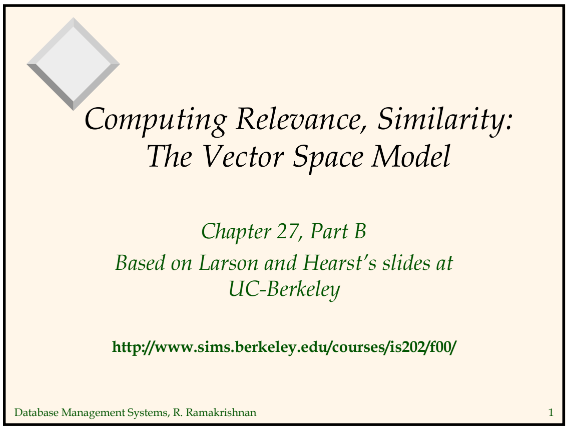
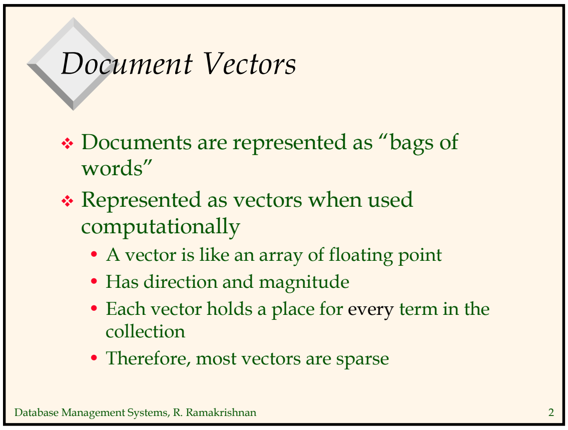


Computing Relevance, Similarity: The Vector Space Model

Chapter 27, Part B

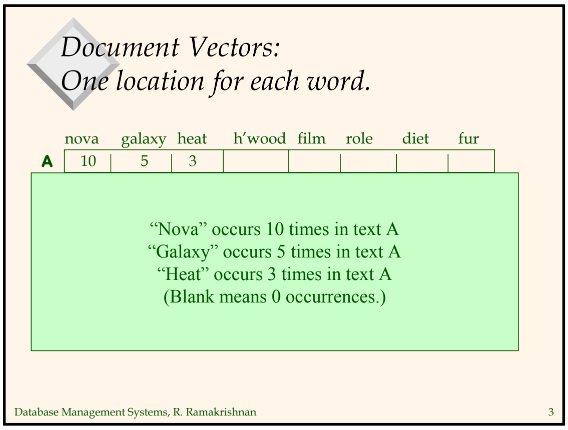
Based on Larson and Hearst's slides at UC-Berkeley

http://www.sims.berkeley.edu/courses/is202/f00/



Document Vectors

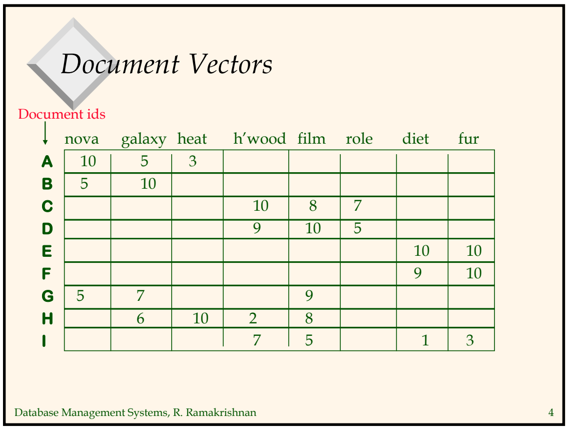
- Documents are represented as "bags of words"
- Represented as vectors when used computationally
  - A vector is like an array of floating point
  - Has direction and magnitude
  - Each vector holds a place for every term in the collection
  - Therefore, most vectors are sparse



Document Vectors: One location for each word.

