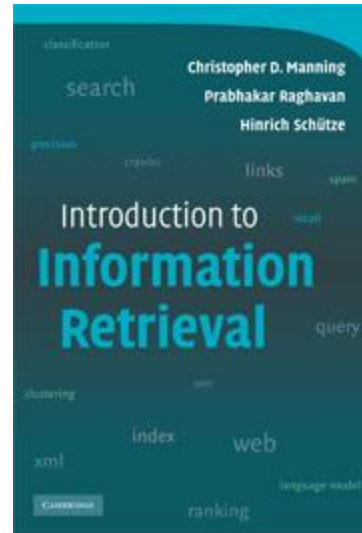


Information Retrieval and Organisation

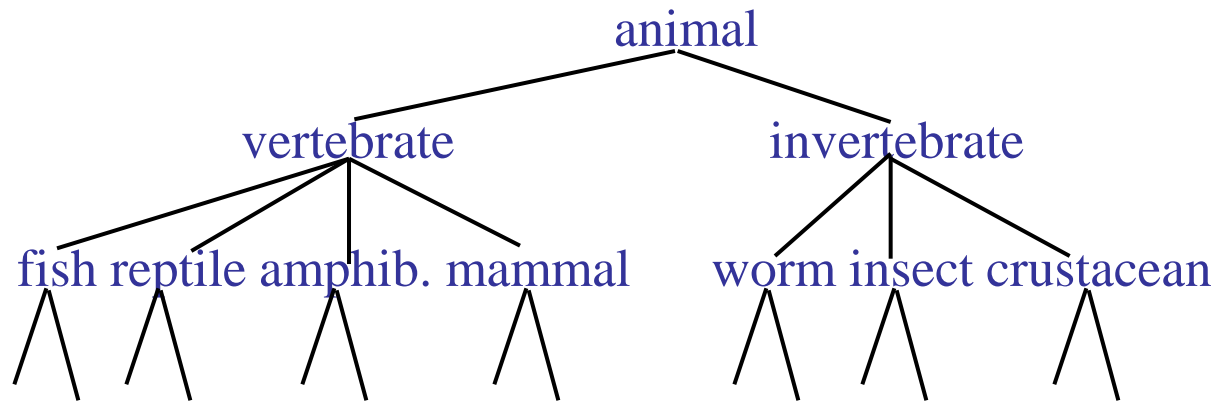


Chapter 17 Hierarchical Clustering

Dell Zhang
Birkbeck, University of London

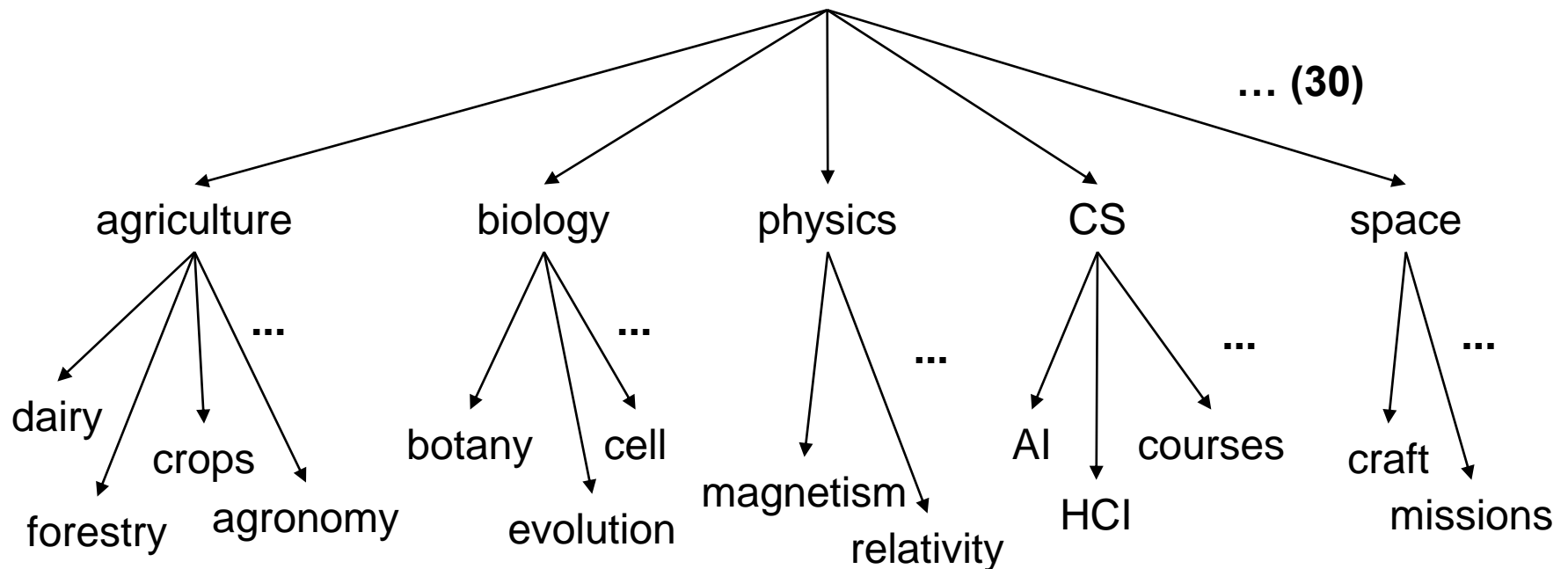
Hierarchical Clustering

- Build a tree-like hierarchical taxonomy (*dendrogram*) from a set of unlabeled documents.



