Chapter 7: Clustering (Unsupervised Data Organization)

7.1 Hierarchical Clustering

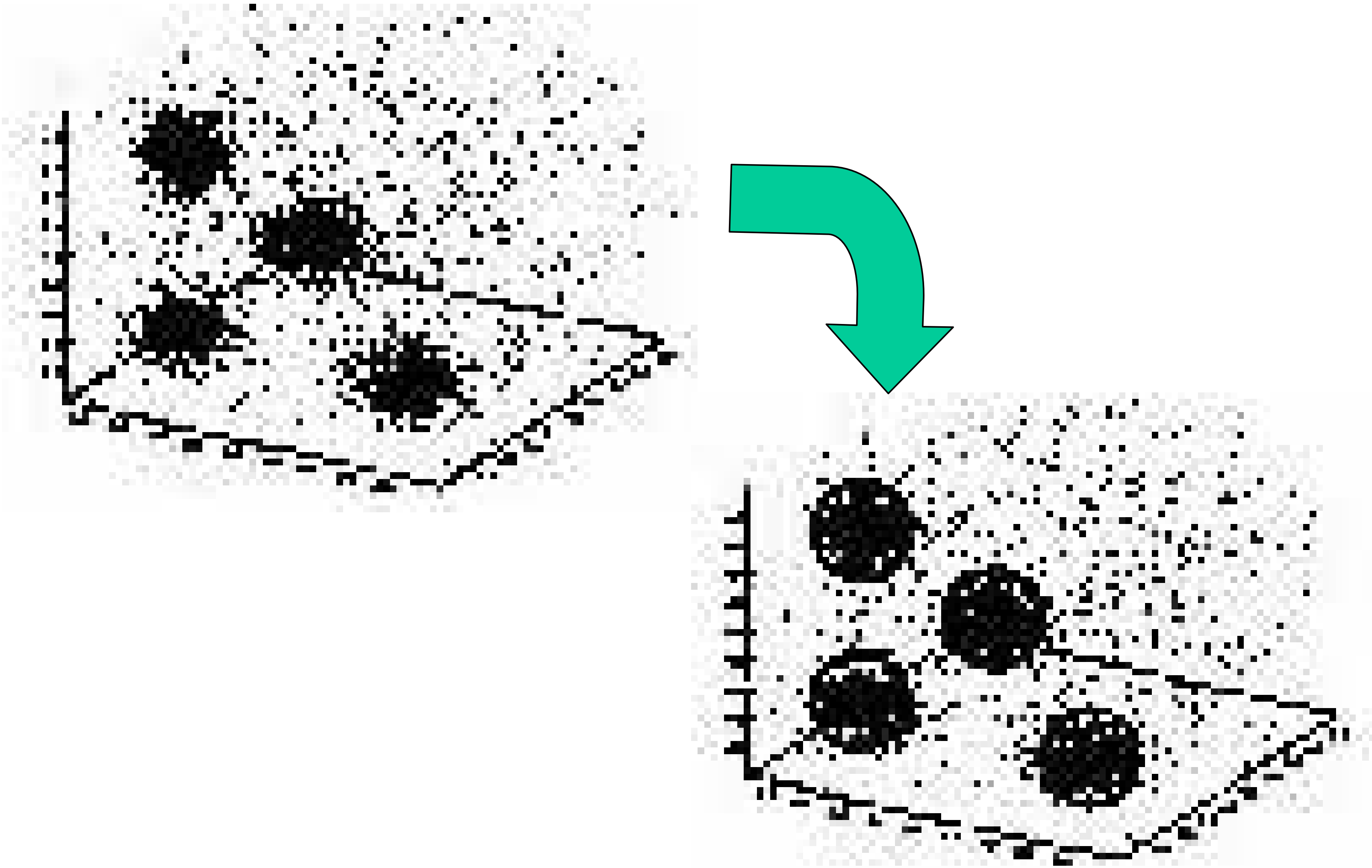
7.2 Flat Clustering

7.3 Embedding into Vector Space for Visualization

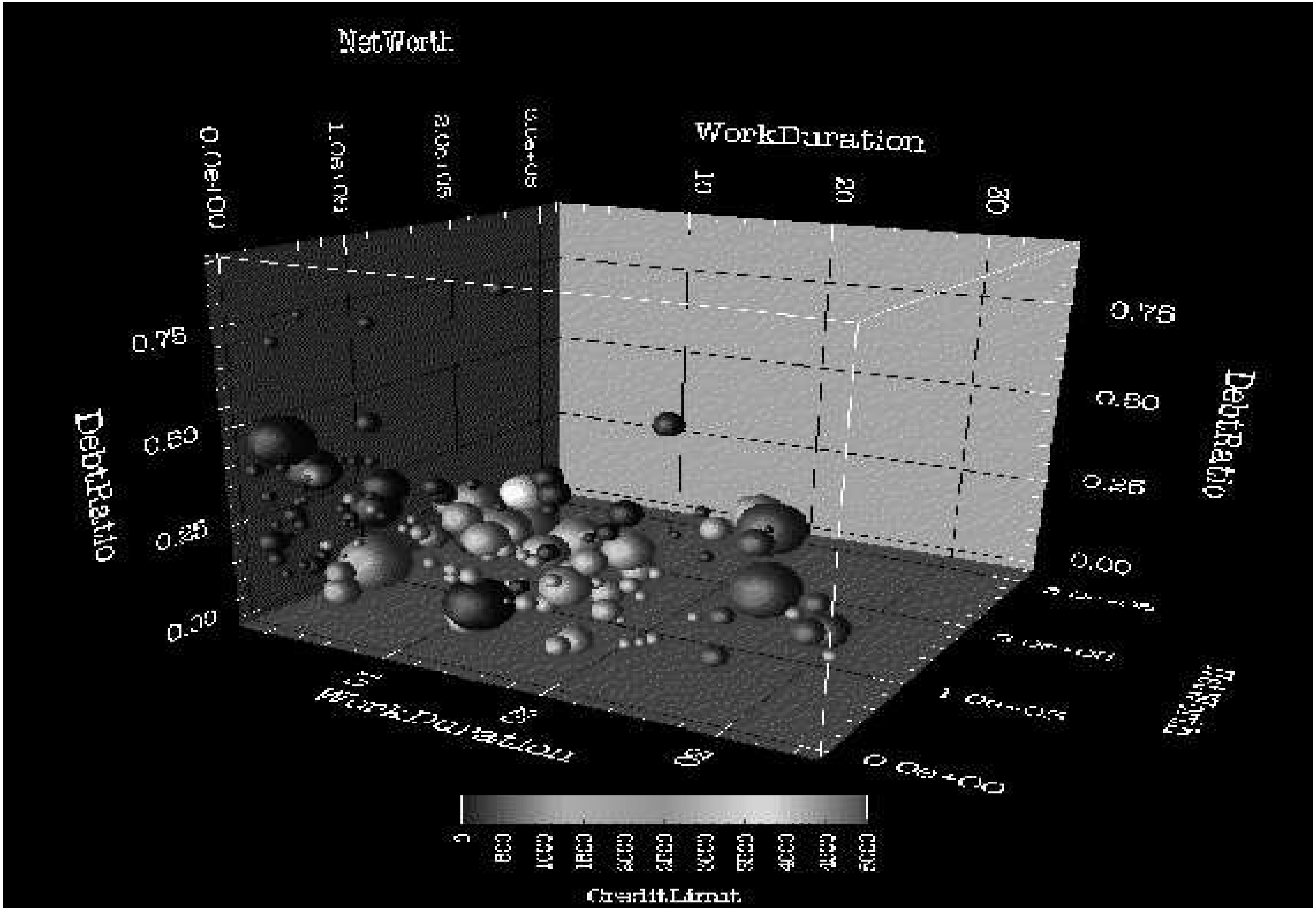
7.4 Applications

Clustering: *unsupervised grouping (partitioning) of objects into classes (clusters) of similar objects*

