Term Weighting and Ranking Algorithms

Ray Larson & Marti Hearst

University of California, Berkeley

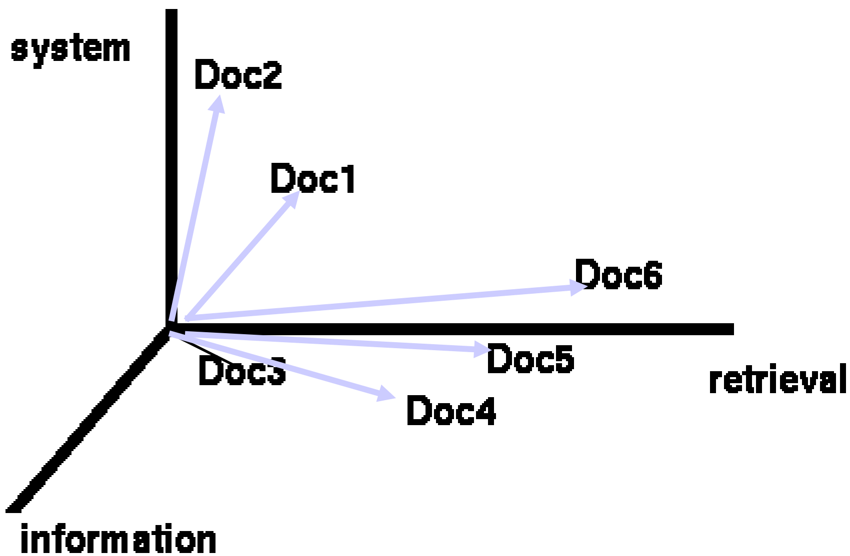
School of Information Management and Systems

SIMS 202: Information Organization and Retrieval: Lecture 17

Review

- Multiple-dimensionality of Document Space
- Automatic Methods for
  - Clustering
  - Creating Thesaurus Terms
- Midterm

Documents in 3D Space

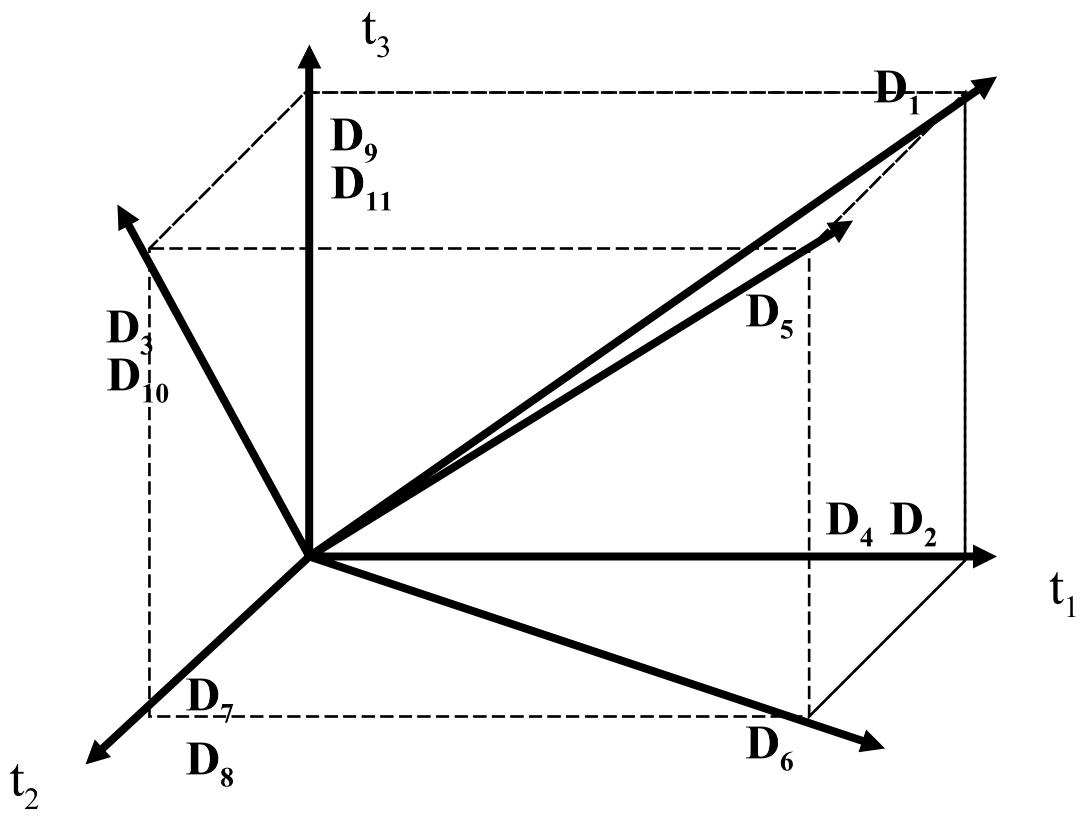


Assumption: Documents that are "close together" in space are similar in meaning.

Vector Space Model

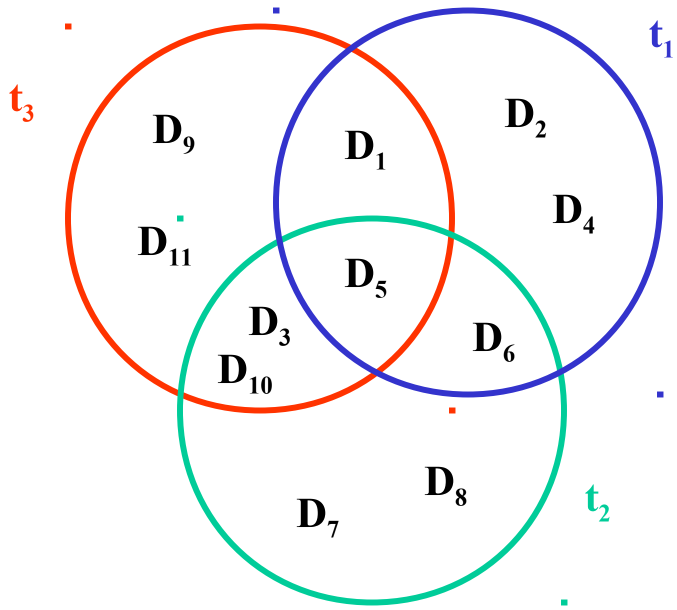
- Documents are represented as vectors in term space
  - Terms are usually stems
  - Documents represented by binary vectors of terms
- Queries represented the same as documents
- Query and Document weights are based on length and direction of their vector
- A vector distance measure between the query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