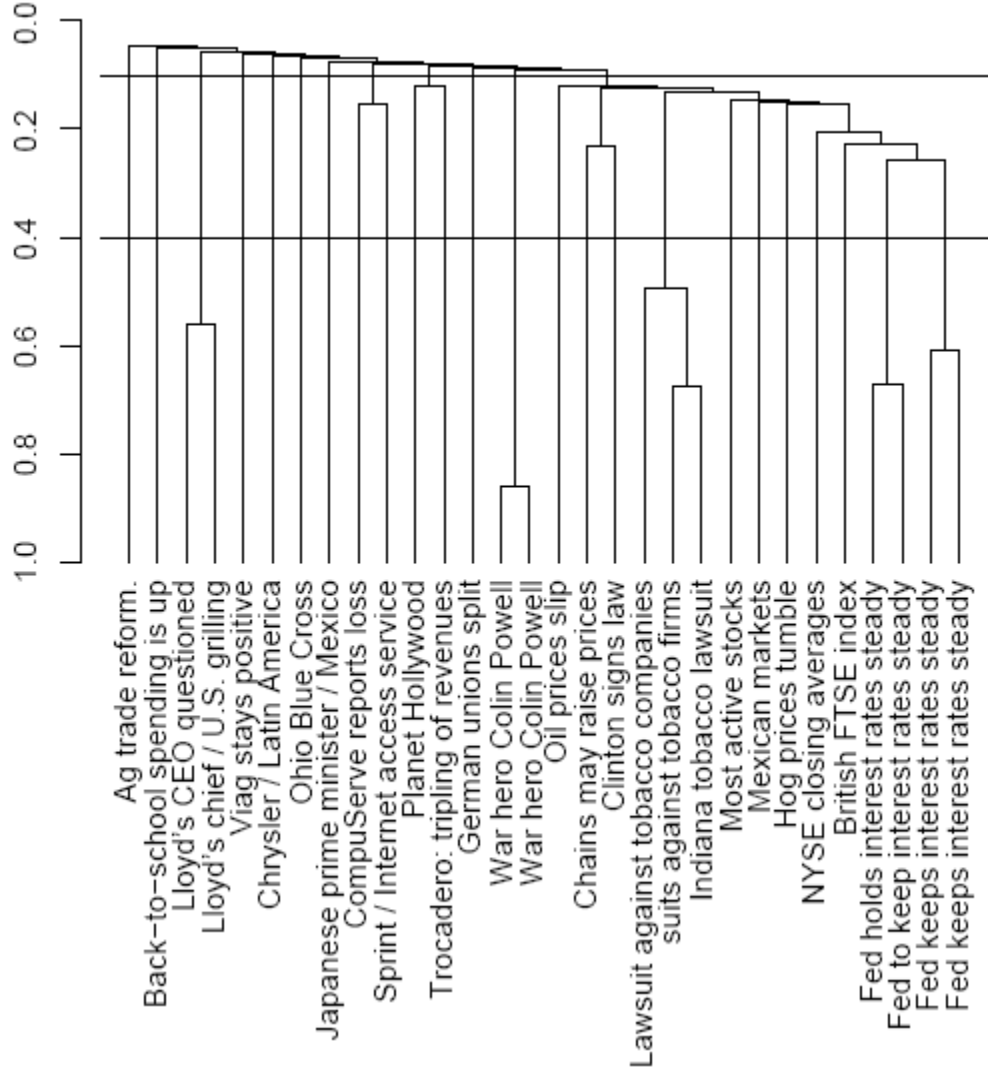
Dendrogram – Example

<http://dir.yahoo.com/science>



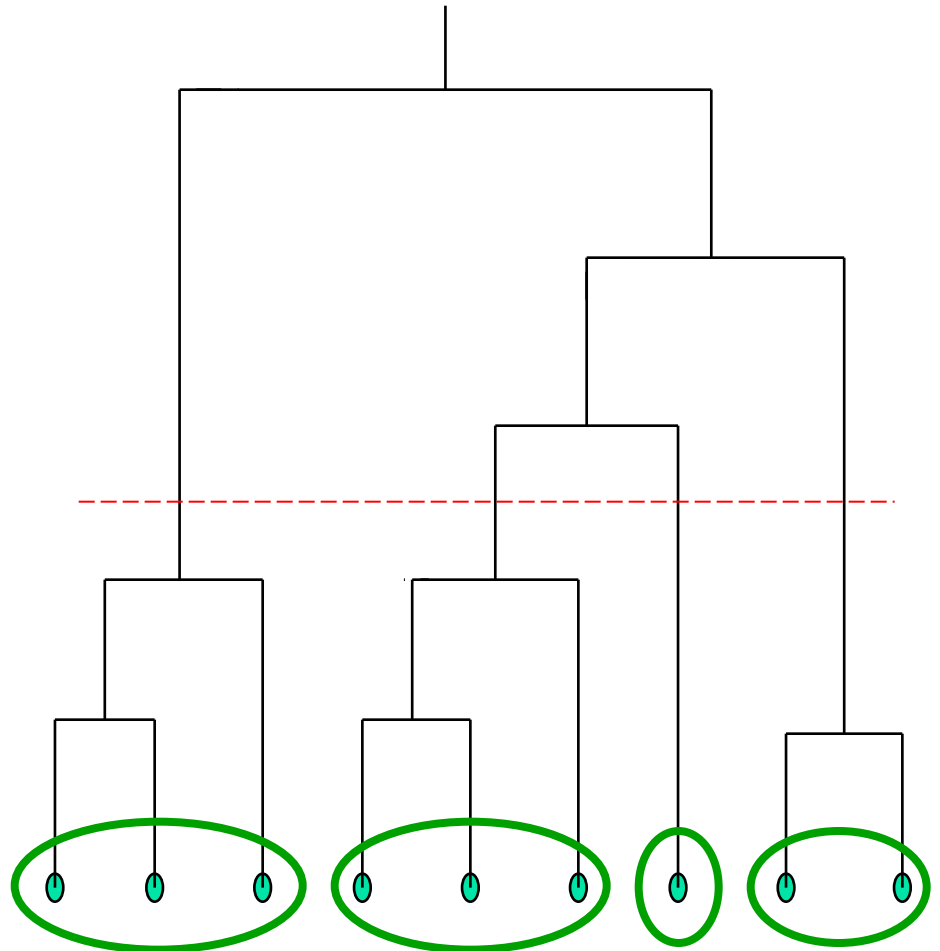
Dendrogram – Example

Clusters of News Stories:
Reuters RCV1



Dendrogram → Clusters

- Clustering can be obtained by cutting the dendrogram at a desired level: each *connected* component forms a cluster.
- The number of clusters is not required in advance.



Divisive vs. Agglomerative

- Divisive (Top-Down)
 - Start with all documents belong to the same cluster. Eventually each node forms a cluster on its own.
 - Recursive application of a (flat) partitioning clustering algorithm
 - e.g., k -means ($k=2$) → bi-secting k -means.
- Agglomerative (Bottom-Up)