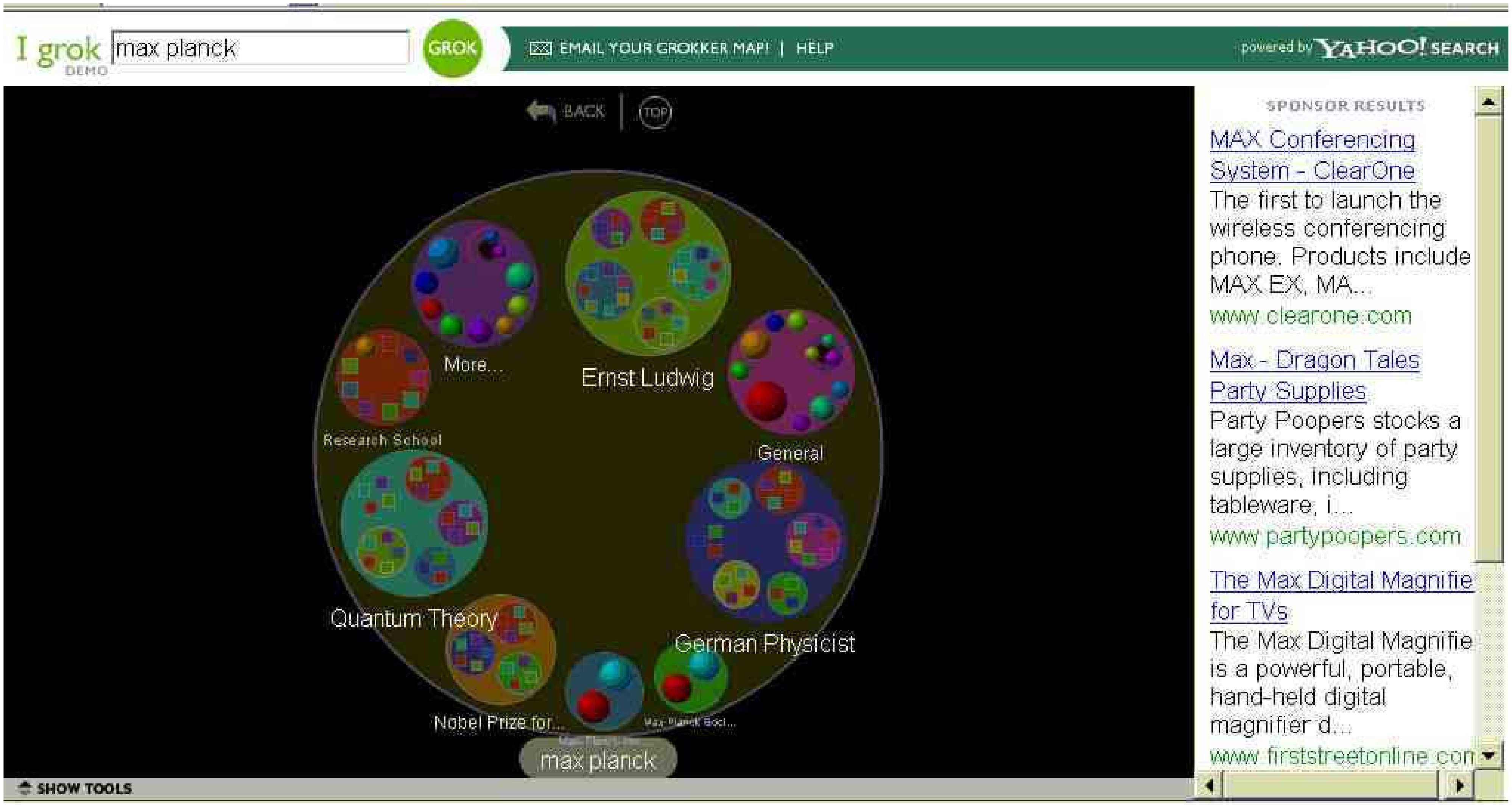
Clustering Example 1



Clustering Example 2

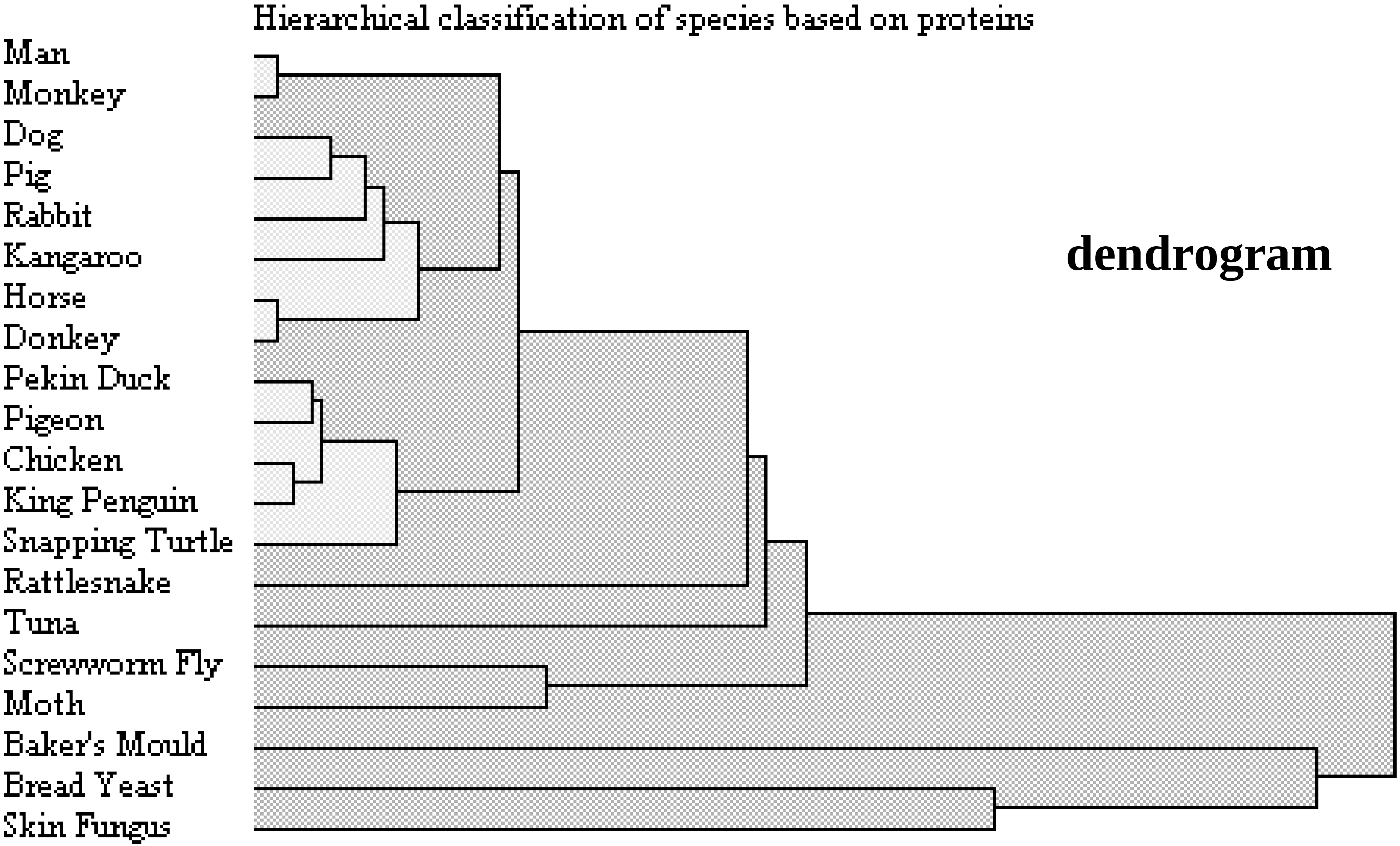


Clustering Search Results for Visualization and Navigation



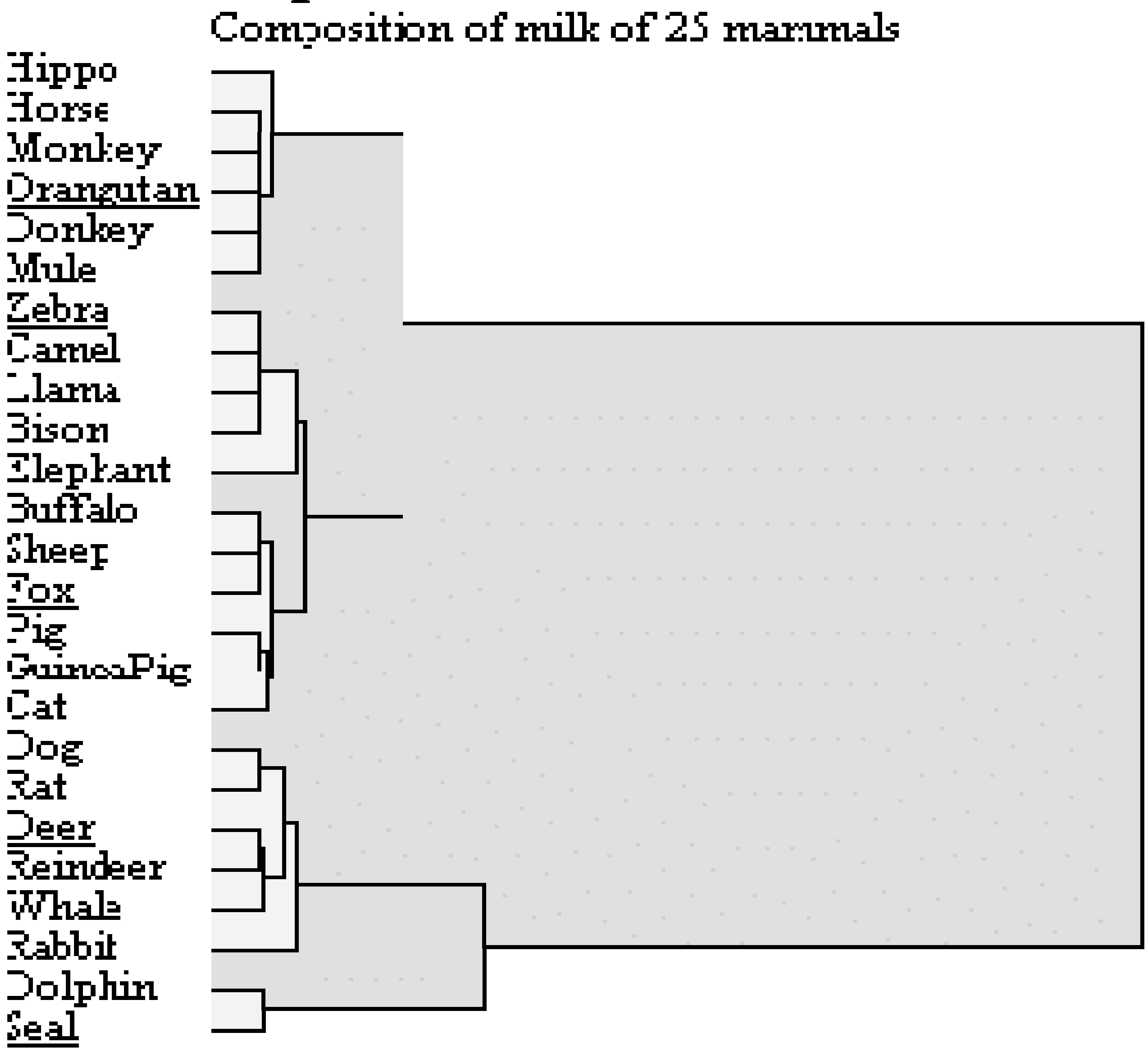
http://www.grokker.com/

Example for Hierarchical Clustering

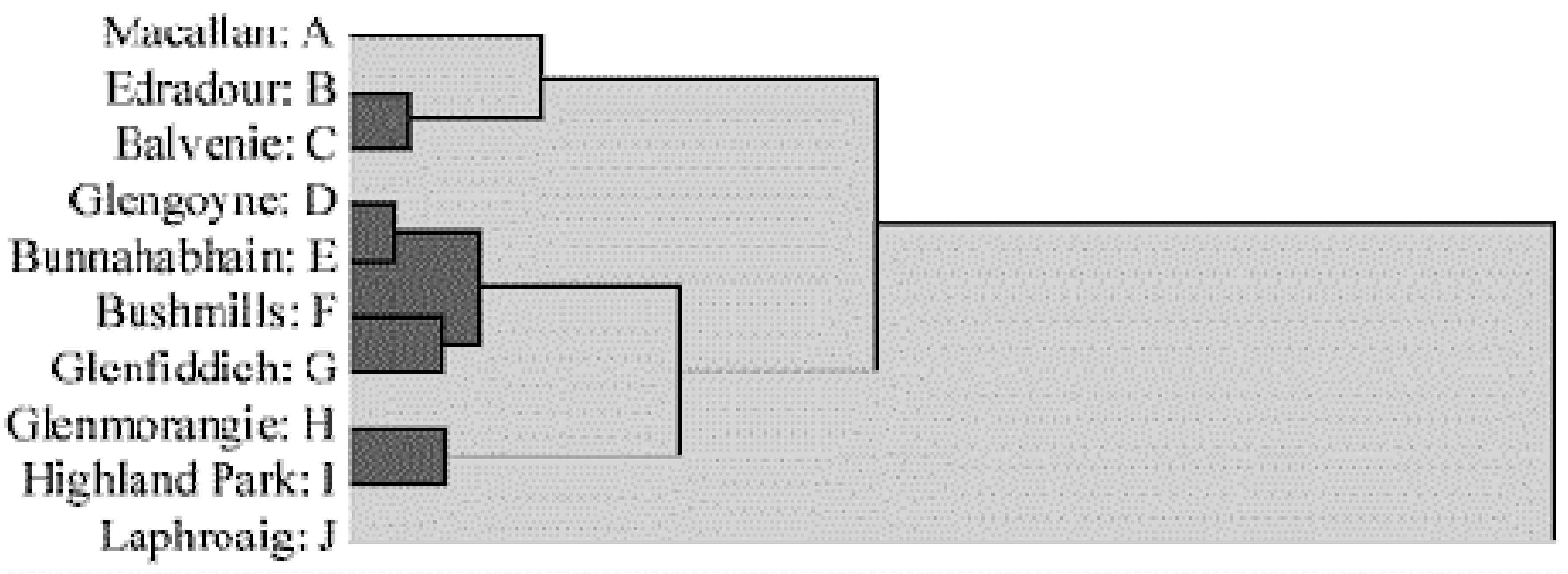


dendrogram

Example for Hierarchical Clustering



Example for Hierarchical Clustering



Clustering: Classification based on Unsupervised Learning

given:

n ***m-dimensional data records*** dj $\in$ D $\subseteq$ dom(A1) $\times$ ... $\times$ dom(Am)

with attributes Ai (e.g. term frequency vectors $\subseteq N_0 \times ... \times N_0$)

or n ***data points*** with pair-wise ***distances (similarities)*** in a ***metric space***

wanted:

k **clusters** c1, ..., ck and an assignment D $\rightarrow$ {c1, ..., ck} such that the

average **intra-cluster similarity** $\frac{1}{k}\sum_k \left(\frac{1}{|c_k|} \sum_{\vec{d} \in c_k} sim(\vec{d}, \vec{c}_k) \right)$

is high and

the average **inter-cluster similarity** $\frac{1}{k(k-1)} \sum_{\substack{i,j \\ i \neq j}} sim(\vec{c}_i, \vec{c}_j)$

is low,

where the ***centroid*** $\vec{c}_k$ of ck is: $\vec{c}_k = \frac{1}{|c_k|} \sum_{\vec{d} \in c_k} \vec{d}$

Desired Clustering Properties

A clustering function f_d maps a dataset D onto a partitioning $\Gamma \subseteq 2^D$ of D, with pairwise disjoint members of Γ and $\cup_{x \in D} f(x) = D$, based on a (metric or non-metric) distance function $d: D \times D \rightarrow R_0^+$ which is symmetric and satisfies $d(x,y)=0 \Leftrightarrow x=y$

Axiom 1: Scale-Invariance

For any distance function d and any $\alpha>0$: $f_d(x) = f_{\alpha d}(x)$ for all $x \in D$

Axiom 2: Richness (Expressiveness)

For every possible partitioning Γ of D there is a distance function d such that f_d produces Γ

Axiom 3: Consistency

