

## 2.5 User Adaptation

Take into account information of a user to filter document particularly relevant to this user

- ◆ Relevance Feedback
  - Retrieval in multiple passes; in each pass the user refines the query based on results of previous queries
- ◆ Explicit User Profiles
  - subscription
  - User-specific weights of terms
- ◆ Social Filtering
  - Similar use get similar documents

### 2.5.1 Relevance Feedback given by the User

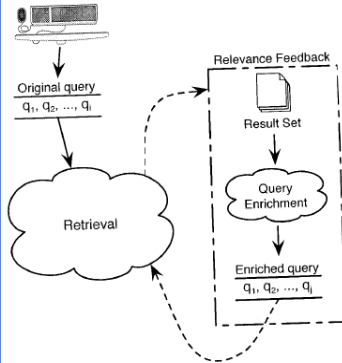
Example:

The user specifies relevance of each document.  
Example: for query "Pisa" only the documents about the education assessment are regarded as relevant

In the next pass, the top ranked documents are only about the education assessment

This example is from the SmartFinder system from empolis.  
The mindaccess system from Insiders GmbH uses the same technology.

## Relevance Feedback: Probabilistic Model



- Assumption: Given a user query, there is an *ideal* answer set
- Idea: An initial answer is iteratively improved based on user feedback
- Approach:
  - ◆ An initial set of documents is retrieved somehow
  - ◆ User inspects these docs looking for the relevant ones (usually, only top 10-20 need to be inspected)
  - ◆ IR system uses this information to refine description of ideal answer set
  - ◆ By repeating this process, it is expected that the description of the ideal answer set will improve
- The description of ideal answer set is modeled in probabilistic terms

(Baeza-Yates & Ribeiro-Neto 1999)

## Probabilistic Ranking

- Given a user query  $q$  and a document  $d_j$ , the probabilistic model tries to estimate the probability that the user will find the document  $d_j$  interesting (i.e., relevant).
- The model assumes that this probability of relevance depends on the query and the document representations only.
- Probabilistic ranking is:

$$sim(d_j, q) = \frac{P(R|\vec{d}_j)}{P(\bar{R}|\vec{d}_j)}$$

- Definitions:

$$w_{ij} \in \{0, 1\}$$

(i.e. weights are binary)

$sim(d_j, q)$  similarity of document  $d_j$  to the query  $q$

$P(R|\vec{d}_j)$  is the probability that document  $d_j$  is relevant

$P(\bar{R}|\vec{d}_j)$  is the probability that document  $d_j$  is **not** relevant

$|\vec{d}_j$  is the document vector of  $d_j$  (Baeza-Yates & Ribeiro-Neto 1999)

## Computing Probabilistic Ranking

- Probabilistic ranking can be computed as:

$$\text{sim}(d_j, q) \sim \sum_{i=1}^l w_{i,q} \times w_{i,j} \times \left( \log \frac{P(k_i|R)}{1 - P(k_i|R)} + \log \frac{1 - P(k_i|\bar{R})}{P(k_i|\bar{R})} \right)$$

- where

$P(k_i|R)$  stands for the probability that the index term  $k_i$  is present in a document randomly selected from the set  $R$  of relevant documents

$P(\bar{k}_i|\bar{R})$  stands for the probability that the index term  $k_i$  is **not** present in a document randomly selected from the set  $R$  of relevant documents

$w_{i,q}$  is the weight of term  $k_i$  in the query

$w_{i,j}$  is the weight of term  $k_i$  in document  $d_j$

(Baeza-Yates & Ribeiro-Neto 1999)

## Relevance Feedback: Probabilistic Model

- The probabilities that a term  $k_i$  is (not) present in a set of relevant documents can be computed as :

$$P(k_i|R) = \frac{V_i}{V}$$

$$P(k_i|\bar{R}) = \frac{n_i - V_i}{N - V}$$

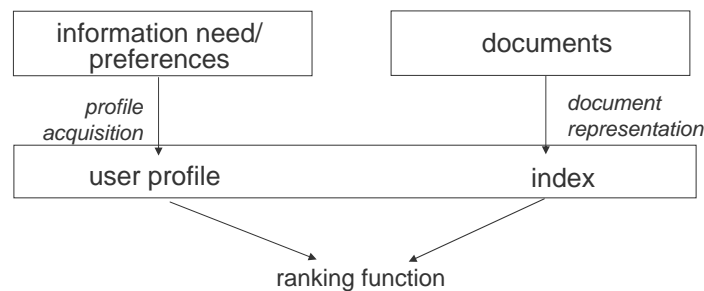
$N$	total number of documents
$n_i$	number of documents containing term $k_i$
$V$	number of relevant documents retrieved by the probabilistic model
$V_i$	number of relevant documents containing term $k_i$