 - Start with each document being a single cluster. Eventually all documents belong to the same cluster.

HAC Algorithm

- **Hierarchical Agglomerative Clustering**
 - Starts with each doc in a separate cluster.
 - Repeat until there is only one cluster:
 - Among the current clusters, determine the pair of closest pair of clusters, c_i and c_j
 - Then merges c_i and c_j to a single cluster.
 - The history of merging forms a binary tree or hierarchy (dendrogram).

HAC Alg.

EFFICIENTHAC($\vec{d}_1, \dots, \vec{d}_N$)

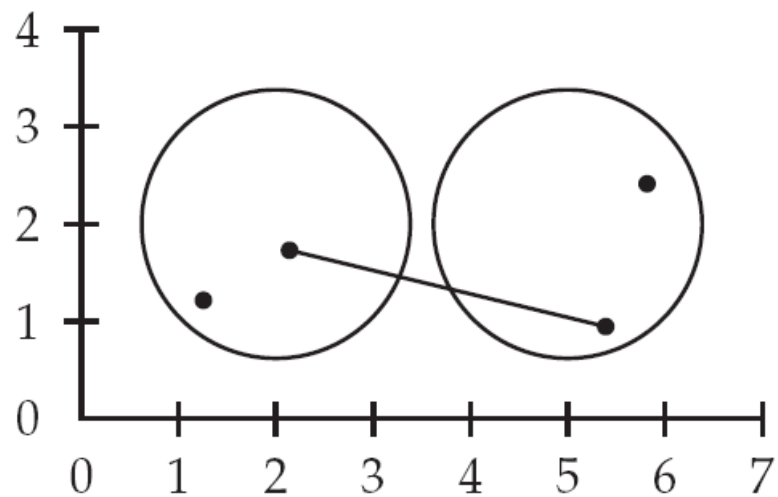
```
1  for  $n \leftarrow 1$  to  $N$ 
2  do for  $i \leftarrow 1$  to  $N$ 
3      do  $C[n][i].sim \leftarrow \vec{d}_n \cdot \vec{d}_i$ 
4           $C[n][i].index \leftarrow i$ 
5       $I[n] \leftarrow 1$ 
6       $P[n] \leftarrow$  priority queue for  $C[n]$  sorted on sim
7       $P[n].DELETE(C[n][n])$  (don't want self-similarities)
8   $A \leftarrow []$ 
9  for  $k \leftarrow 1$  to  $N - 1$ 
10 do  $k_1 \leftarrow \arg \max_{\{k: I[k]=1\}} P[k].MAX().sim$ 
11      $k_2 \leftarrow P[k_1].MAX().index$ 
12      $A.APPEND(\langle k_1, k_2 \rangle)$ 
13      $I[k_2] \leftarrow 0$ 
14      $P[k_1] \leftarrow []$ 
15     for each  $i$  with  $I[i] = 1 \wedge i \neq k_1$ 
16         do  $P[i].DELETE(C[i][k_1])$ 
17              $P[i].DELETE(C[i][k_2])$ 
18              $C[i][k_1].sim \leftarrow SIM(i, k_1, k_2)$ 
19              $P[i].INSERT(C[i][k_1])$ 
20              $C[k_1][i].sim \leftarrow SIM(i, k_1, k_2)$ 
21              $P[k_1].INSERT(C[k_1][i])$ 
22 return  $A$ 
```