Documents in Vector Space



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*	



Similarity Measures

$$|Q \cap D|$$

Simple matching (coordination level match)

$$2\frac{|Q \cap D|}{|Q| + |D|}$$

Dice's Coefficient

$$\frac{|Q \cap D|}{|Q \bullet D|}$$

Jaccard's Coefficient

$$\frac{|Q \cap D|}{|Q|^{1/2} \times |D|^{1/2}}$$

Cosine Coefficient

$$\frac{|Q \cap D|}{\min(|Q|, |D|)}$$

Overlap Coefficient

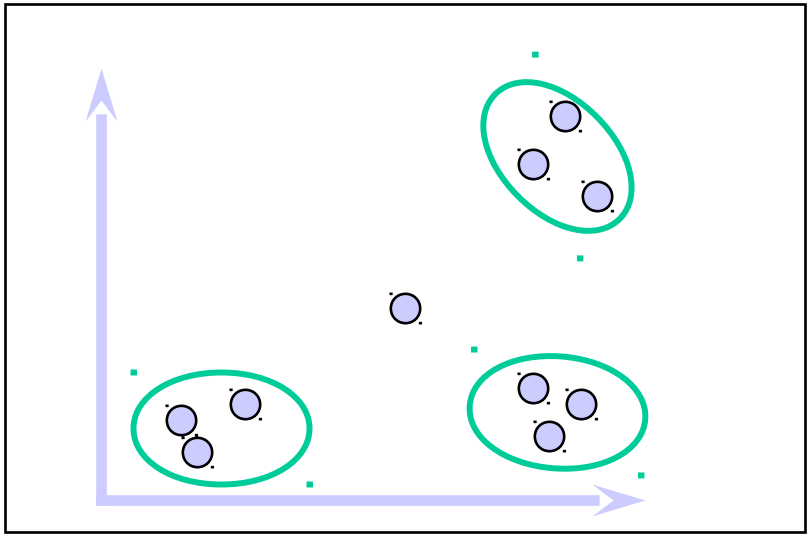
Text Clustering

Clustering is

"The art of finding groups in data."