	nova	galaxy	heat	h'wood	film	role	diet	fur
A	10	5	3					

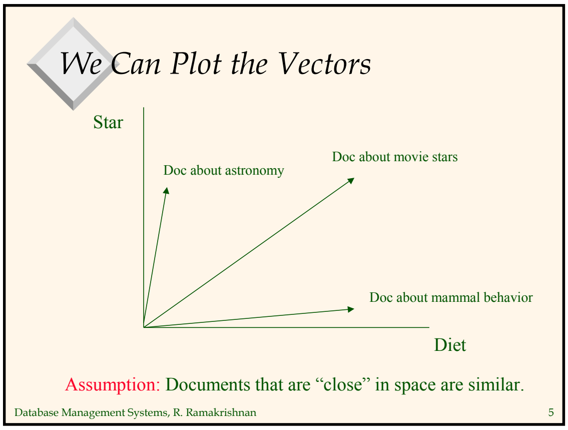
"Nova" occurs 10 times in text A
"Galaxy" occurs 5 times in text A
"Heat" occurs 3 times in text A
(Blank means 0 occurrences.)



Document Vectors

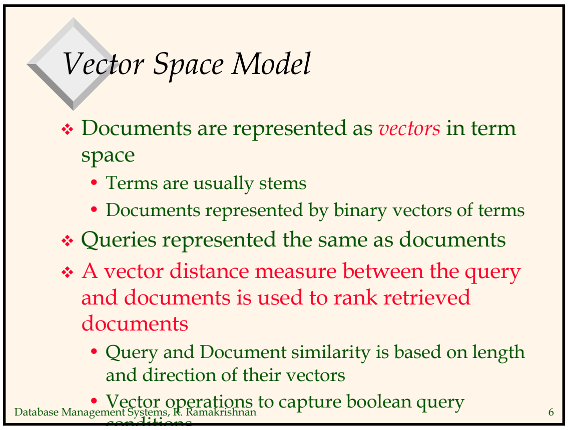
Document ids

	nova	galaxy	heat	h'wood	film	role	diet	fur
A	10	5	3					
B	5	10						
C				10	8	7		
D				9	10	5		
E							10	10
F							9	10
G	5	7			9			
H		6	10	2	8			
I				7	5		1	3



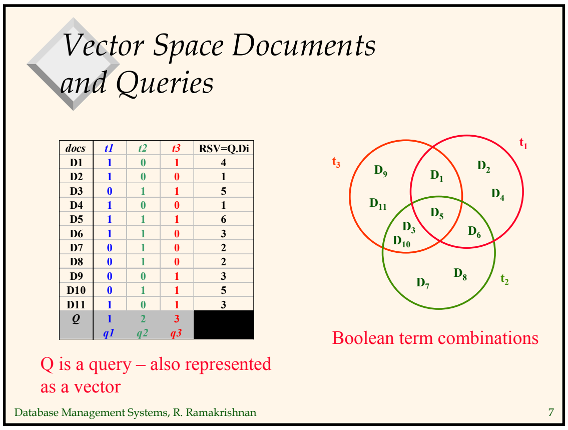
We Can Plot the Vectors

Assumption: Documents that are "close" in space are similar.



Vector Space Model

- Documents are represented as *vectors* in term space
  - Terms are usually stems
  - Documents represented by binary vectors of terms
- Queries represented the same as documents
- A vector distance measure between the query and documents is used to rank retrieved documents
  - Query and Document similarity is based on length and direction of their vectors
  - Vector operations to capture boolean query conditions



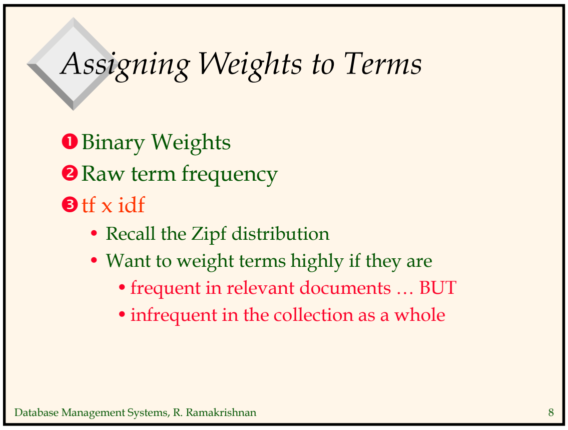
Vector Space Documents and Queries

docs	*t1*	*t2*	*t3*	RSV=Q.Di
D1	1	0	1	4
D2	1	0	0	1
D3	0	1	1	5
D4	1	0	0	1
D5	1	1	1	6
D6	1	1	0	3
D7	0	1	0	2
D8	0	1	0	2
D9	0	0	1	3
D10	0	1	1	5
D11	1	0	1	3
Q	1 *q1*	2 *q2*	3 *q3*	