Time Complexity

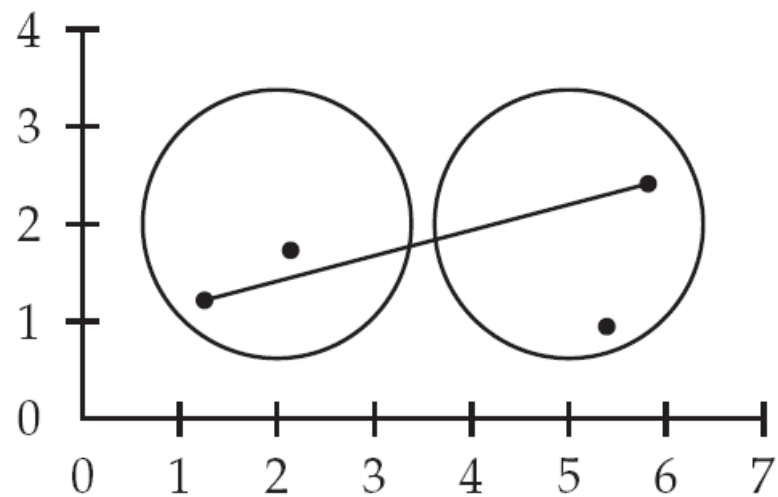
- In the initialization step, compute similarity of all pairs of n documents which is $O(n^2)$.
- In each of the subsequent $n-2$ merging iterations, compute the distance between the most recently created cluster and all other existing clusters.
- The overall time complexity is often $O(n^3)$ if done naively or $O(n^2 \log n)$ if done more cleverly using a priority-queue.

HAC Variants

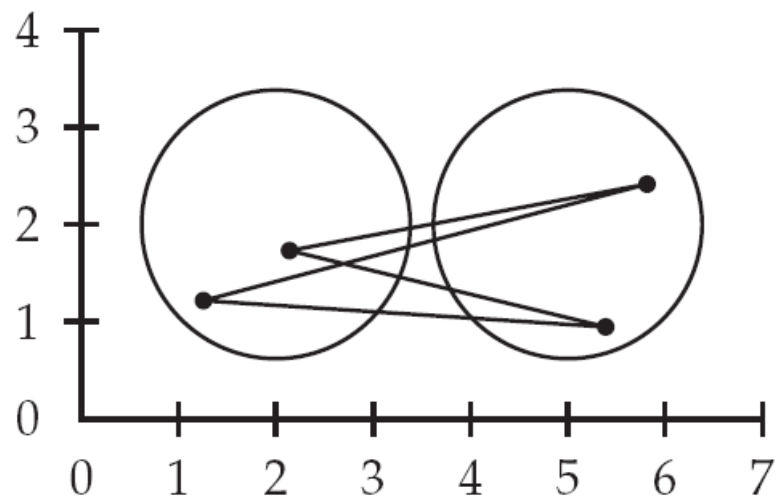
- How to define the closest pair of clusters
 - Single-Link
 - maximum similarity between pairs of docs
 - Complete-Link
 - minimum similarity between pairs of docs
 - Average-Link
 - average similarity between pairs of docs
 - Centroid
 - maximum similarity between cluster centroids