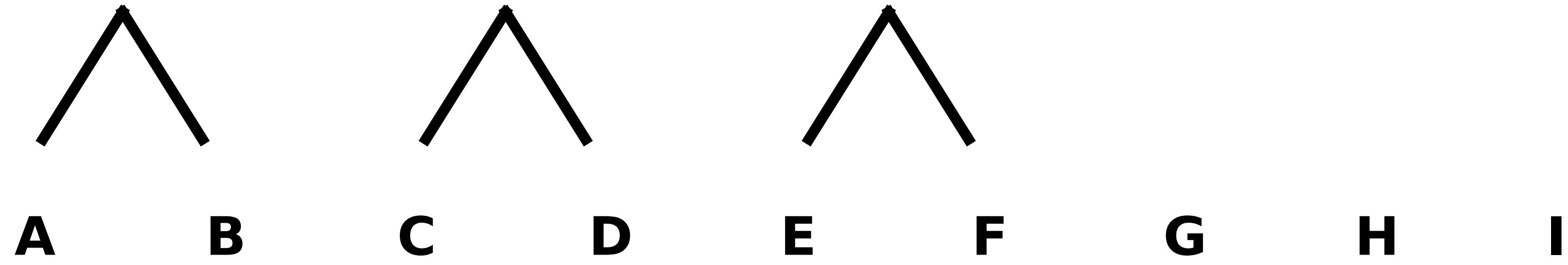
-- Kaufmann and Rousseeu

Term 1

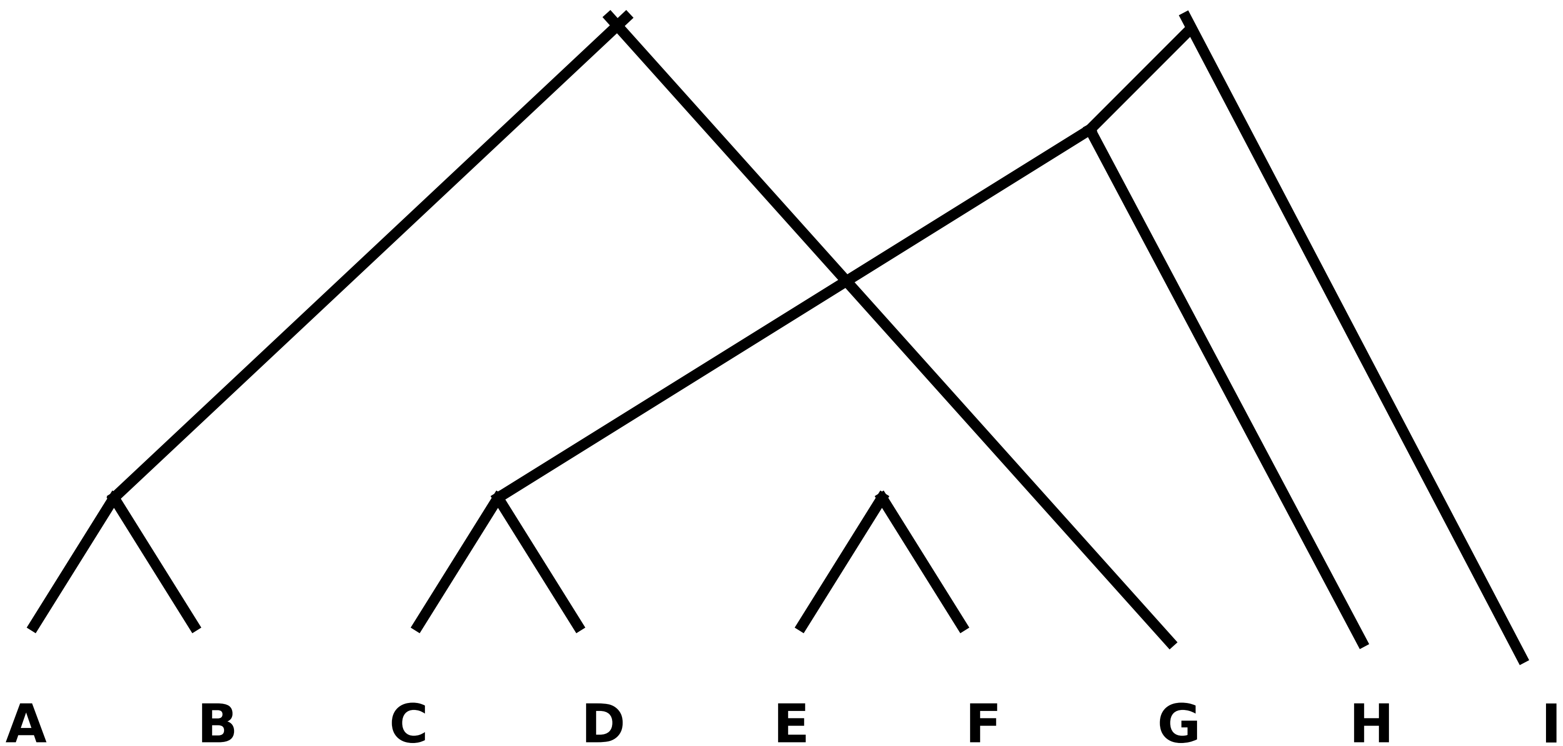


Term 2

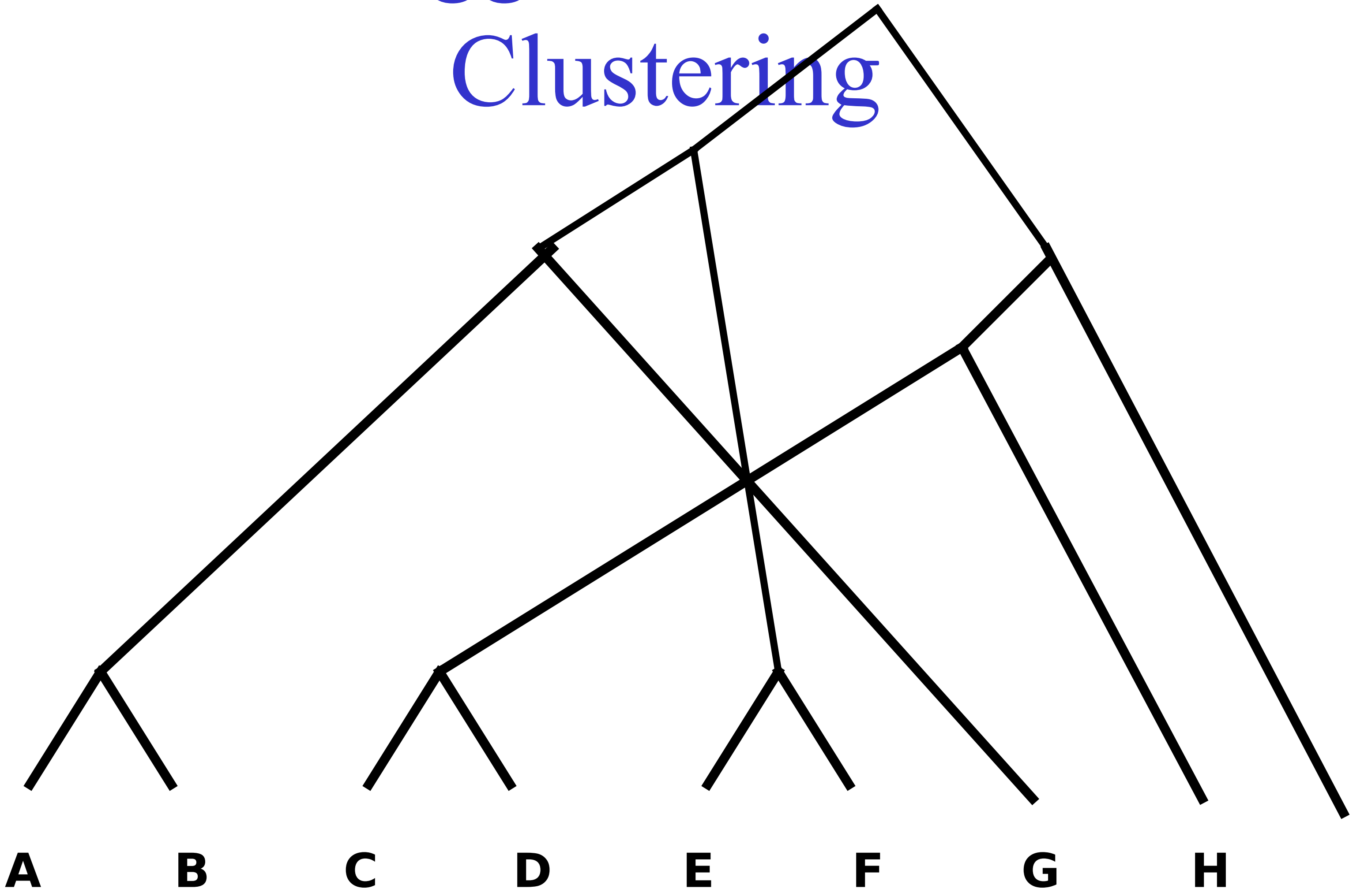
Agglomerative Clustering



Agglomerative Clustering

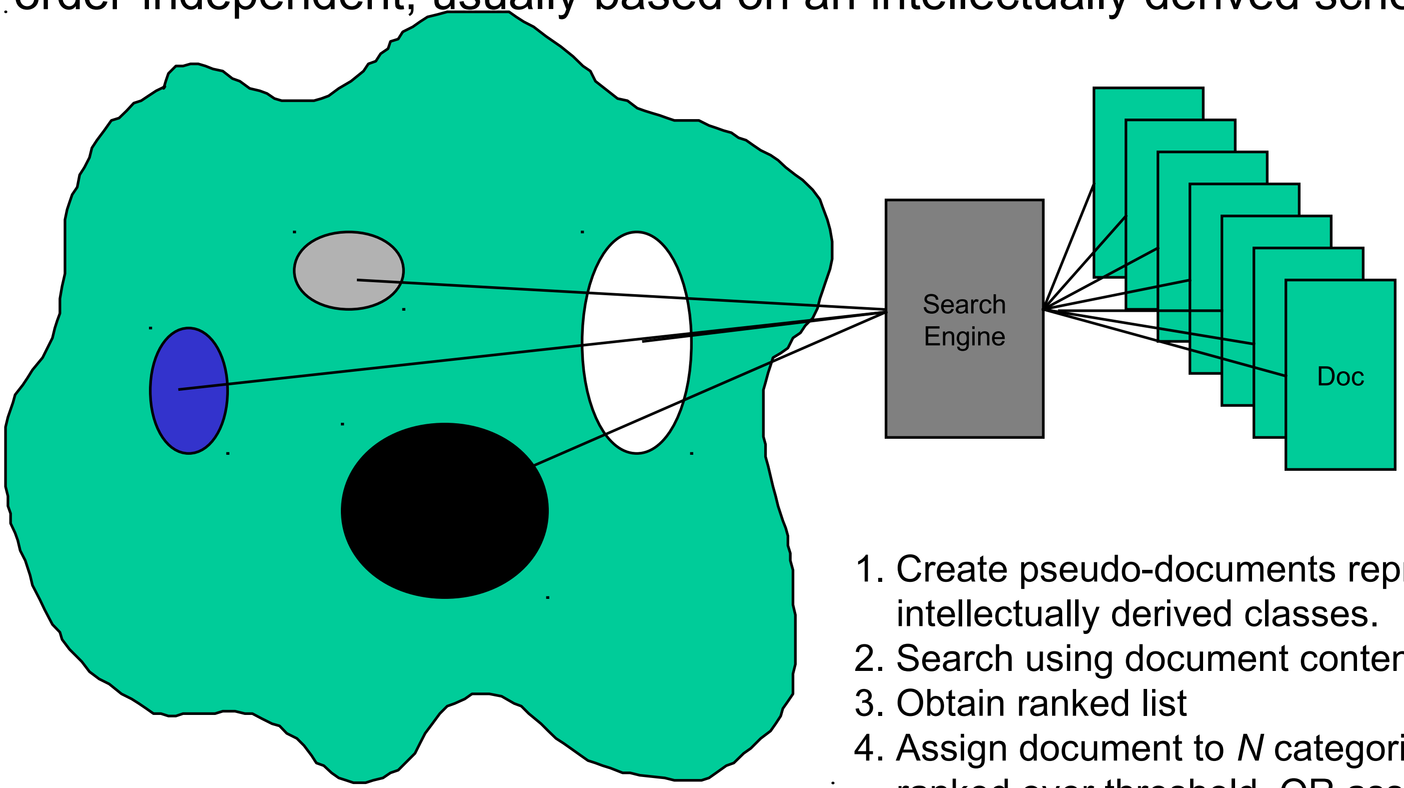


Agglomerative Clustering

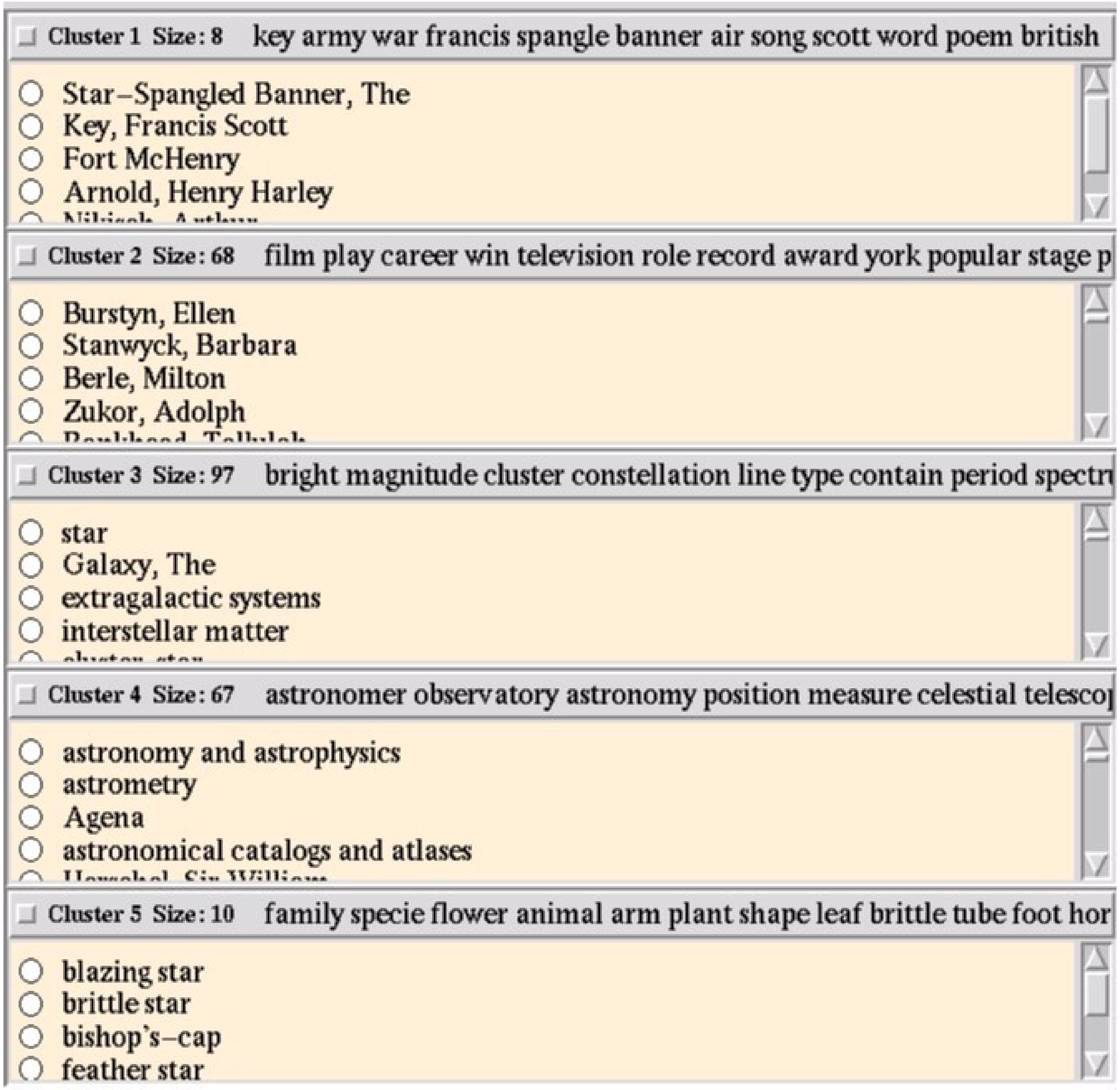


Automatic Class Assignment

Automatic Class Assignment: Polythetic, Exclusive or Overlapping, usually ordered clusters are order-independent, usually based on an intellectually derived scheme



1. Create pseudo-documents representing intellectually derived classes.
2. Search using document contents
3. Obtain ranked list
4. Assign document to *N* categories ranked over threshold. OR assign to top-ranked category


Cluster 1 Size: 8 key army war francis spangle banner air song scott word poem british
Star–Spangled Banner, The
Key, Francis Scott
Fort McHenry
Arnold, Henry Harley
Cluster 2 Size: 68 film play career win television role record award york popular stage p
Burstyn, Ellen
Stanwyck, Barbara
Berle, Milton
Zukor, Adolph
Cluster 3 Size: 97 bright magnitude cluster constellation line type contain period spectr
star
Galaxy, The
extragalactic systems
interstellar matter
Cluster 4 Size: 67 astronomer observatory astronomy position measure celestial telesco
astronomy and astrophysics
astrometry
Agena
astronomical catalogs and atlases
Cluster 5 Size: 10 family specie flower animal arm plant shape leaf brittle tube foot hor
blazing star
brittle star
bishop's–cap
feather star