Q is a query – also represented as a vector

t_3
t_1
D_9
D_1
D_2
D_4
D_{11}
D_5
D_3
D_{10}
D_6
D_8
D_7
t_2

Boolean term combinations



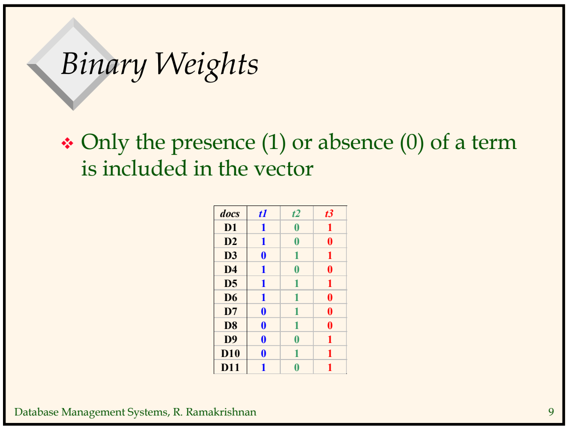
Assigning Weights to Terms

❶ Binary Weights

❷ Raw term frequency

❸ tf x idf

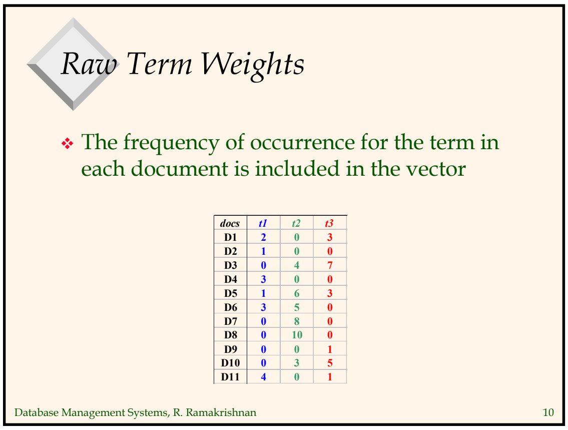
- Recall the Zipf distribution
- Want to weight terms highly if they are
  - frequent in relevant documents … BUT
  - infrequent in the collection as a whole



Binary Weights

❖ Only the presence (1) or absence (0) of a term is included in the vector

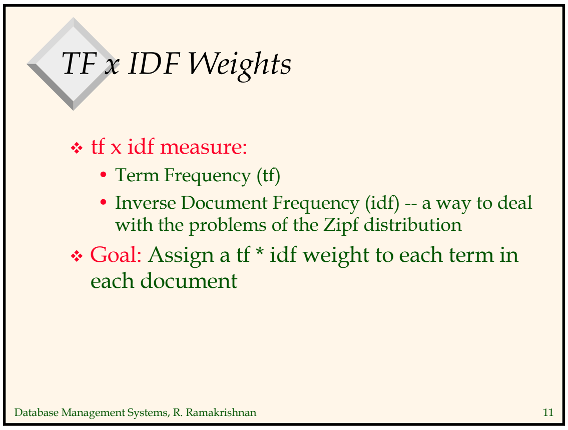
docs	*t1*	*t2*	*t3*
D1	1	0	1
D2	1	0	0
D3	0	1	1
D4	1	0	0
D5	1	1	1
D6	1	1	0
D7	0	1	0
D8	0	1	0
D9	0	0	1
D10	0	1	1
D11	1	0	1



Raw Term Weights

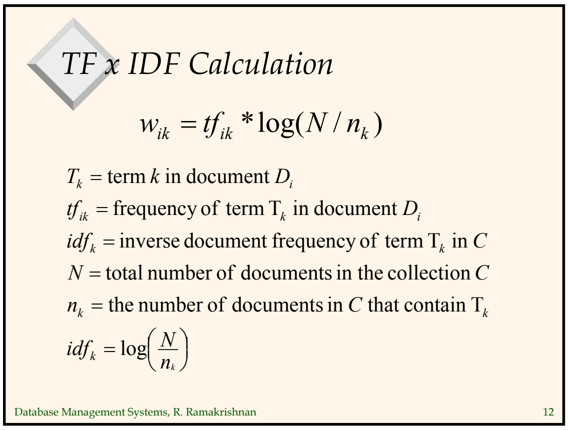
- The frequency of occurrence for the term in each document is included in the vector

docs	*t1*	*t2*	*t3*
D1	2	0	3
D2	1	0	0
D3	0	4	7
D4	3	0	0
D5	1	6	3
D6	3	5	0
D7	0	8	0
D8	0	10	0
D9	0	0	1
D10	0	3	5
D11	4	0	1