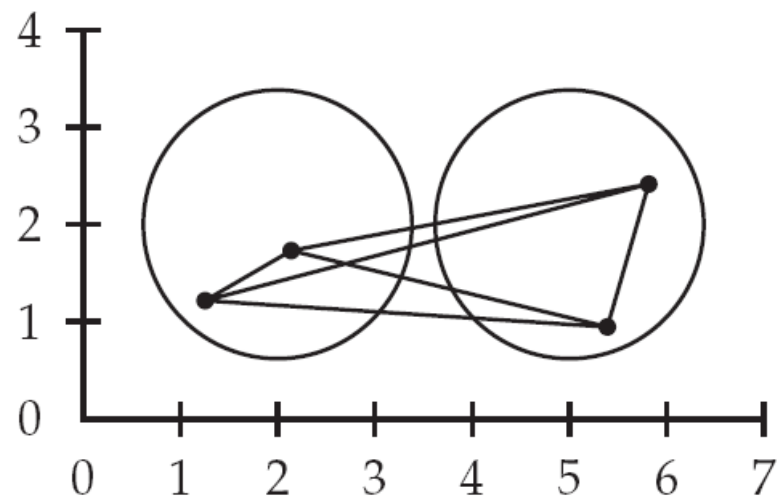
(a) single link: maximum similarity



(b) complete link: minimum similarity



(c) centroid: average inter-similarity



(d) group-average: average of all similarities

Single-Link

- The similarity between a pair of clusters is defined by the single *strongest* link (i.e., maximum cosine-similarity) between their members:

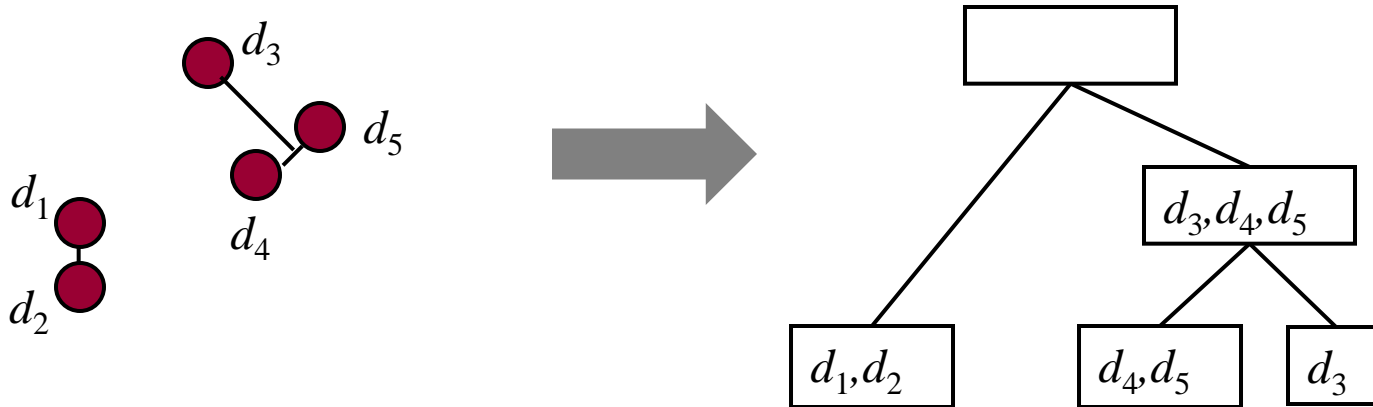
$$\text{sim}(c_i, c_j) = \max_{x \in c_i, y \in c_j} \text{sim}(x, y)$$

- After merging c_i and c_j , the similarity of the resulting cluster to another cluster, c_k , is:

$$\text{sim}((c_i \cup c_j), c_k) = \max(\text{sim}(c_i, c_k), \text{sim}(c_j, c_k))$$