- There are different ways to find the relevant document  $V$  :
  - ◆ Automatically:  $V$  can be specified as the top  $r$  documents found
  - ◆ By user feedback: The user specifies for each retrieved document whether it is relevant or not

## 2.5.2 Explicit User Profiles

Idea: Using knowledge about the user to provide information that is particularly relevant for him/her

- ♦ users specify topics of interest as a set of terms
- ♦ these terms represent the user profile
- ♦ documents containing the terms of the user profile are preferred



## User profiles for subscribing to information

	IDF	d1	d2	d3	U1	U2
accident	0.5	2	0	1	1	1
car	0.5	1	1	0	1	0
cause	1	0	0	1	0	0
crowd	1	0	0	1	0	0
die	1	1	0	0	0	1
drive	1	0	0	1	0	0
four	1	0	0	1	0	0
heavy	1	2	0	0	0	0
injur	1	0	0	1	0	0
more	1	0	2	0	0	0
morning	1	1	0	0	0	0
people	0.5	1	0	2	0	0
quarter	1	0	1	0	0	0
register	1	0	1	0	0	0
truck	1	0	0	1	1	1
trucker	1	0	0	1	0	0
vehicle	1	0	1	0	1	0
vienna	0.33	1	1	1	0	0
yesterday	1	1	0	0	0	0

- user profiles are treated as queries
- Example: news feed  
As soon as a new document arrives it is tested for similarity with the user profiles
- Vector space model can be applied
- A document is regarded relevant if the ranking reaches a specified threshold

Example: User 1 is interested in any car accident

User 2 is interested in deadly car accidents with trucks

## User profiles for Individual Queries

Example: users profiles with term

	IDF	d1	d2	d3	U1	U2	q
accident	0.5	2	0	1	1	1	1
car	0.5	1	1	0	0.8	0.2	0
cause	1	0	0	1	0	0	0
crowd	1	0	0	1	0	0	0
die	1	1	0	0	0	0.8	0
drive	1	0	0	1	0	0	0
four	1	0	0	1	0	0	0
heavy	1	2	0	0	0.2	0.6	1
injur	1	0	0	1	0	0	0
more	1	0	2	0	0	0	0
morning	1	1	0	0	0	0	0
people	0.5	1	0	2	0.5	0.8	0
quarter	1	0	1	0	0	0	0
register	1	0	1	0	0	0	0
truck	1	0	0	1	0.6	1	0
trucker	1	0	0	1	0	0.6	0
vehicle	1	0	1	0	1	0.1	1
vienna	0.33	1	1	1	0	0	1
yesterday	1	1	0	0	0	0	0

■ Users specify importance of terms

■ User profiles are used as additional term weights

■ Different ranking for different users

■ Example

ranking for user 1

$IDF * d1 * U1 * q =$

$IDF * d2 * U1 * q =$

$IDF * d3 * U1 * q =$

ranking for user 2

$IDF * d1 * U2 * q =$

$IDF * d2 * U2 * q =$

$IDF * d3 * U2 * q =$



## Acquisition and Maintenance of User Profiles

There are different ways to specify user profiles

manual: users specifies topics of interests (and weights) explicitly

- selection of predefined terms or query
- Problem: Maintenance

user feedback: user collects relevant documents

- terms in selected document are regarded as important
- Problem: How to motivate the user to give feedback
- (a similar approach is used by spam filters - classification)

Heuristics: observing user behaviour

- Example: If a user has opened a document for long time, it is assumed that he/she read it and therefore it might be relevant
- Problem: Heuristics might be wrong



## ***Social Filtering***

- Idea: Information is relevant, if other users who showed similar behaviour regarded the information as relevant
  - ◆ Relevance is specified by the users
  - ◆ User profiles are compared
- Example: A simple variant can be found at Amazon
  - ◆ purchases of books and CDs are stored
  - ◆ „people who bought this book also bought ...“