TF x IDF Weights

- tf x idf measure:
  - Term Frequency (tf)
  - Inverse Document Frequency (idf) -- a way to deal with the problems of the Zipf distribution
- Goal: Assign a tf * idf weight to each term in each document



TF x IDF Calculation

$$w_{ik} = tf_{ik} * \log(N / n_k)$$

T_k = term k in document D_i

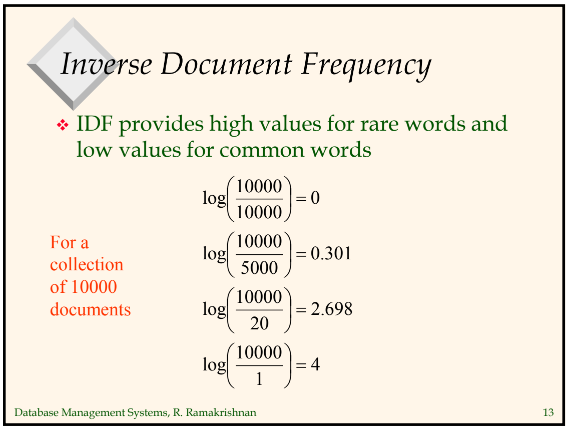
tf_{ik} = frequency of term T_k in document D_i

idf_k = inverse document frequency of term T_k in C

N = total number of documents in the collection C

n_k = the number of documents in C that contain T_k

$$idf_k = \log\left(\frac{N}{n_k}\right)$$



Inverse Document Frequency

- IDF provides high values for rare words and low values for common words

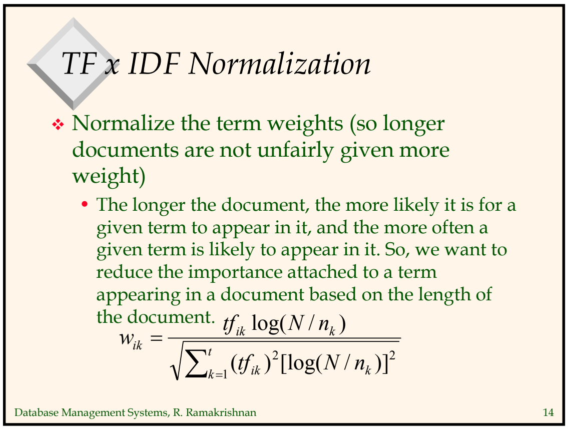
For a collection of 10000 documents

$$\log\left(\frac{10000}{10000}\right) = 0$$

$$\log\left(\frac{10000}{5000}\right) = 0.301$$

$$\log\left(\frac{10000}{20}\right) = 2.698$$

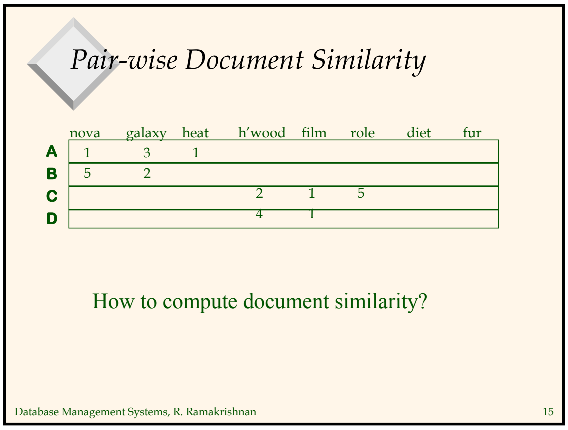
$$\log\left(\frac{10000}{1}\right) = 4$$



TF x IDF Normalization

- Normalize the term weights (so longer documents are not unfairly given more weight)
  - The longer the document, the more likely it is for a given term to appear in it, and the more often a given term is likely to appear in it. So, we want to reduce the importance attached to a term appearing in a document based on the length of the document.