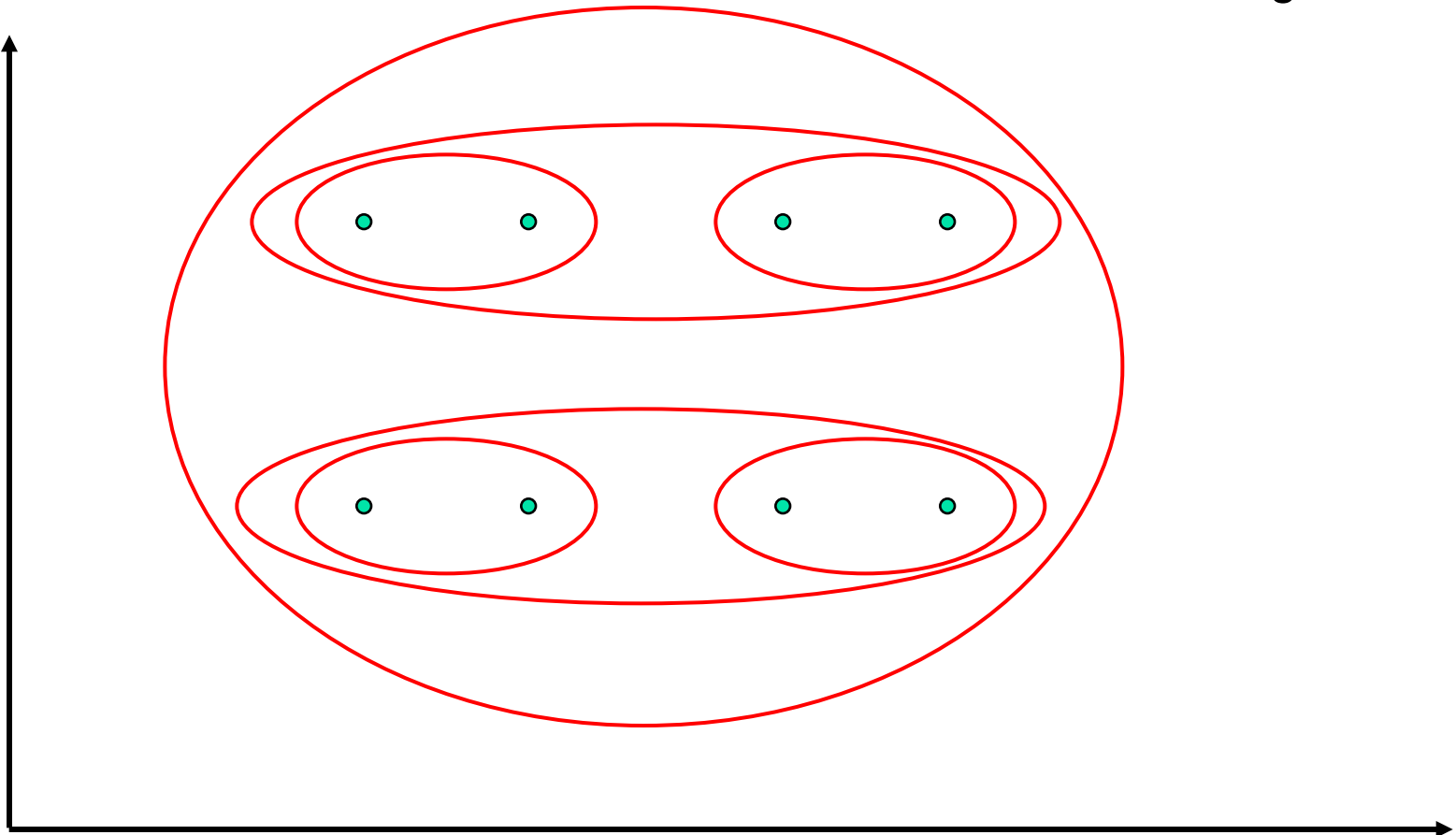
HAC – Example

- As clusters *agglomerate*, documents fall into a dendrogram.