d is a Γ-transformation of d if for all x,y in same $S \in \Gamma$: $d'(x,y) \le d(x,y)$ and for all x, y in different S, $S' \in \Gamma$: $d'(x,y) \ge d(x,y)$.
If f_d produces Γ then $f_{d'}$ produces Γ, too.

Impossibility Theorem (J. Kleinberg: NIPS 2002):

For each dataset D with $|D| \ge 2$ there is no clustering function f that satisfies Axioms 1,2, and 3 for every possible choice of d

Hierarchical vs. Flat Clustering

Hierarchical Clustering:
- detailed and insightful
- hierarchy built
  in natural manner
  from fairly simple algorithms
- relatively expensive
- no prevalent algorithm

Flat Clustering:
- data overview & coarse analysis
- level of detail depends
  on the choice of the
  number of clusters
- relatively efficient
- K-Means and EM are simple
  standard algorithms

7.1 Hierarchical Clustering: Agglomerative Bottom-up Clustering (HAC)

Principle:

- start with each d_i forming its own singleton cluster c_i
- in each iteration combine the most similar clusters c_i, c_j into a new, single cluster

for i:=1 to n do c_i := {d_i} od;
C := {c_1, ..., c_n}; /* set of clusters */
while |C| > 1 do
   determine $c_i, c_j \in C$ with maximal inter-cluster similarity;
   $C := C - \{c_i, c_j\} \cup \{c_i \cup c_j\}$;
od;

Divisive Top-down Clustering

Principle:

- start with a single cluster that contains all data records
- in each iteration identify the least „coherent“ cluster and divide it into two new clusters

$c_1 := \{d_1, ..., d_n\}$;
$C := \{c_1\}$; /* set of clusters */
while there is a cluster $c_j \in C$ with $|c_j|>1$ do
  determine c_i with the lowest intra-cluster similarity;
  partition c_i into c_{i1} and c_{i2} (i.e. $c_i = c_{i1} \cup c_{i2}$ and $c_{i1} \cap c_{i2} = \emptyset$)
  such that the inter-cluster similarity between c_{i1} and c_{i2}
  is minimized;
od;

For partitioning a cluster one can use another clustering method (e.g. a bottom-up method)

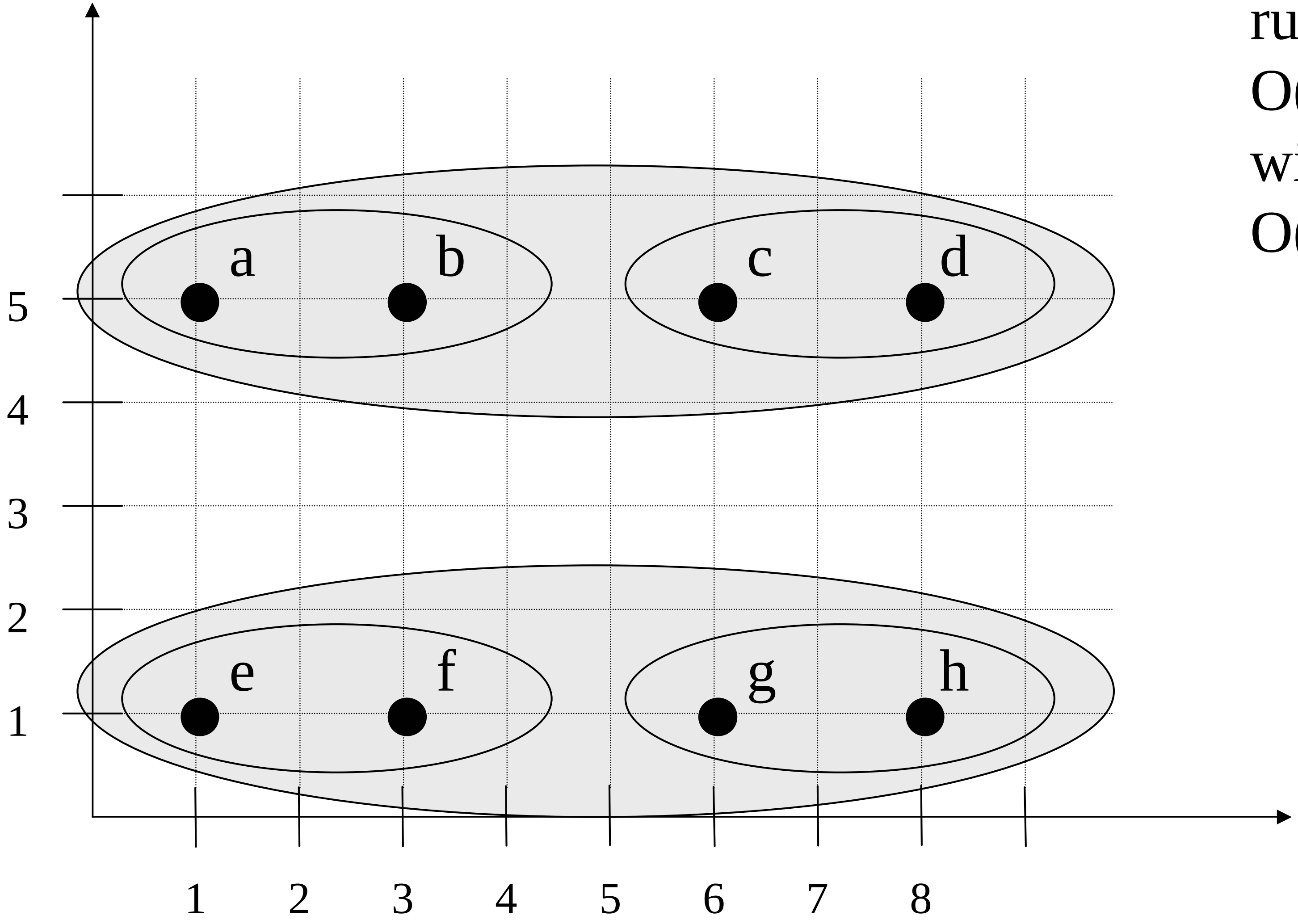
Alternative Similarity Metrics for Clusters