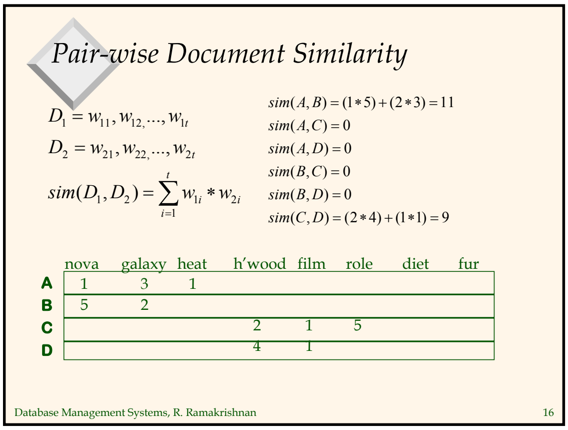
$$w_{ik} = \frac{tf_{ik}\log(N/n_k)}{\sqrt{\sum_{k=1}^{t}(tf_{ik})^2[\log(N/n_k)]^2}}$$



Pair-wise Document Similarity

	nova	galaxy	heat	h'wood	film	role	diet	fur
A	1	3	1					
B	5	2						
C				2	1	5		
D				4	1			

How to compute document similarity?

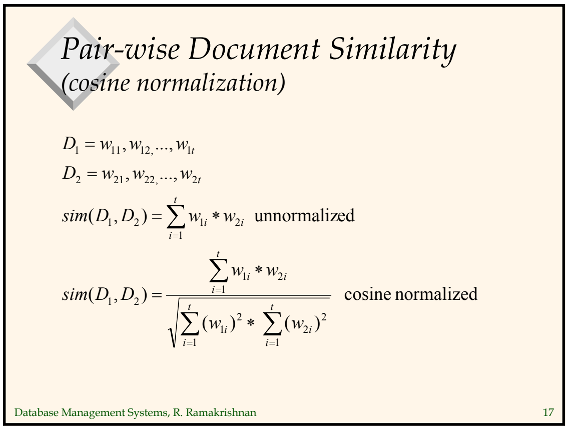


Pair-wise Document Similarity

$$D_1 = w_{11}, w_{12,} \dots, w_{1t}$$
$$D_2 = w_{21}, w_{22,} \dots, w_{2t}$$
$$sim(D_1, D_2) = \sum_{i=1}^{t} w_{1i} * w_{2i}$$

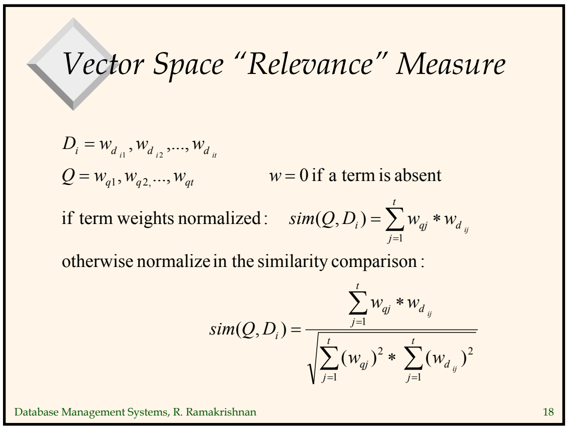
$$sim(A,B) = (1*5) + (2*3) = 11$$
$$sim(A,C) = 0$$
$$sim(A,D) = 0$$
$$sim(B,C) = 0$$
$$sim(B,D) = 0$$
$$sim(C,D) = (2*4) + (1*1) = 9$$

	nova	galaxy	heat	h'wood	film	role	diet	fur
A	1	3	1					
B	5	2						
C				2	1	5		
D				4	1			



Pair-wise Document Similarity
(cosine normalization)

$$D_1 = w_{11}, w_{12,} \dots, w_{1t}$$
$$D_2 = w_{21}, w_{22,} \dots, w_{2t}$$
$$sim(D_1, D_2) = \sum_{i=1}^{t} w_{1i} * w_{2i} \quad \text{unnormalized}$$
$$sim(D_1, D_2) = \frac{\sum_{i=1}^{t} w_{1i} * w_{2i}}{\sqrt{\sum_{i=1}^{t} (w_{1i})^2 * \sum_{i=1}^{t} (w_{2i})^2}} \quad \text{cosine normalized}$$