Today

- Document Ranking
  - term weights
  - similarity measures
    - vector space model
    - probabilistic models

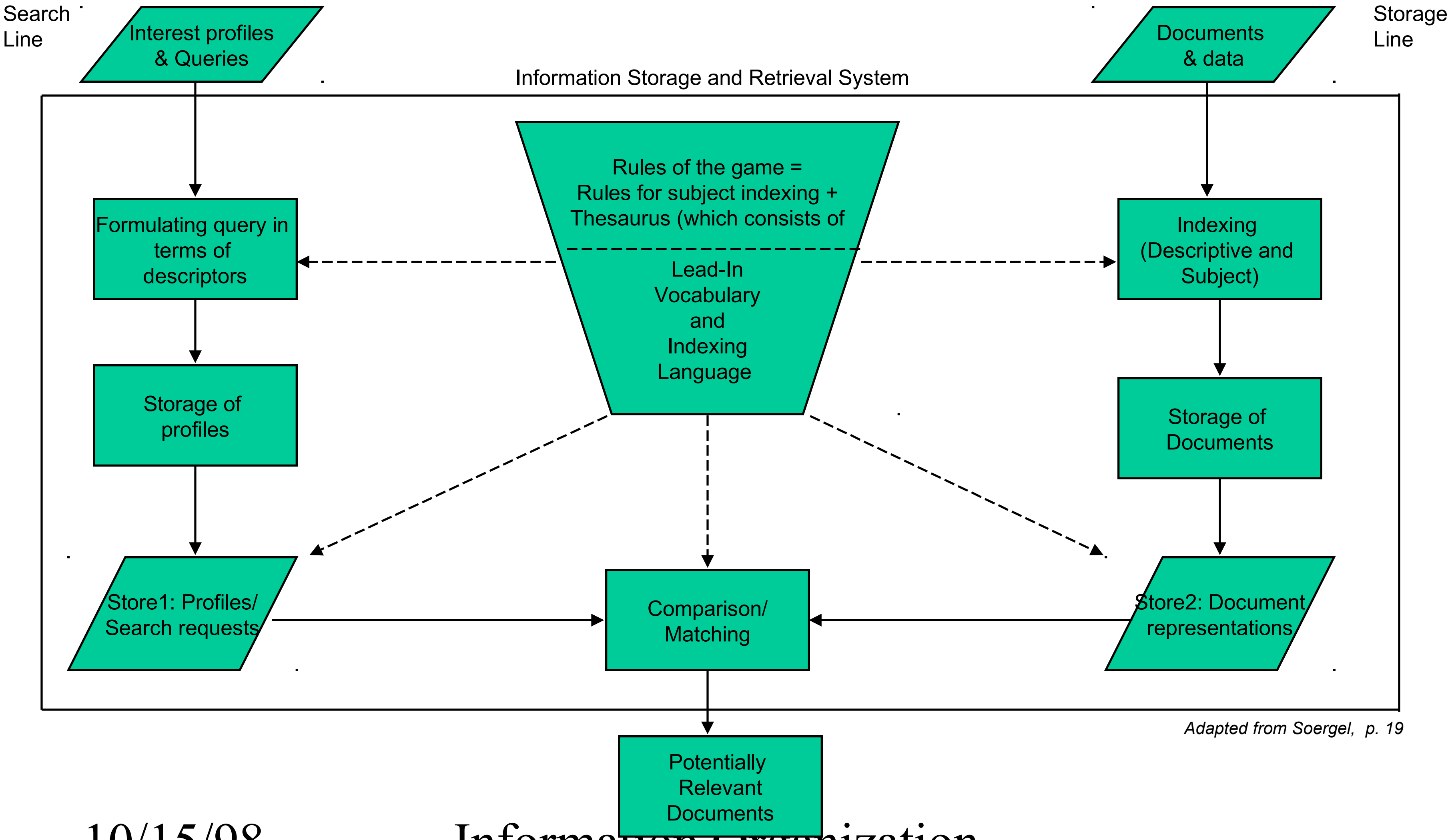
Finding Out About

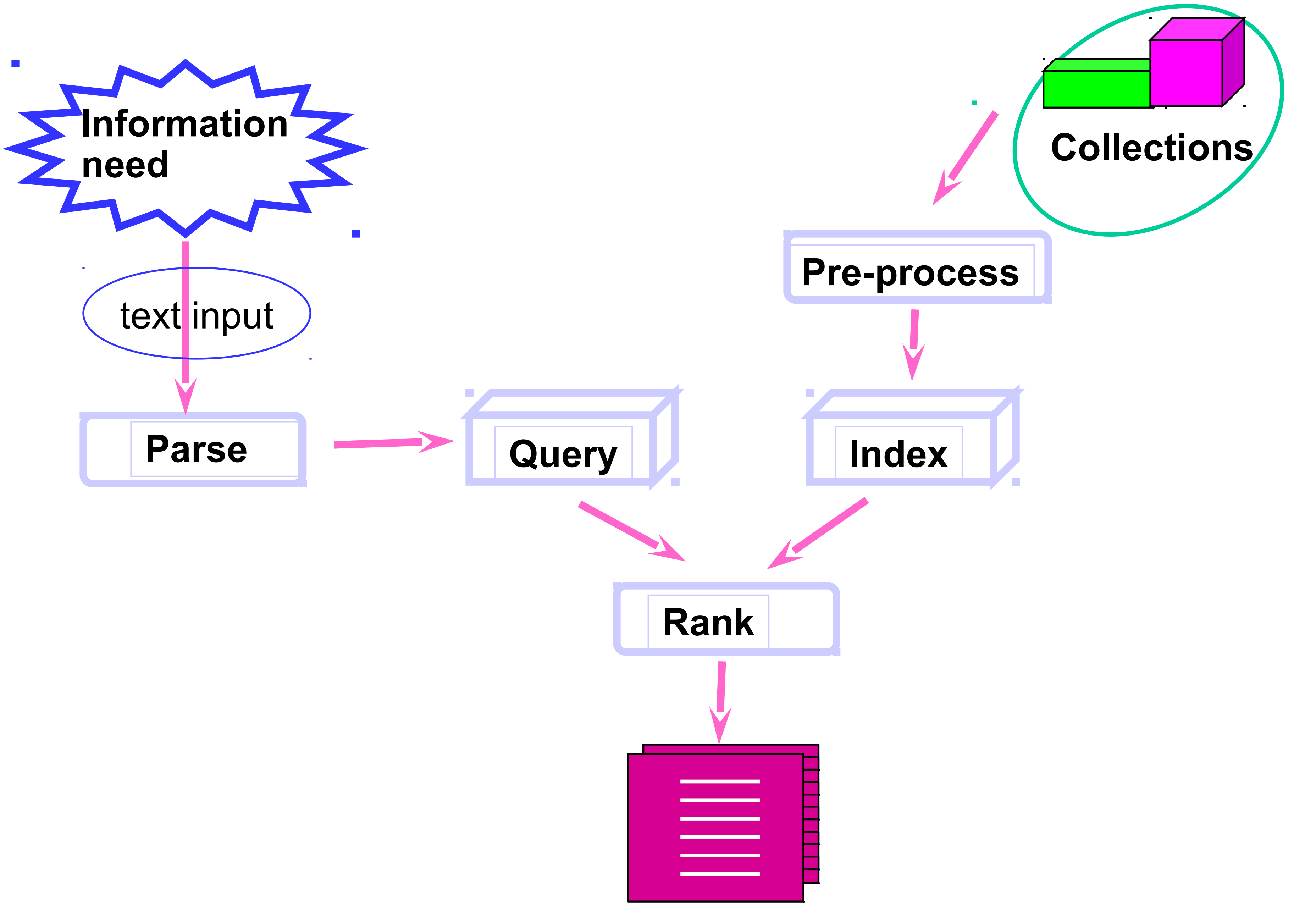
- Three phases:
  - Asking of a question
  - Construction of an answer
  - Assessment of the answer
- Part of an iterative process