given: similarity on data records - sim: $D \times D \rightarrow R$ oder [0,1]
define: similarity between clusters – sim: $2^D \times 2^D \rightarrow R$ or [0,1]

Alternatives:
- **Centroid method**: sim (c,c‘) = sim(d, d‘) with centroid d of c and centroid d‘ of c‘
- **Single-Link method**: sim(c,c‘) = sim(d, d‘) with $d \in c$, $d' \in c'$, such that d and d‘ have the highest similarity
- **Complete-Link method**: sim(c,c‘) = sim(d, d‘) with $d \in c$, $d' \in c'$, such that d and d‘ have the lowest similarity
- **Group-Average method**: $$\frac{1}{|c| \cdot |c'|} \sum_{d \in c, d' \in c'} sim(d, d')$$

For hierarchical clustering the following axiom must hold:
max {sim(c,c‘), sim(c,c‘‘)} $\geq$ sim(c, c‘ $\cup$ c‘‘) for all c, c‘, c‘‘ $\in 2^D$

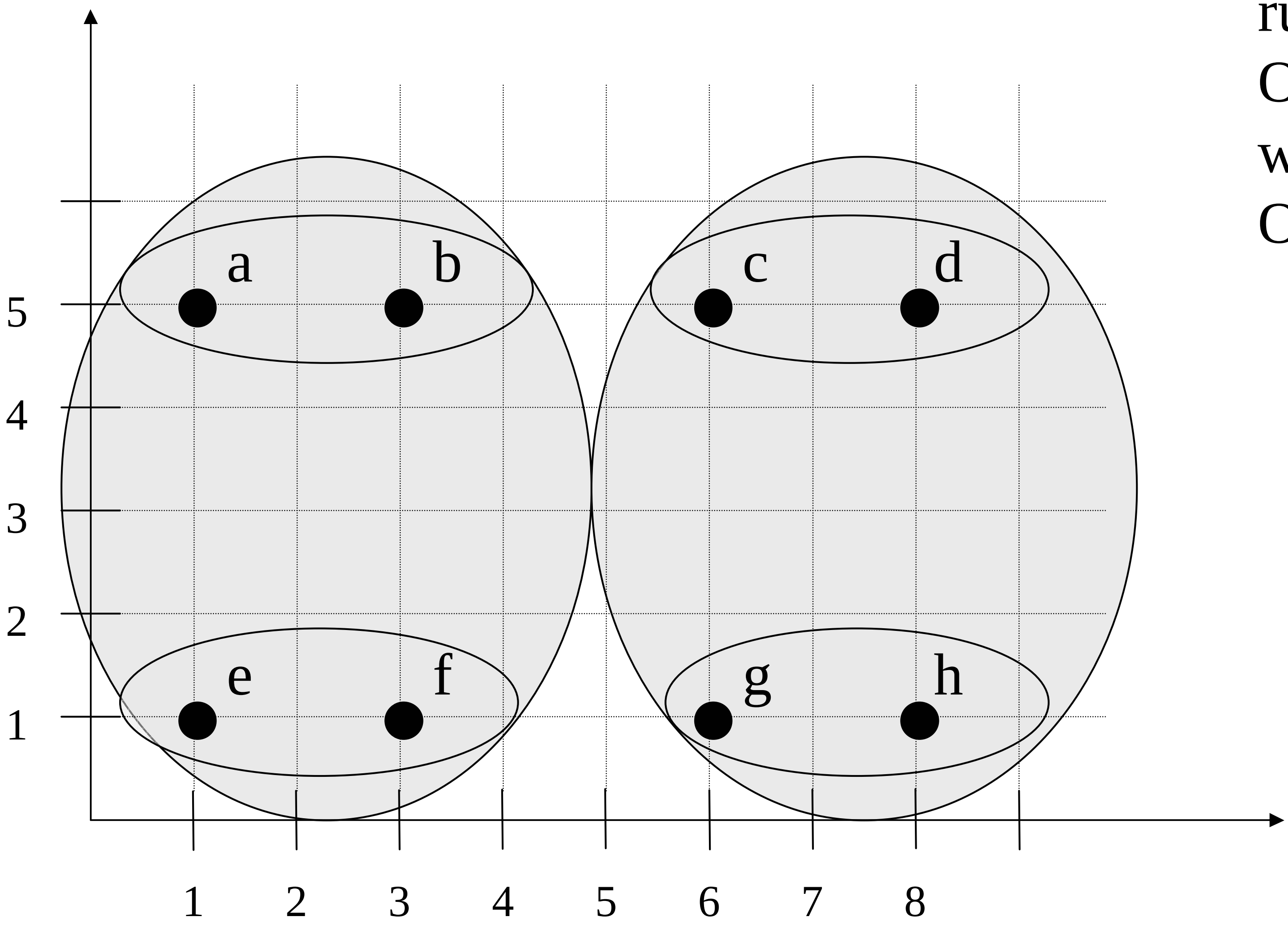
Example for Bottom-up Clustering with Single-Link Metric (Nearest Neighbor)



run-time:
$O(n^2)$
with space
$O(n^2)$

emphasizes „local" cluster coherence (chaining effect)
→ tendency towards long clusters

Example for Bottom-up Clustering with Complete-Link Metric (Farthest Neighbor)



run-time: $O(n^2 \log n)$ with space $O(n^2)$

emphasizes „global“ cluster coherence
→ tendency towards round clusters with small diameter

Relationship to Graph Algorithms

Single-Link clustering:

- corresponds to construction of **maximum (minimum) spanning tree** for undirected, weighted graph G = (V,E) with V=D, E=D×D and edge weight sim(d,d‘) (dist(d,d‘)) for (d,d‘)∈ E
- from the maximum spanning tree the cluster hierarchy can be derived by recursively removing the shortest (longest) edge

Single-Link clustering is related to the problem of finding **maximal connected components** (Zusammenhangskomponenten) on a graph that contains only edges (d,d‘) for which sim(d,d‘) is above some threshold

Complete-Link clustering is related to the problem of finding **maximal cliques** in a graph.

Bottom-up Clustering with Group-Average Metric (1)

Merge step combines those clusters c_i and c_j

for which the intra-cluster similarity $c := c_i \cup c_j$

$$S(c) := \frac{1}{|c| \cdot (|c| - 1)} \sum_{\substack{d, d' \in c \\ d \neq d'}} sim(d, d')$$ becomes maximal

naive implementation has run-time $O(n^3)$:

n-1 merge steps each with $O(n^2)$ computations

Bottom-up Clustering with Group-Average Metric (2)

efficient implementation – with total run-time O(n^2) – for cosine similarity with length-normalized vectors, i.e. using scalar product for sim

precompute similarity of all document pairs

and compute $\vec{s}(c) := \sum_{\vec{d} \in c} \vec{d}$

for each cluster after every merge step

Then:

$$S(c_i \cup c_j) = \frac{\left(\vec{s}(c_i) + \vec{s}(c_j)\right) \cdot \left(\vec{s}(c_i) + \vec{s}(c_j)\right) - (|c_i| + |c_j|)}{(|c_i| + |c_j|)(|c_i| + |c_j| - 1)}$$

Thus each merge step can be carried out in constant time.