Vector Space "Relevance" Measure

$$D_i = w_{d_{i1}}, w_{d_{i2}}, \dots, w_{d_{it}}$$
$$Q = w_{q1}, w_{q2,} \dots, w_{qt} \qquad w = 0 \text{ if a term is absent}$$

if term weights normalized: $sim(Q, D_i) = \sum_{j=1}^{t} w_{qj} * w_{d_{ij}}$

otherwise normalize in the similarity comparison:

$$sim(Q, D_i) = \frac{\sum_{j=1}^{t} w_{qj} * w_{d_{ij}}}{\sqrt{\sum_{j=1}^{t} (w_{qj})^2 * \sum_{j=1}^{t} (w_{d_{ij}})^2}}$$

Computing Relevance Scores

Say we have query vector $Q = (0.4, 0.8)$

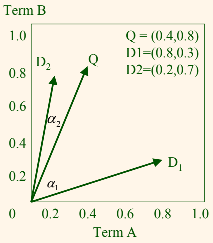
Also, document $D_2 = (0.2, 0.7)$

What does their similarity comparison yield?

$$sim(Q, D_2) = \frac{(0.4 * 0.2) + (0.8 * 0.7)}{\sqrt{[(0.4)^2 + (0.8)^2] * [(0.2)^2 + (0.7)^2]}}$$

$$= \frac{0.64}{\sqrt{0.42}} = 0.98$$

Vector Space with Term Weights and Cosine Matching



$D_i = (d_{i1}, w_{di1}; d_{i2}, w_{di2}; \ldots; d_{it}, w_{dit})$

$Q = (q_{i1}, w_{qi1}; q_{i2}, w_{qi2}; \ldots; q_{it}, w_{qit})$

$$sim(Q, D_i) = \frac{\sum_{j=1}^{t} w_{q_j} w_{d_{ij}}}{\sqrt{\sum_{j=1}^{t} (w_{q_j})^2 \sum_{j=1}^{t} (w_{d_{ij}})^2}}$$

$$sim(Q, D2) = \frac{(0.4 \cdot 0.2) + (0.8 \cdot 0.7)}{\sqrt{[(0.4)^2 + (0.8)^2] \cdot [(0.2)^2 + (0.7)^2]}}$$

$$= \frac{0.64}{\sqrt{0.42}} = 0.98$$

$$sim(Q, D_1) = \frac{.56}{\sqrt{0.58}} = 0.74$$

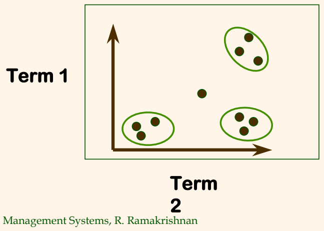
Text Clustering

- Finds overall similarities among groups of documents
- Finds overall similarities among groups of tokens
- Picks out some themes, ignores others

Text Clustering

Clustering is

> "The art of finding groups in data."
> -- Kaufmann and Rousseeu

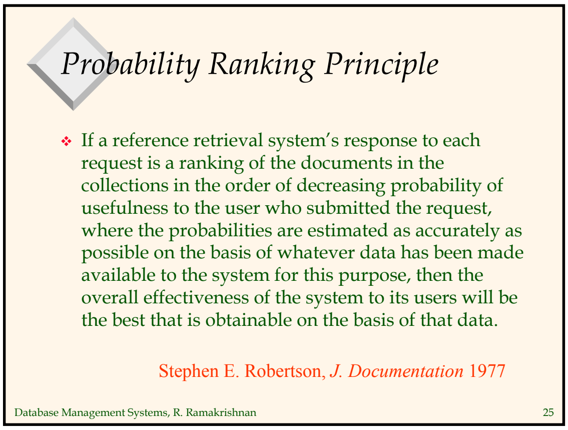


Problems with Vector Space

- There is no real theoretical basis for the assumption of a term space
  - It is more for visualization than having any real basis
  - Most similarity measures work about the same
- Terms are not really orthogonal dimensions
  - Terms are not independent of all other terms; remember our discussion of correlated terms in text