Ranking Algorithms

- Assign weights to the terms in the query.
- Assign weights to the terms in the documents.
- Compare the weighted query terms to the weighted document terms.
- Rank order the results.

Structure of an IR System




Information need
text input
Parse
Query
Collections
Pre-process
Index
Rank

Vector Representation

(revisited; see Salton article in *Science*)

- Documents and Queries are represented as vectors.
- Position 1 corresponds to term 1, position 2 to term 2, position t to term t
- The weight of the term is stored in each position

$$D_i = w_{d_{i1}}, w_{d_{i2}}, \ldots, w_{d_{it}}$$

$$Q = w_{q1}, w_{q2}, \ldots, w_{qt}$$

$$w = 0 \text{ if a term is absent}$$

Assigning Weights to Terms

- Binary weights
- Raw term frequency
- tf x idf
- Automatically-derived thesaurus terms

Assigning Weights to Terms

- Binary Weights
- Raw term frequency
- 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
- Automatically derived thesaurus terms

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

Assigning 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

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

$$T_k = \text{term } k \text{ in document } D_i$$

$$tf_{ik} = \text{frequency of term } \mathrm{T}_k \text{ in document } D_i$$

$$idf_k = \text{inverse document frequency of term } \mathrm{T}_k \text{ in } C$$

$$N = \text{total number of documents in the collection } C$$

$$n_k = \text{the number of documents in } C \text{ that contain } \mathrm{T}_k$$

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

Inverse Document Frequency

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

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

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

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

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

tf x idf normalization

- Normalize the term weights (so longer documents are not unfairly given more weight)
  - normalize usually means force all values to fall within a certain range, usually between 0 and 1, inclusive.

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

Vector space similarity
(use the weights to compare the documents)

Now, the similarity of two documents is :

$$sim(D_i, D_j) = \sum_{k=1}^{t} w_{ik} * w_{jk}$$

This is also called the cosine, or normalized inner product.

(Normalization was done when weighting the terms.)

Vector Space Similarity Measure

combine tf x idf into a similarity measure

$$D_i = w_{d_{i1}}, w_{d_{i2}}, \ldots, w_{d_{it}}$$

$$Q = w_{q1}, w_{q2,} \ldots, w_{qt} \qquad w = 0 \text{ if a term is absent}$$

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

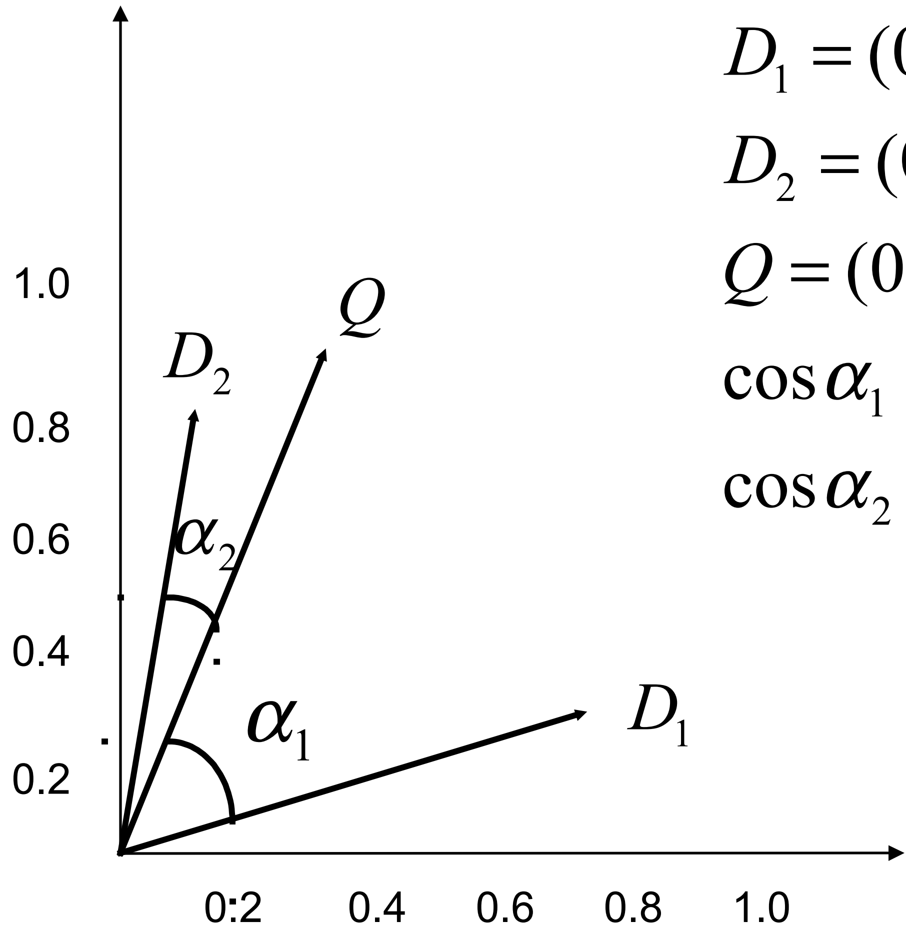
otherwise normalize in the similarity comparison:

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

To Think About

- How does this ranking algorithm behave?
  - Make a set of hypothetical documents consisting of terms and their weights
  - Create some hypothetical queries
  - How are the documents ranked, depending on the weights of their terms and the queries' terms?