Cluster Quality Measures (1)

With regard to **ground truth:**
known class labels $L_1, \ldots, L_g$ for data points $d_1, \ldots, d_n$:
$L(d_i) = L_j \in \{L_1, \ldots, L_g\}$

With cluster assignment $\Gamma(d_1), \ldots, \Gamma(d_n) \in c_1, \ldots, c_k$
cluster c_j has **purity** $\max_{\nu=1..g} |\{d \in c_j \mid L(d) = L_\nu\}| \,/\, |c_j|$

Complete clustering has purity $\sum_{j=1..k} purity(c_j)/k$

<u>Alternatives:</u>

- **Entropy** within cluster $\sum_{\nu=1..g} \frac{|c_j \cap L_\nu|}{|c_j|} \log_2 \frac{|c_j|}{|c_j \cap L_\nu|}$

- **MI** between cluster and classes

$$\sum_{c \in \{c_j, \bar{c}_j\}, L \in \{L_1, \ldots, L_g\}} \frac{|c \cap L|/n}{|c| \cdot |L|/n} \log_2 \frac{|c| \cdot |L|/n}{|c \cap L|/n}$$

Cluster Quality Measures (2)

Without any ground truth:

ratio of intra-cluster to inter-cluster similarities

$$\frac{1}{k}\sum_{k}\left(\frac{1}{|c_k|}\sum_{\vec{d}\in c_k} sim(\vec{d},\vec{c}_k)\right) \;/\; \left(\frac{1}{k\,(k-1)}\sum_{\substack{i,j\\ i\neq j}} sim(\vec{c}_i,\vec{c}_j)\right)$$

or other ***cluster validity measures*** of this kind
(e.g. considering variance of intra- and inter-cluster distances)

7.2 Flat Clustering: Simple Single-Pass Method

<u>given:</u> data records d1, ..., dn
<u>wanted:</u> (up to) k clusters C:={c1, ..., ck}

```
C := {{d1}}; /* random choice for the first cluster */
for i:=2 to n do
   determine cluster cj ∈ C with the largest value of
   sim(di, cj) (e.g. sim(di, c⃗j) with centroid c⃗j );
    if sim(di, cj) ≥ threshold
    then assign di to cluster cj
    else if |C| < k
           then C := C ∪ {{di}}; /* create new cluster */
           else assign di to cluster cj
           fi
    fi
od
```

K-Means Method for Flat Clustering (1)

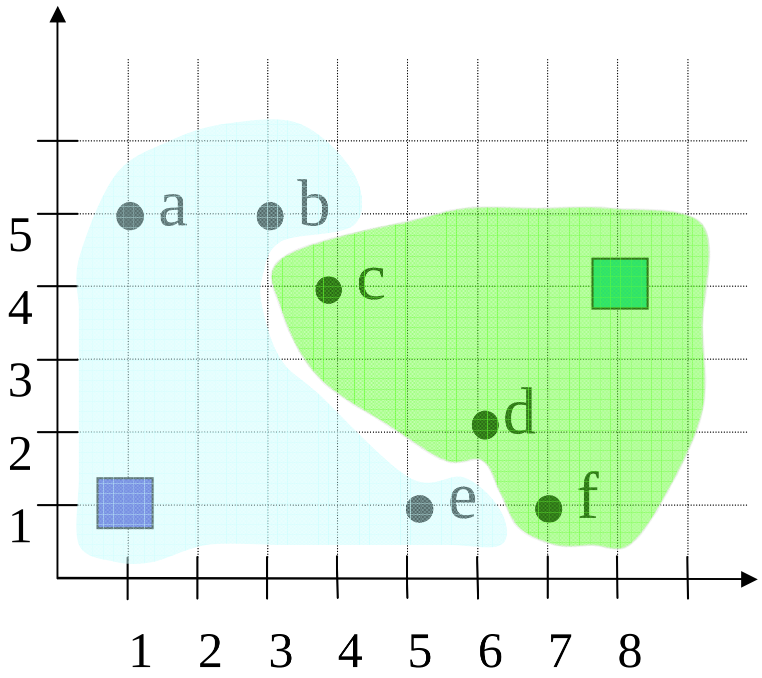
Idea:

- determine **k prototype vectors**, one for each cluster
- **assign each data record to the most similar prototype vector** and compute new prototype vector (e.g. by averaging over the vectors assigned to a prototype)
- **iterate** until clusters are sufficiently stable

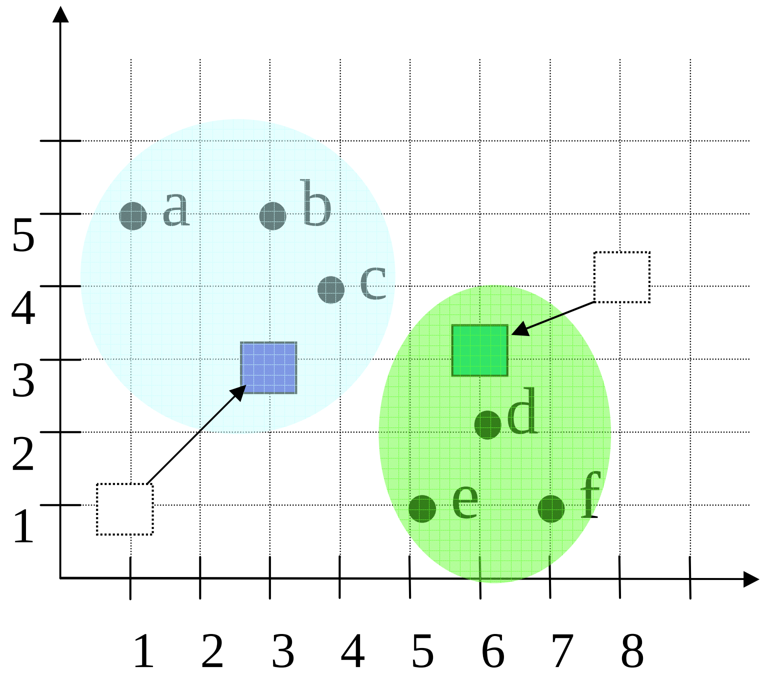
randomly choose k prototype vectors $\vec{c}_1, ..., \vec{c}_k$
while not yet sufficiently stable do
  for i:=1 to n do
    assign di to cluster cj for which $sim(\vec{d}_i, \vec{c}_j)$ is minimal
  od;
  for j:=1 to k do $\vec{c}_j := \frac{1}{|c_j|} \sum_{\vec{d} \in c_j} \vec{d}$ od;
od;

Example for K-Means Clustering

K=2 • data records ■ prototype vectors



after 1st iteration



after 2nd iteration

K-Means Method for Flat Clustering (2)

- run-time is O(n) (assuming constant number of iterations)
- a suitable number of clusters, K, can be determined experimentally or based on the MDL principle
- the initial prototype vectors could be chosen by using another – very efficient – clustering method
  (e.g. bottom-up clustering on random sample of the data records).
- for sim any arbitrary metric can be used

Choice of K (Model Selection)

- application-dependent (e.g. for visualization)
- driven by empirical evaluation of cluster quality
  (e.g. cross-validation with held-out labeled data)
- driven by quality measure without ground truth
- driven by MDL principle

LSI and pLSI Reconsidered

LSI and pLSI can also be seen as
unsupervised clustering methods (*spectral clustering*):

simple variant for k clusters

- map each data point into k-dimensional space
- assign each point to its highest-value dimension
  (strongest spectral component)

Conversely, we could compute k clusters
for the data points (using any clustering algorithm)
and project data points onto k centroid vectors („axes“ of k-dim. space)
to represent data in LSI-style manner

EM Method for Model-based Soft Clustering (Expectation Maximization)

Approach:

- generalize K-Means method such that each data record belongs to a cluster (actually all k clusters) with a certain probability based on a parameterized multivariate prob. distribution f
  $\rightarrow$ random variable $Z_{ij} = 1$ if d_i belongs to c_j, 0 otherwise
- estimate parameters θ of the prob. distribution $f(\theta,x)$ such that the likelihood that the observed data is indeed a sample from this distribution is maximized
  $\rightarrow$ **Maximum-Likelihood Estimation (MLE):**
  maximize $L(d_1,...,d_n, \theta) = P[d_1, ..., d_n$ is a sample from $f(\theta,x)]$
  or maximize log L;
  if analytically intractable $\rightarrow$ use **EM iteration procedure**

Postulate probability distribution e.g.
mixture of k multivariate Normal distributions

EM Clustering Method with Mixture of k Multivariate Normal Distributions

<u>Assumption:</u> data records are a sample from a mixture of k multivariate Normal distributions with the density:

$$f(\vec{x}, \pi_1, \ldots, \pi_k, \vec{\mu}_1, \ldots, \vec{\mu}_k, \Sigma_1, \ldots, \Sigma_k)$$

$$= \sum_{j=1}^{k} \pi_j \; n(\vec{x}, \vec{\mu}_j, \Sigma_j) = \sum_{j=1}^{k} \pi_j \; \frac{1}{\sqrt{(2\pi)^m \left|\Sigma_j\right|}} e^{-\frac{1}{2}(\vec{x}-\vec{\mu}_j)^T \Sigma_j^{-1} (\vec{x}-\vec{\mu}_j)}$$

with expectation values $\vec{\mu}_j$
and invertible, positive definite, symmetric
m×m covariance matrices Σ_j

$\rightarrow$ maximize log-likelihood function:

$$\log L(\vec{x}_1, \ldots, \vec{x}_n, \theta) := \log \prod_{i=1}^{n} P[\vec{x}_i \mid \theta] = \sum_{i=1}^{n} \left(\log \sum_{j=1}^{k} \pi_j \; n(\vec{x}_i, \vec{\mu}_j, \Sigma_j) \right)$$

EM Iteration Procedure (1)

introduce latent variables Z_{ij}: point x_i generated by cluster j

initialization of EM method, for example, by:
setting $\pi_1 = ... = \pi_k = 1/k$, using K-Means cluster centroids for $\vec{\mu}_1, ..., \vec{\mu}_k$
and unity matrices (1s on diagonal) for $\Sigma_1, ..., \Sigma_k$

iterate until parameter estimations barely change anymore:

1) **Expectation step (E step)**:
   compute $E[Z_{ij}]$ based on the previous round's estimation
   for θ, i.e. $\pi_1, ..., \pi_k$, $\vec{\mu}_1, ..., \vec{\mu}_k$ and $\Sigma_1, ..., \Sigma_k$
2) **Minimization step (M step)**:
   improve parameter estimation for θ based on
   the previous round's values for $E[Z_{ij}]$

convergence is guaranteed, but may result in
local maximum of log-likelihood function

EM Iteration Procedure (2)