Probabilistic Models

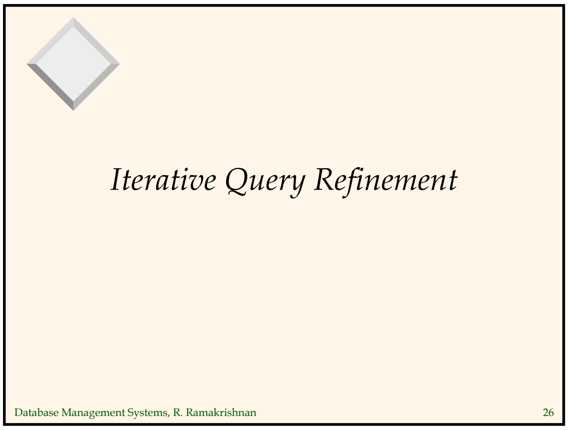
- Rigorous formal model attempts to predict the probability that a given document will be relevant to a given query
- Ranks retrieved documents according to this probability of relevance (Probability Ranking Principle)
- Relies on accurate estimates of probabilities



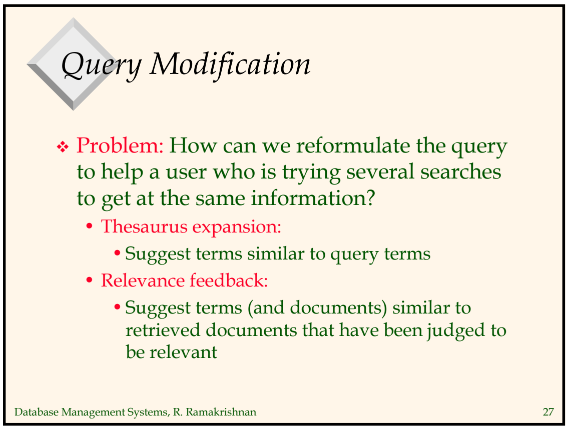
Probability Ranking Principle

- If a reference retrieval system's response to each request is a ranking of the documents in the collections in the order of decreasing probability of usefulness to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data has been made available to the system for this purpose, then the overall effectiveness of the system to its users will be the best that is obtainable on the basis of that data.

Stephen E. Robertson, *J. Documentation* 1977



Iterative Query Refinement



Query Modification

- Problem: How can we reformulate the query to help a user who is trying several searches to get at the same information?
  - Thesaurus expansion:
    - Suggest terms similar to query terms
  - Relevance feedback:
    - Suggest terms (and documents) similar to retrieved documents that have been judged to be relevant

Relevance Feedback

- Main Idea:
  - Modify existing query based on relevance judgements
    - Extract terms from relevant documents and add them to the query
    - AND/OR re-weight the terms already in the query
- There are many variations:
  - Usually positive weights for terms from relevant docs
  - Sometimes negative weights for terms from non-relevant docs
- Users, or the system, guide this process by

Rocchio Method

- Rocchio automatically
  - Re-weights terms
  - Adds in new terms (from relevant docs)
    - have to be careful when using negative terms
    - Rocchio is ***not*** a machine learning algorithm

Rocchio Method

$$Q_1 = \alpha\ Q_0 + \frac{\beta}{n_1}\sum_{i=1}^{n_1} R_i - \frac{\gamma}{n_2}\sum_{i=1}^{n_2}\ S_i$$

where

Q_0 = the vector for the initial query

R_i = the vector for the relevant document i

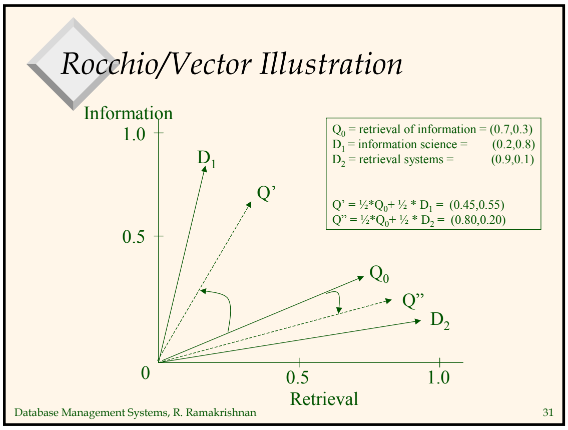
S_i = the vector for the non - relevant document i