Computing Similarity Scores



$$D_1 = (0.8, 0.3)$$

$$D_2 = (0.2, 0.7)$$

$$Q = (0.4, 0.8)$$

$$\cos\alpha_1 = 0.74$$

$$\cos\alpha_2 = 0.98$$

Computing a similarity score

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

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

What does their similarity comparison yield?

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

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

Other Major Ranking Schemes

- Probabilistic Ranking
  - Attempts to be more theoretically sound than the vector space (v.s.) model
    - try to predict the probability of a document's being relevant, given the query
    - there are many many variations
    - usually more complicated to compute than v.s.
    - usually many approximations are required
  - Works about the same (sometimes better) than vector space approaches

Other Major Ranking Schemes

- Staged Logistic Regression
  - A variation on probabilistic ranking
  - Used successfully here at Berkeley in the Cheshire II system

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)
- Rely on accurate estimates of probabilities

Probabilistic Models: Some Notation

- $\mathbf{D}$ = All present and future documents
- $\mathbf{Q}$ = All present and future queries
- (D_i, Q_j) = A document query pair
- $\mathbf{x}$ = class of similar documents, $\mathbf{x} \subseteq \mathbf{D}$
- $\mathbf{y}$ = class of similar queries, $\mathbf{y} \subseteq \mathbf{Q}$
- Relevance is a relation:

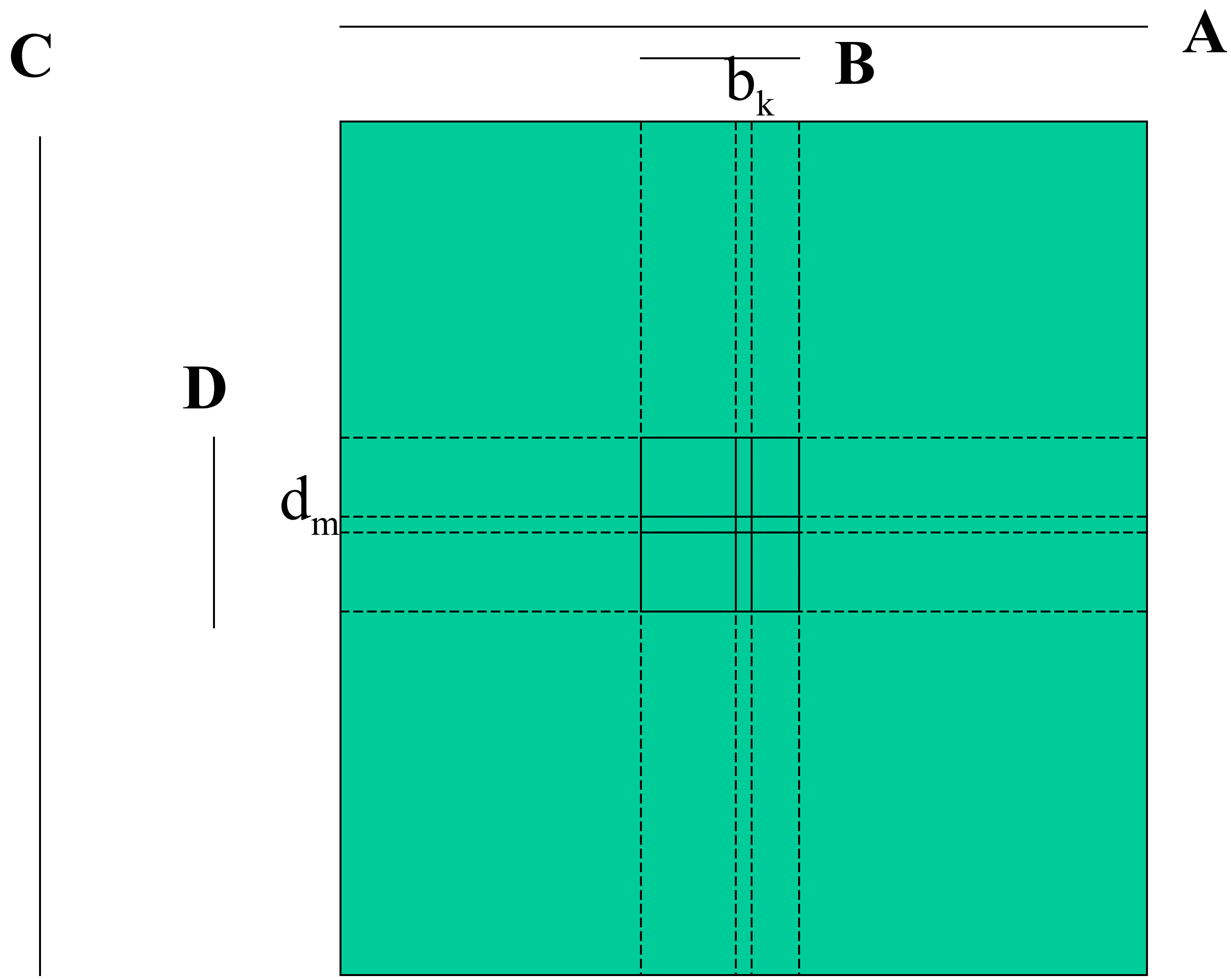
  $R = \{(D_i, Q_j) \mid D_i \in \mathbf{D}, Q_j \in \mathbf{Q}, \text{document } D_i \text{ is}$

  $\text{judged relevant by the user submitting } Q_j\}$

Probabilistic Models

- Model 1 -- Probabilistic Indexing, $P(\mathbf{R}|\mathbf{y},D_i)$
- Model 2 -- Probabilistic Querying, $P(\mathbf{R}|Q_j,\mathbf{x})$
- Model 3 -- Merged Model, $P(\mathbf{R}|\, Q_j, D_i)$
- Model 0 -- $P(\mathbf{R}|\mathbf{y},\mathbf{x})$
- Probabilities are estimated based on prior usage or relevance estimation

Probabilistic Models



Logistic Regression

- Based on work by William Cooper, Fred Gey and Daniel Dabney.
- Builds a regression model for relevance prediction based on a set of training data
- Uses less restrictive independence assumptions than Model 2
  - Linked Dependence

Probabilistic Models: Logistic Regression

- Estimates for relevance based on log-linear model with various statistical measures of document content as independent variables.