Expectation step (E step):

$$h_{ij} := E[Z_{ij} \mid \vec{x}_i, \theta] = \frac{\pi_j \, P[\vec{x}_i \mid n_j(\theta)]}{\sum_{l=1}^{k} \pi_l \, P[\vec{x}_i \mid n_l(\theta)]}$$

Maximization step (M step):

$$\vec{\mu}_j := \frac{\sum_{i=1}^{n} h_{ij} \vec{x}_i}{\sum_{i=1}^{n} h_{ij}} \qquad \Sigma_j := \frac{\sum_{i=1}^{n} h_{ij} (\vec{x}_i - \vec{\mu}_j)(\vec{x}_i - \vec{\mu}_j)^T}{\sum_{i=1}^{n} h_{ij}}$$

$$\pi_j := \frac{\sum_{i=1}^{n} h_{ij}}{\sum_{j=1}^{k} \sum_{i=1}^{n} h_{ij}} = \frac{\sum_{i=1}^{n} h_{ij}}{n}$$

Example for EM Clustering Method

given:
n=20 terms from articles of the New York Times:
ballot, polls, Gov, seats, profit, finance, payments, NFL, Reds, Sox, inning, quarterback, score, scored, researchers, science, Scott, Mary, Barbara, Edward
with m=20-dimensional feature vectors $\vec{d}_i$
with d_{ij} = # articles that contain both term i and term j

Result of EM clustering for the estimation of h_{ij} for k=5:

	1	2	3	4	5
ballot	0.63	0.12	0.04	0.09	0.11
polls	0.58	0.11	0.06	0.10	0.14
Gov	0.58	0.12	0.03	0.10	0.17
seats	0.55	0.14	0.08	0.08	0.15
profit	0.11	0.59	0.02	0.14	0.15
finance	0.15	0.55	0.01	0.13	0.16
payments	0.12	0.66	0.01	0.09	0.11
NFL	0.13	0.05	0.58	0.09	0.16
Reds	0.05	0.01	0.86	0.02	0.06
Sox	0.05	0.01	0.86	0.02	0.06

	1	2	3	4	5
inning	0.03	0.01	0.93	0.01	0.02
quarterback	0.06	0.02	0.82	0.03	0.07
score	0.12	0.04	0.65	0.06	0.13
scored	0.08	0.03	0.79	0.03	0.07
researchers	0.08	0.12	0.02	0.68	0.10
science	0.12	0.12	0.03	0.54	0.19
Scott	0.12	0.12	0.11	0.11	0.54
Mary	0.10	0.10	0.05	0.15	0.59
Barbara	0.15	0.11	0.04	0.12	0.57
Edward	0.16	0.18	0.02	0.12	0.51

Clustering with Density Estimator

Influence function $g_y(x) : \left(R_0^+\right)^m \rightarrow R$

influence of data record y
on a point x in its local environment

e.g. $g_y(x) = e^{-\frac{dist(x,y)^2}{2\sigma^2}}$ with $dist(x,y) := \frac{1}{1 + sim(x,y)}$

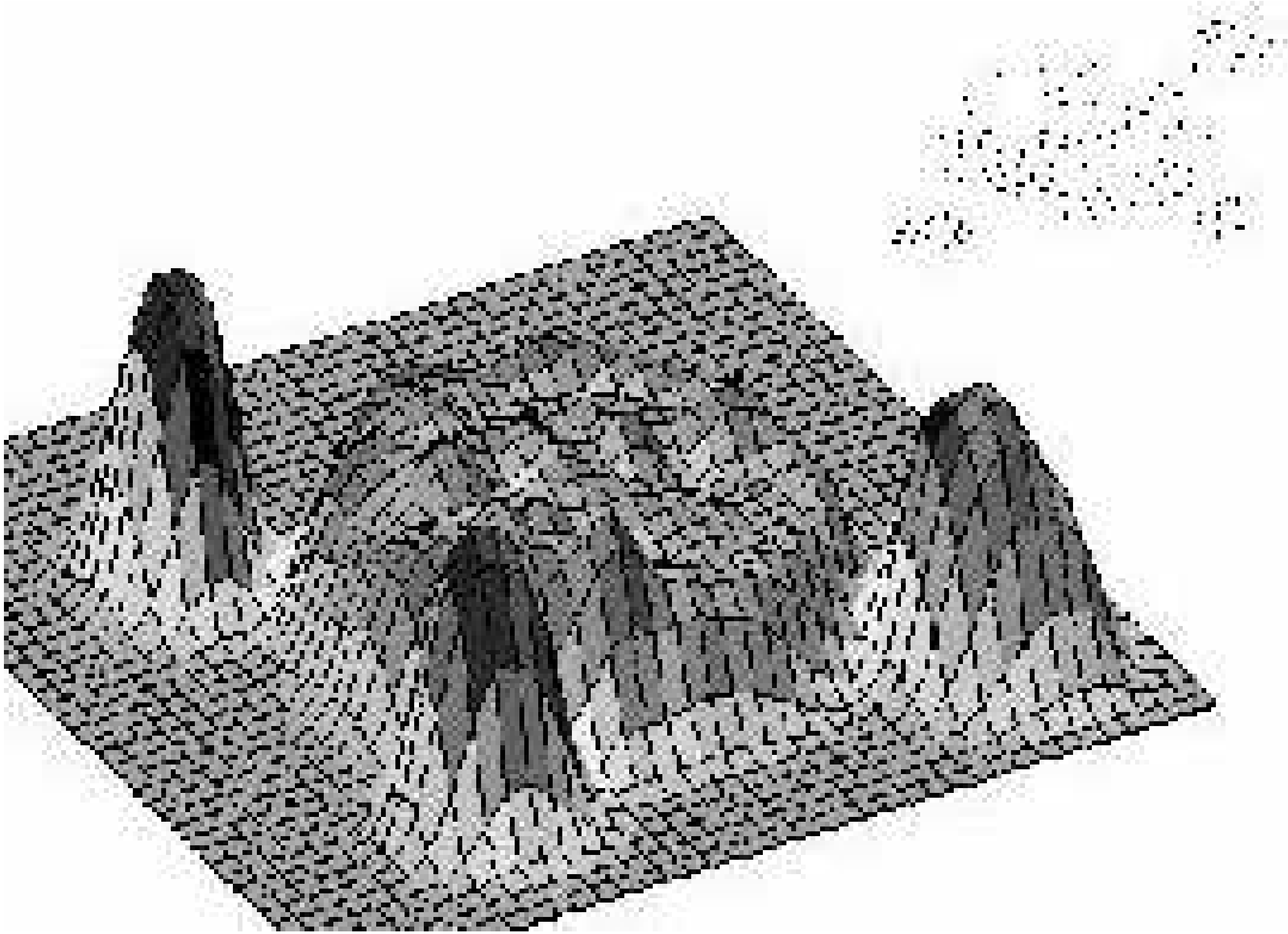
Density function $f(x) : \left(R_0^+\right)^m \rightarrow R$

density at point x: sum of all influences y on x

$$f(x) = \sum_{y \in D} g_y(x)$$

clusters correspond to local maxima of the density function

Example for Clustering with Density Estimator



Source: D. Keim and A. Hinneburg, Clustering Techniques for Large Data Sets, Tutorial, KDD Conf. 1999

Incremental DBSCAN Method for Density-based Clustering [Ester et al.: KDD 1996]

DBSCAN = Density-Based Clustering for Applications with Noise

<u>simplified version of the algorithm:</u>

for each data point d do {
  insert d into spatial index (e.g., R-tree);
  locate all points with distance to d < *max_dist*;
  if these points form a single cluster then add d to this cluster
  else {
    if there are at least *min_points* data points
      that do not yet belong to a cluster
      such that for all point pairs the distance < *max_dist*
    then construct a new cluster with these points };
};

average run-time is O(n * log n);
data points that are added later can be easily assigned to a cluster;
points that do not belong to any cluster are considered „noise“

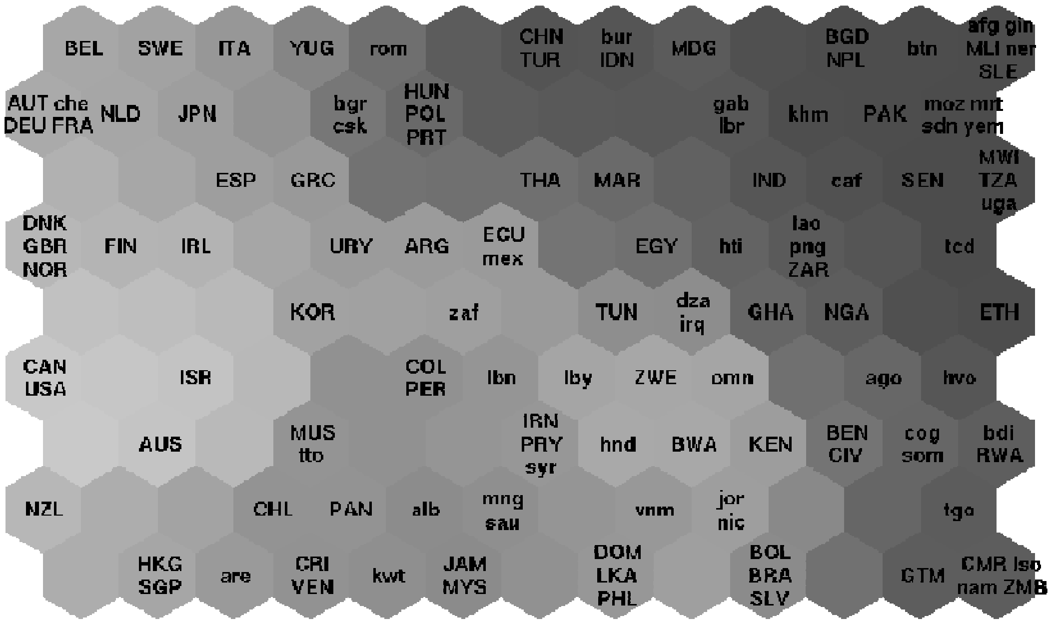
7.3 Self-Organizing Maps (SOMs, Kohonen Maps)

similar to K-Means
but **embeds data and clusters in a low-dimensional space** (e.g. 2D) and aims to **preserve cluster-cluster neighborhood – for visualization**
(recall: clustering does not assume a vector space, only a metric space)

clusters c1, c2, ... and data x1, x2, ... are points with distance function sim (xi, xj), sim (ci, xj), sim (ci, cj)

initialize map with k cluster nodes arbitrarily placed
(often on a triangular or rectangular grid)
for each x determine node C(x) closest to x and small node set N(x) close to x
repeat
    for randomly chosen x
        update all nodes c'∈ N(x): $\vec{c}' := \vec{c}' + \lambda(t) \cdot sim(c', C(x)) \cdot (\vec{x} - \vec{c}')$
        under influence of data point x (with learning rate λ(t))
        („data activates neuron C(x) and other neurons c' in its neighborhood")
until sufficient convergence (with gradually reduced λ(t))
assign data point x to the closest cluster („winner neuron")

