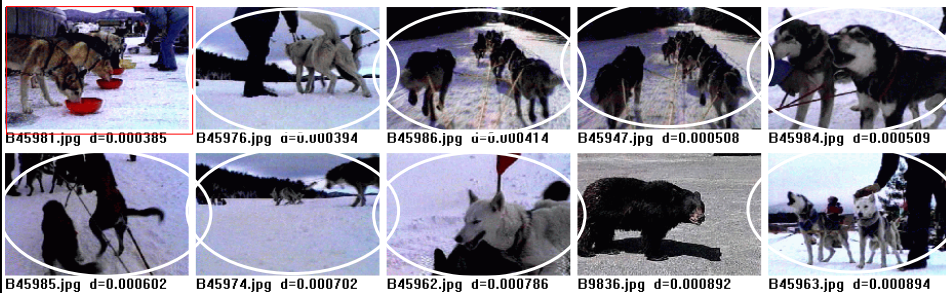
These are the results of the “refined” (new) query, generated using the 2nd strategy we will see

Precision = 0.8 (including the query image)



A practical example (4)

QueryImage



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

Precision = 0.9 (including the query image)

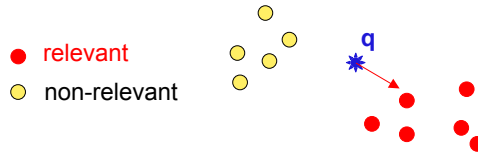


Basic query refinement strategies

- 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



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:

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

where:

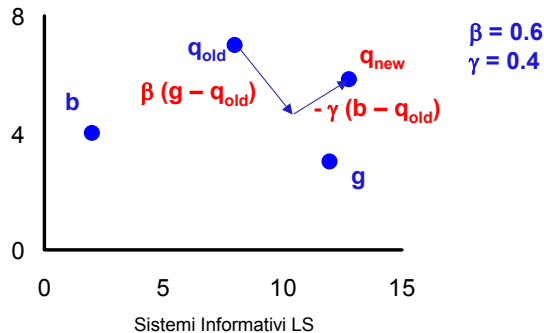
- q_{old} is the previous query point
- Rel is the set of relevant objects that have been retrieved by q_{old} ,
- NonRel is the set of non-relevant objects that have been retrieved by q_{old} ,
- β and γ are non-negative parameters that control at which speed the query point moves towards relevant objects 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") objects, and subtracts the (scaled) centroid, b , of non-relevant ("bad") objects:

$$\begin{aligned} q_{\text{new}} &= q_{\text{old}} + \beta \times (g - q_{\text{old}}) - \gamma \times (b - q_{\text{old}}) \\ &= (1 - \beta + \gamma) \times q_{\text{old}} + \beta \times g - \gamma \times b \end{aligned}$$

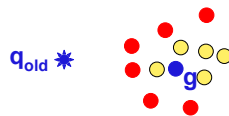


11

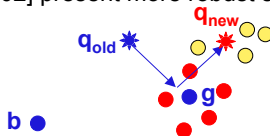


QPM: some observations

- Let $\gamma = 0$ and $\beta = 1$. Then $q_{\text{new}} = g$, 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



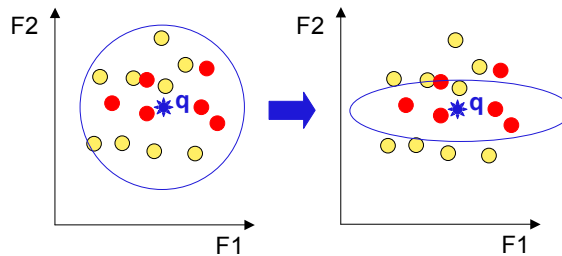
Sistemi Informativi LS

12



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”

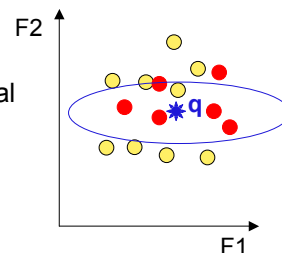


- The feature F2 allows a better discrimination than F1 of relevant and non-relevant objects



Variance-based re-weighting

- For the relevant case of **weighted Euclidean distances**, the re-weighting strategy is easily implemented as follows:
 - Let $Rel = \{p_1, \dots, p_{|Rel|}\}$ 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 ($i=1, \dots, D$)
- 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]

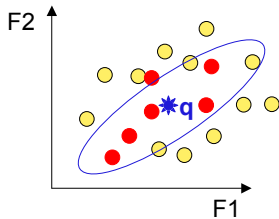




Other approaches

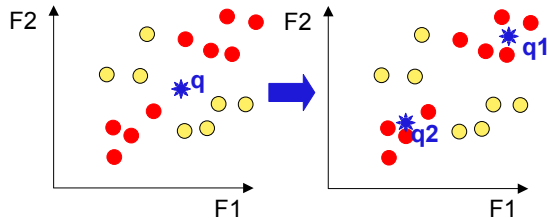
- Several other approaches to implement relevance feedback strategies exist
- In particular:

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



However, when $|Rel| < D$, the corresponding linear optimization problem is unconstrained, and the approach is not applicable

Query expansion techniques replace the original query point with multiple query points



The technique requires smarter execution strategies, so as to avoid deterioration of performance due to the multiple query points [COM+04]

Sistemi Informativi LS

15



Beyond relevance feedback

- Relevance feedback is the basic mechanism to implement an effective user-system interaction
- Relevance feedback principles can also be used 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 also exploits the **feedback** (“opinions”) **expressed by other users**, we move towards the areas of *collaborative filtering* and *recommender systems*

Sistemi Informativi LS

16