HAC – Example

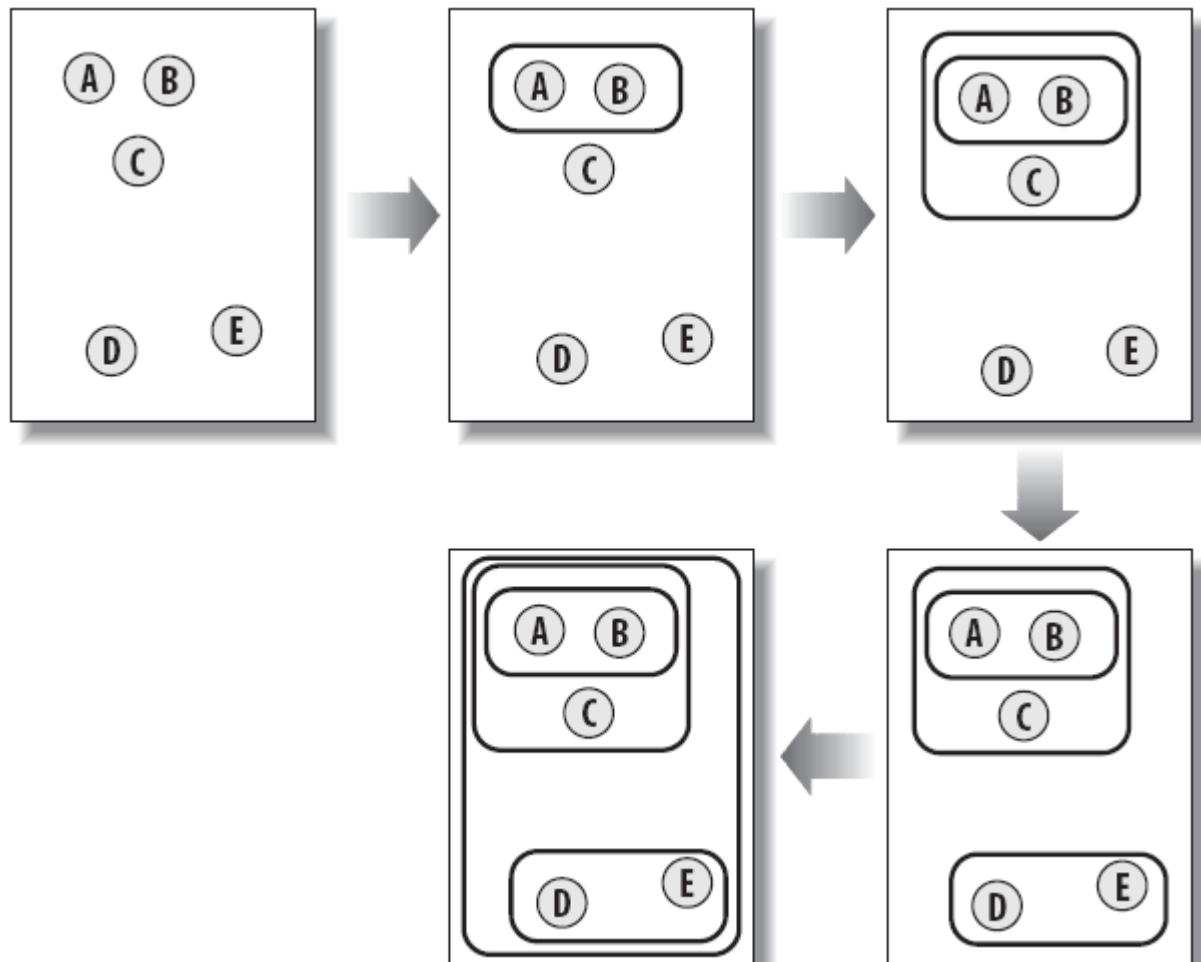
Single-Link



HAC – Exercise

Digital Camera	Megapixel	Zoom
A	1	8
B	3	8
C	2	6
D	1.5	1
E	4	2

HAC – Exercise



Chaining Effect

- Single-Link HAC can result in “straggly” (long and thin) clusters due to the chaining effect.