SOM Example (1)



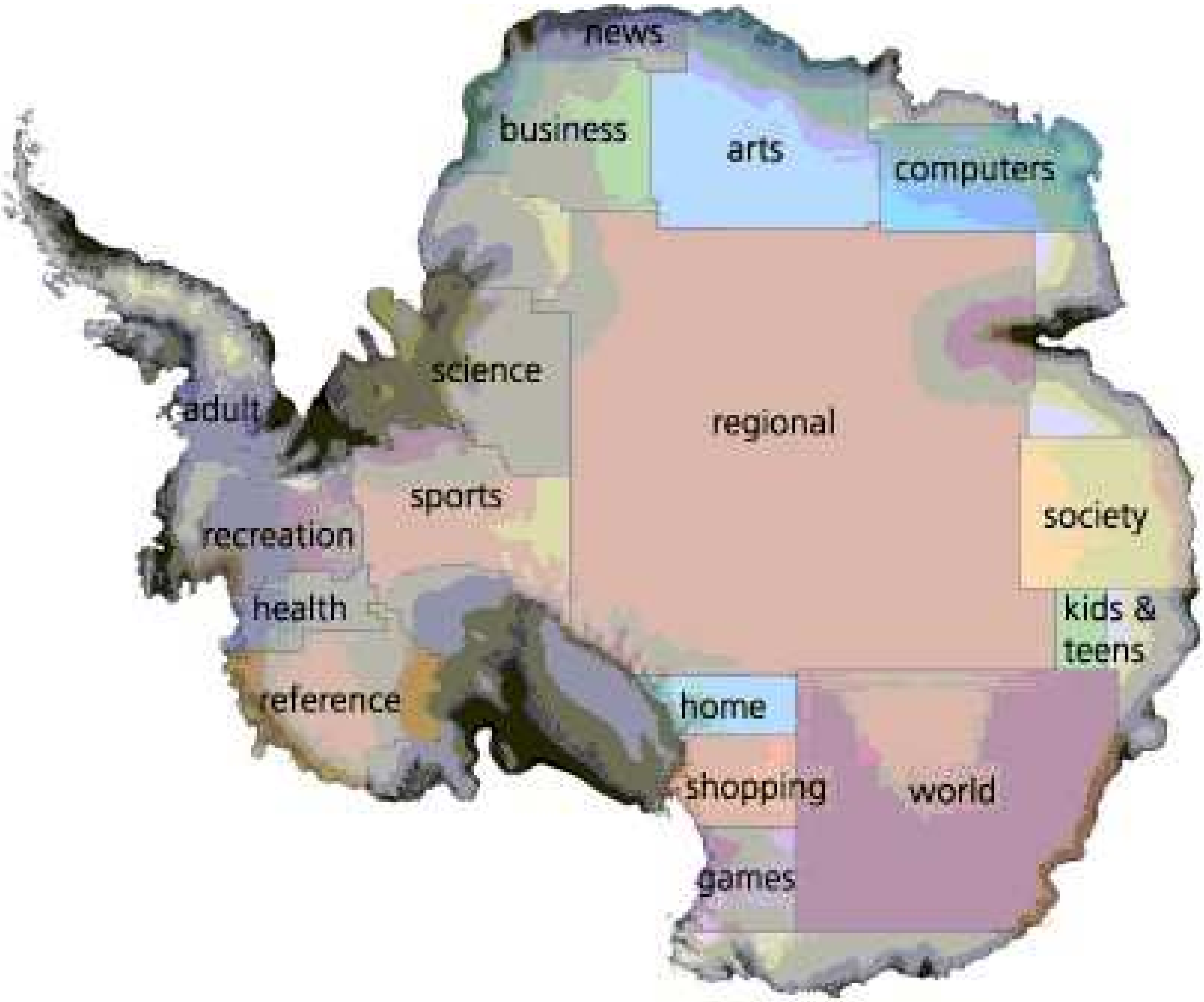
The Country Names

AFG	Afghanistan	GTM	Guatemala	NZL	New Zealand
AGO	Angola	HKG	Hong Kong	OAN	Taiwan, China
ALB	Albania	HND	Honduras	OMN	Oman
ARE	United Arab Emirates	HTI	Haiti	PAK	Pakistan
ARG	Argentina	HUN	Hungary	PAN	Panama
AUS	Australia	HVO	Burkina Faso	PER	Peru
AUT	Austria	IDN	Indonesia	PHL	Philippines
BDI	Burundi	IND	India	PNG	Papua New Guinea
BEL	Belgium	IRL	Ireland	POL	Poland
BEN	Benin	IRN	Iran, Islamic Rep.	PRT	Portugal
BGD	Bangladesh	IRQ	Iraq	PRY	Paraguay
BGR	Bulgaria	ISR	Israel	ROM	Romania
BOL	Bolivia	ITA	Italy	RWA	Rwanda
BRA	Brazil	JAM	Jamaica	SAU	Saudi Arabia
BTN	Bhutan	JOR	Jordan	SDN	Sudan
BUR	Myanmar	JPN	Japan	SEN	Senegal
BWA	Botswana	KEN	Kenya	SGP	Singapore
CAF	Central African Rep.	KHM	Cambodia	SLE	Sierra Leone
CAN	Canada	KOR	Korea, Rep.	SLV	El Salvador
CHE	Switzerland	KWT	Kuwait	SOM	Somalia
CHL	Chile	LAO	Lao PDR	SWE	Sweden
CHN	China	LBN	Lebanon	SYR	Syrian Arab Rep.
CIV	Cote d'Ivoire	LBR	Liberia	TCD	Chad
CMR	Cameroon	LBY	Libya	TGO	Togo
COG	Congo	LKA	Sri Lanka	THA	Thailand
COL	Colombia	LSO	Lesotho	TTO	Trinidad and Tobago
CRI	Costa Rica	MAR	Morocco	TUN	Tunisia
CSK	Czechoslovakia	MDG	Madagascar	TUR	Turkey
DEU	Germany	MEX	Mexico	TZA	Tanzania
DNK	Denmark	MLI	Mali	UGA	Uganda
DOM	Dominican Rep.	MNG	Mongolia	URY	Uruguay
DZA	Algeria	MOZ	Mozambique	USA	United States
ECU	Ecuador	MRT	Mauritania	VEN	Venezuela
EGY	Egypt, Arab Rep.	MUS	Mauritius	VNM	Viet Nam
ESP	Spain	MWI	Malawi	YEM	Yemen, Rep.
ETH	Ethiopia	MYS	Malaysia	YUG	Yugoslavia
FIN	Finland	NAM	Namibia	ZAF	South Africa
FRA	France	NER	Niger	ZAR	Zaire
GAB	Gabon	NGA	Nigeria	ZMB	Zambia
GBR	United Kingdom	NIC	Nicaragua	ZWE	Zimbabwe
GHA	Ghana	NLD	Netherlands		
GIN	Guinea	NOR	Norway		
GRC	Greece	NPL	Nepal		

from http://www.cis.hut.fi/research/som-research/worldmap.html

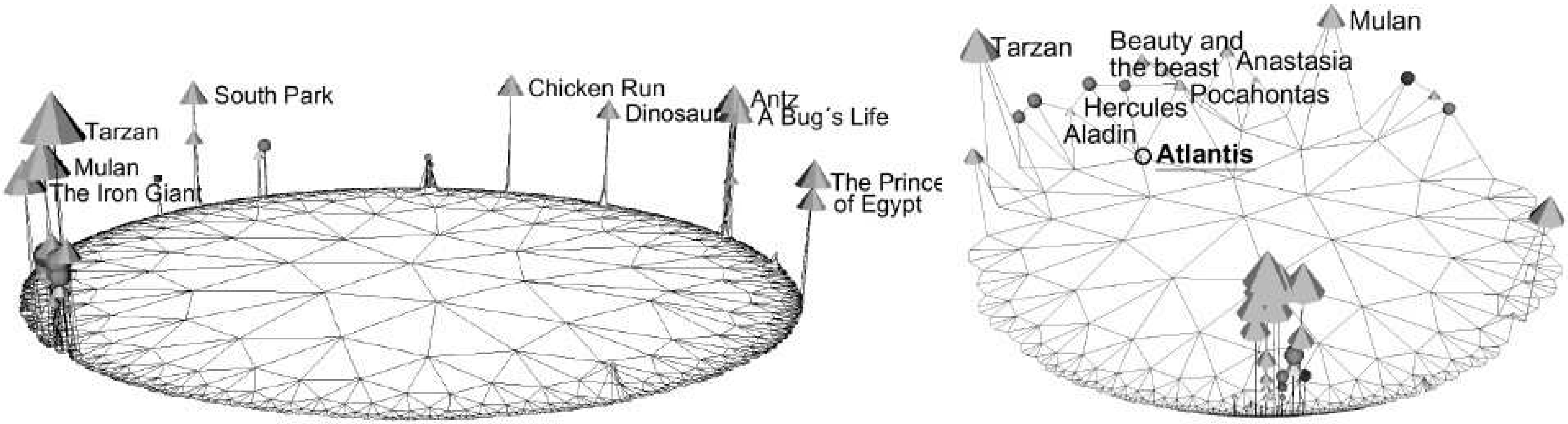
see also http://maps.map.net/ for another - interactive - example