Log odds of relevance is a linear function of attributes:

$$\log O(R|q_i, d_j, t_k) = c_0 + c_1 v_1 + c_2 v_2 + \ldots + c_n v_n$$

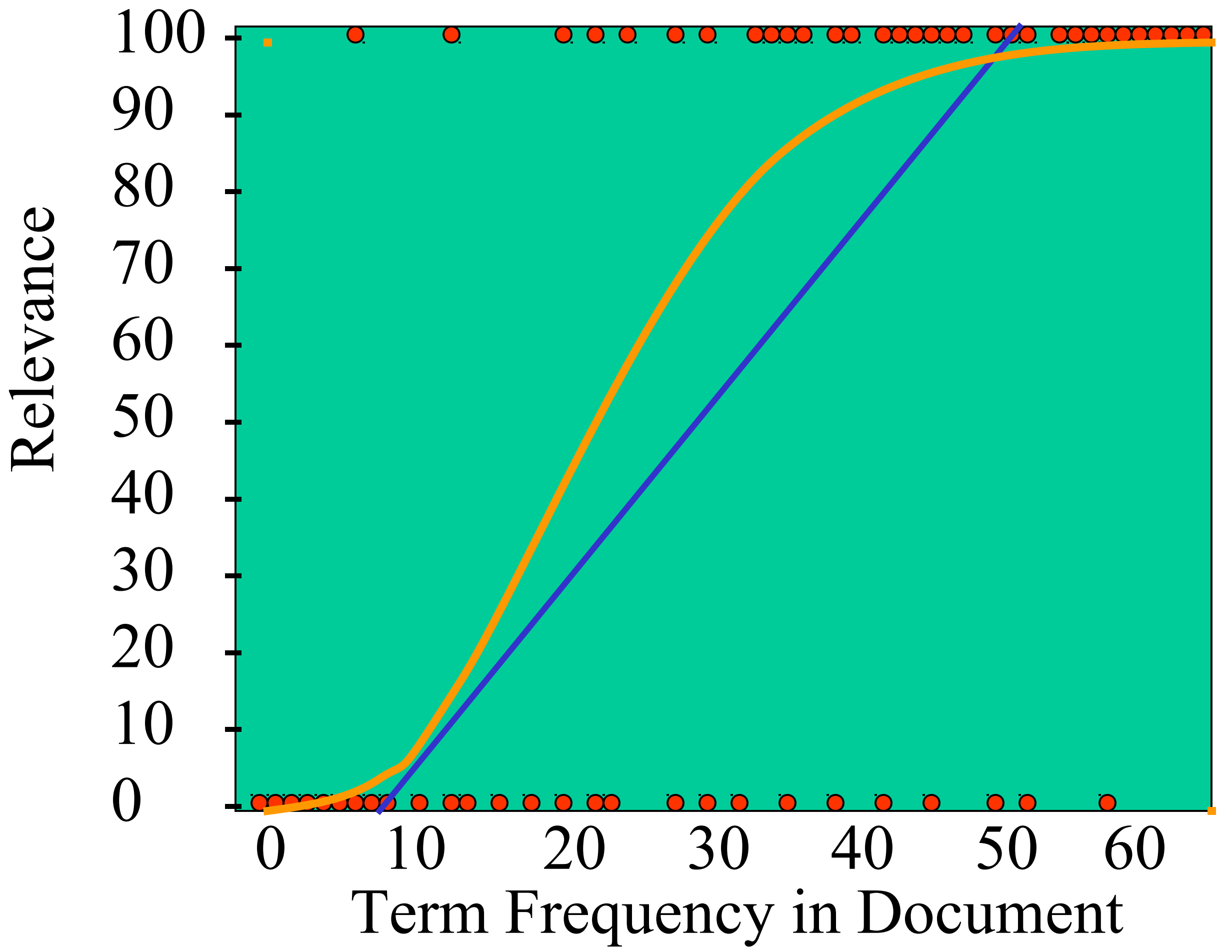
Term contributions summed:

$$\log O(R \mid q_i, d_j) = \sum_{k=1}^{m} [\log O(R \mid q_i, d_j, t_k) - \log O(R)]$$

Probability of Relevance is inverse of log odds:

$$P(R \mid q_i, d_j) = \frac{1}{1 + e^{-\log(O(R|q_i, d_j))}}$$

Logistic Regression



Probabilistic Models: Logistic Regression attributes

$X_1 = \frac{1}{M}\sum_{1}^{M} \log QAF_{t_j}$ Average Absolute Query Frequency

$X_2 = \sqrt{QL}$ Query Length

$X_3 = \frac{1}{M}\sum_{1}^{M} \log DAF_{t_j}$ Average Absolute Document Frequency

$X_4 = \sqrt{DL}$ Document Length

$X_5 = \frac{1}{M}\sum_{1}^{M} \log IDF_{t_j}$ Average Inverse Document Frequency

$IDF = \frac{N - n_{t_j}}{n_{t_j}}$ Inverse Document Frequency

$X_6 = \log M$ Number of Terms in common between query and document -- logged

Probabilistic Models: Logistic Regression

Probability of relevance is based on Logistic regression from a sample set of documents to determine values of the coefficients.

At retrieval the probability estimate is obtained by:

$$P(R \mid Q, D) = c_0 + \sum_{i=1}^{6} c_i X_i$$

For the 6 *X* attribute measures shown previously

Simplified Logistic Regression

- Pick a set of X feature types
  - sum of frequencies of all terms in query x1
  - sum of frequencies of all query terms in document x2
  - query length x3
  - document length x4
  - sum of idf's for all terms in query x5
- Determine weights, c, to indicate how important each feature type is (use training examples)
- To assign a score to the document:
  - add up the feature weight times the term weight for each feature and each term in the query

$$score(D,Q) \approx \sum_{i=1}^{5} c_i x_i$$

Probabilistic Models

Advantages

- Strong theoretical basis
- In principle should supply the best predictions of relevance given available information
- Can be implemented similarly to Vector

Disadvantages

- Relevance information is required -- or is "guestimated"
- Important indicators of relevance may not be term -- though terms only are usually used
- Optimally requires on-going collection of relevance information

Vector and Probabilistic Models

- Support “natural language” queries
- Treat documents and queries the same
- Support relevance feedback searching
- Support ranked retrieval
- Differ primarily in theoretical basis and in how the ranking is calculated
  – Vector assumes relevance
  – Probabilistic relies on relevance judgments or estimates