n_1 = the number of relevant documents chosen

n_2 = the number of non - relevant documents chosen

α, β and γ tune the importance of relevant and nonrelevant terms

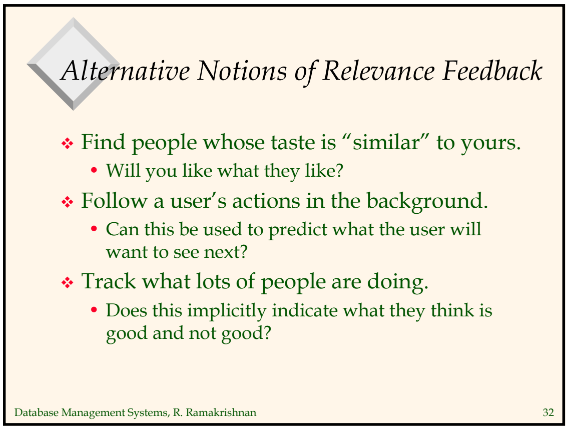
(in some studies best to set β to 0.75 and γ to 0.25)



Rocchio/Vector Illustration

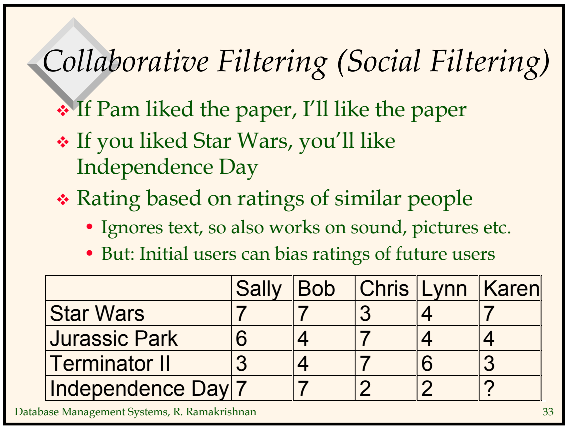
Q_0 = retrieval of information = (0.7,0.3)
D_1 = information science = (0.2,0.8)
D_2 = retrieval systems = (0.9,0.1)

$Q' = ½*Q_0 + ½ * D_1 =$ (0.45,0.55)
$Q'' = ½*Q_0 + ½ * D_2 =$ (0.80,0.20)



Alternative Notions of Relevance Feedback

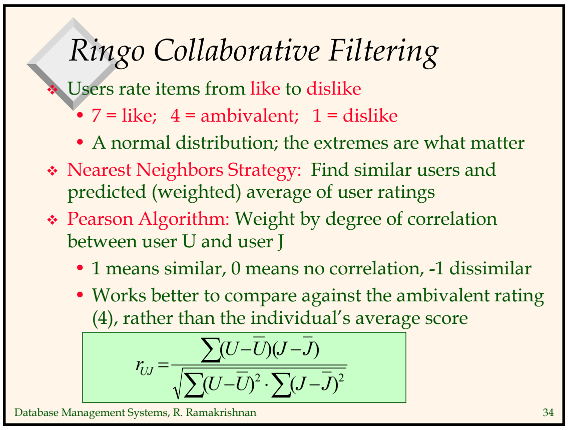
- Find people whose taste is “similar” to yours.
  - Will you like what they like?
- Follow a user’s actions in the background.
  - Can this be used to predict what the user will want to see next?
- Track what lots of people are doing.
  - Does this implicitly indicate what they think is good and not good?



Collaborative Filtering (Social Filtering)

- If Pam liked the paper, I’ll like the paper
- If you liked Star Wars, you’ll like Independence Day
- Rating based on ratings of similar people
  - Ignores text, so also works on sound, pictures etc.
  - But: Initial users can bias ratings of future users

	Sally	Bob	Chris	Lynn	Karen
Star Wars	7	7	3	4	7
Jurassic Park	6	4	7	4	4
Terminator II	3	4	7	6	3
Independence Day	7	7	2	2	?



Ringo Collaborative Filtering

- Users rate items from like to dislike
  - 7 = like; 4 = ambivalent; 1 = dislike
  - A normal distribution; the extremes are what matter
- Nearest Neighbors Strategy: Find similar users and predicted (weighted) average of user ratings
- Pearson Algorithm: Weight by degree of correlation between user U and user J
  - 1 means similar, 0 means no correlation, -1 dissimilar
  - Works better to compare against the ambivalent rating (4), rather than the individual's average score

$$r_{UJ} = \frac{\sum (U - \bar{U})(J - \bar{J})}{\sqrt{\sum (U - \bar{U})^2 \cdot \sum (J - \bar{J})^2}}$$