SOM Example (2): WWW Map (2001)



Source: www.antarcti.ca, 2001

SOM Example (3): Hyperbolic Visualization



Source: J. Ontrup, H. Ritter: Hyperbolic Self-Organizing Maps for Semantic Navigation, NIPS 2001

SOM Example (4): „Islands of Music“

Source: E. Pampalk: Islands of Music: Analysis, Organization, and Visualization of Music Archives, Master Thesis, Vienna University of Technology

http://www.ofai.at/~elias.pampalk/music/

Multi-dimensional Scaling (MDS)

<u>Goal:</u>
map data (from metric space) into low-dimensional vector space such that the distances of data x_i are approximately preserved by the Euclidean distances of the images $\hat{x}_i = f(x_i)$ in the vector space

$$\rightarrow \text{minimize stress} = \frac{\sum_{i,j} \left(\left\| \hat{x}_i - \hat{x}_j \right\| - \text{dist}(x_i, x_j) \right)^2}{\sum_{i,j} \text{dist}(x_i, x_j)^2}$$

$\rightarrow$ solve iteratively with hill climbing:
start with random (or heuristic) placement of data in vector space
find point pair with highest tension
move points locally so as to reduce the stress
(on a fictitious spring that connects the points)

$O(n^2)$ run-time in each iteration, impractical for very large data sets

FastMap

Idea:

pretend that the data are points in an unknown n-dim. vector space and project them into a k-dimensional space by determining their coordinates in k rounds, one dimension at a time

Algorithm:

determine two **pivot objects a and b** (e.g. objects far apart)

conceptually **project all data points x onto the line between a and b**

$\rightarrow$ solve for x_1: $$\mathrm{dist}(b,x)^2 = \mathrm{dist}(a,x)^2 + \mathrm{dist}(a,b)^2 - 2x_1 \mathrm{dist}(a,b)$$

(cosine law)

consider **(n-1)-dim. hyperplane perpendicular to the projection line**

with new distances: $$\mathrm{dist}_{n-1}(x,y)^2 = \mathrm{dist}_n(x,y)^2 - (x_1 - y_1)^2$$

(Pythagoras)

recursively call FastMap for (n-1)-dimensional data

7.4 Applications: Cluster-based Information Retrieval

for user query q:

- compute ranking of cluster centroids with regard to q
- evaluate query q on the cluster or clusters
  with the most similar centroid(s)
  (possibly in conjunction with relevance feedback by user)

cluster browsing:

user can navigate through cluster hierarchy

each cluster c_k is represented by its ***medoid***:

the document $d' \in c_k$ for which the sum $\sum_{d \in C_k - \{d'\}} sim(d', d)$

is maximal (or has highest similarity to cluster centroid)

Automatic Labeling of Clusters

- Variant 1:
  classification of cluster centroid $\vec{c}_k$
  with a separate, supervised, classifier
- Variant 2:
  using term or terms with the highest
  (tf*idf-) weight in the cluster centroid $\vec{c}_k$
- Variant 2‘:
  computing an approximate centroid $\vec{c}_k\,'$ based
  on m‘ (m‘ << m) terms with the highest weights in the cluster‘s docs
  and using the highest-weight term or terms of $\vec{c}_k\,'$
- Variant 3:
  identifying most characteristic terms or phrases for each cluster,
  using MI or other entropy measures

Clustering Query Logs

Motivation:

- statistically identify FAQs (for intranets and portals), taking into account variations in query formulation
- capture correlation between queries and subsequent clicks

Model/Notation:

a **user session is a pair (q, D+)** with a query q and
D+ denoting the result docs on which the user clicked;
len(q) is the number of keywords in q

Similarity Measures between User Sessions

- tf*idf based similarity between query keywords only
- edit distance based similarity: sim(p,q) = 1 – ed(p,q) / max(len(p),len(q))

  Examples: Where does silk come from? Where does dew come from?
  How far away is the moon? How far away is the nearest star?

- similarity based on common clicks: $$sim(p,q) = \frac{|D_p^+ \cap D_q^+|}{\max(|D_p^+|, |D_q^+|)}$$

  Example: atomic bomb, Manhattan project, Nagasaki, Hiroshima, nuclear weapon

- similarity based on common clicks and document hierarchy:

$$sim(p,q) = \frac{1}{2}\left(\left(\sum_{d' \in D_p^+} \max\{s(d',d'') \mid d'' \in D_q^+\}\right) / |D_p^+| + \left(\sum_{d' \in D_q^+} \max\{s(d',d'') \mid d'' \in D_p^+\}\right) / |D_q^+|\right)$$

with $s(d',d'') = \frac{level(lca(d',d'')) - 1}{\max level - 1}$

p=law of thermodynamics
$D+_p$ = {/Science/Physics/Conservation Laws, ...}
q=Newton law
$D+_q$ = {/Science/Physics/Gravitation, ...}

- linear combinations of different similarity measures

Query Expansion based on Relevance Feedback

Given: a query q, a result set (or ranked list) D,
a user's assessment u: D $\rightarrow$ {+, –}
yielding positive docs $D^+ \subseteq D$ and negative docs $D^- \subseteq D$

Goal: derive query q' that better captures the user's intention
or a better suited similarity function, e.g., by
- changing weights in the query vector or
- changing weights for different aspects of similarity
(color vs. shape in multimedia IR, different colors,
relevance vs. authority vs. recency)

Classical approach: ***Rocchio method*** (for term vectors)

$$\boldsymbol{q}' = \alpha \boldsymbol{q} + \frac{\beta}{|\boldsymbol{D}^+|} \sum_{\boldsymbol{d} \in \boldsymbol{D}^+} \boldsymbol{d} - \frac{\gamma}{|\boldsymbol{D}^-|} \sum_{\boldsymbol{d} \in \boldsymbol{D}^-} \boldsymbol{d}$$

with $\alpha, \beta, \gamma \in [0,1]$ and typically $\alpha > \beta > \gamma$

Pseudo-Relevance Feedback

based on J. Xu, W.B. Croft: Query expansion using local and global document analysis, SIGIR Conference, 1996

Lazy users may perceive feedback as too bothersome

Evaluate query and simply view top n results as positive docs:

Add these results to the query and re-evaluate or

Select „best“ terms from these results and expand the query

Experimental Evaluation

on MS Encarta corpus,
with 4 Mio. query log entries and 40 000 doc. subset

Considers short queries and long phrase queries, e.g.:

Michael Jordan	Michael Jordan in NBA matches
genome project	Why is the genome project so crucial for humans?
Manhattan project	What is the result of Manhattan project on Word War II?
Windows	What are the features of Windows that Microsoft brings us?

(Phrases are decomposed into N-grams that are in dictionary)

Query expansion with related terms/phrases:
Avg. precision [%] at different recall values:

Short queries:

Recall	q alone	PseudoRF (n=100,m=30)	Query Log (m=40)
10%	40.67	45.00	62.33
20%	27.00	32.67	44.33
30%	20.89	26.44	36.78
100%	8.03	13.13	17.07

Long queries:

Recall	q alone	PseudoRF (n=100,m=30)	Query Log (m=40)
10%	46.67	41.67	57.67
20%	31.17	34.00	42.17
30%	25.67	27.11	34.89
100%	11.37	13.53	16.83

Additional Literature for Chapter 7

- S. Chakrabarti, Chapter 4: Similarity and Clustering
- C.D. Manning / H. Schütze, Chapter 14: Clustering
- R.O. Duda / P.E. Hart / D.G. Stork, Ch. 10: Unsupervised Learning and Clustering
- M.H. Dunham, Data Mining, Prentice Hall, 2003, Chapter 5: Clustering
- D. Hand, H. Mannila, P. Smyth: Principles of Data Mining, MIT Press, 2001, Chapter 9: Descriptive Modeling
- M. Ester, J. Sander: Knowledge Discovery in Databases, Springer, 2000, Kapitel 3: Clustering
- C. Faloutsos: Searching Multimedia Databases by Content, 1996, Ch. 11:FastMap
- M. Ester et al.: A density-based algorithm for discovering clusters in large spatial databases with noise, KDD Conference, 1996
- J. Kleinberg: An impossibility theorem for clustering, NIPS Conference, 2002
- G. Karypis, E.-H. Han: Concept Indexing: A Fast Dimensionality Reduction Algorithm with Applications to Document Retrieval & Categorization, CIKM 2000
- M. Vazirgiannis, M. Halkidi, D. Gunopulos: Uncertainty Handling and Quality Assessment in Data Mining, Springer, 2003
- Ji-Rong Wen, Jian-Yun Nie, Hong-Jiang Zhang: Query Clustering Using User Logs, ACM TOIS Vol.20 No.1, 2002
- Hang Cui, Ji-Rong Wen, Jian-Yun Nie, Wei-Ying Ma: Query Expansion by Mining User Logs, IEEE-CS TKDE 15(4), 2003