Introduction to Information Retrieval

http://informationretrieval.org

IIR 17: Hierarchical Clustering

Hinrich Schütze

Institute for Natural Language Processing, Universität Stuttgart

2008.07.01

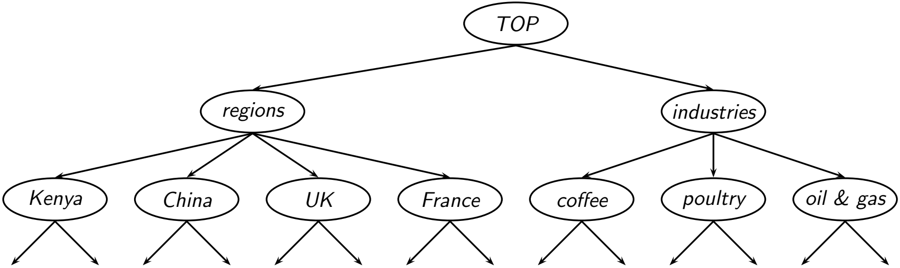
Overview

Outline

Outline

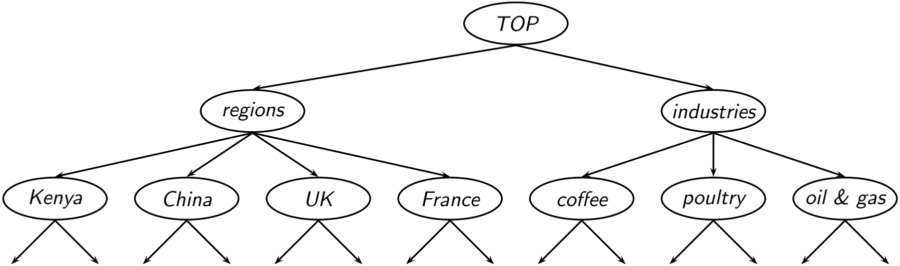
Hierarchical clustering

Our goal in hierarchical clustering is to create a hierarchy like the one we saw earlier in Reuters:



Hierarchical clustering

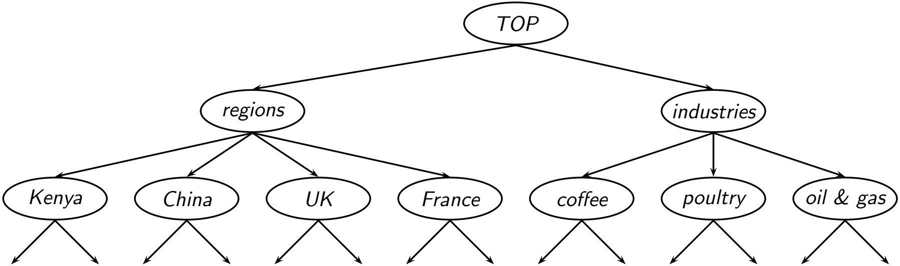
Our goal in hierarchical clustering is to create a hierarchy like the one we saw earlier in Reuters:



We want to create this hierarchy automatically.

Hierarchical clustering

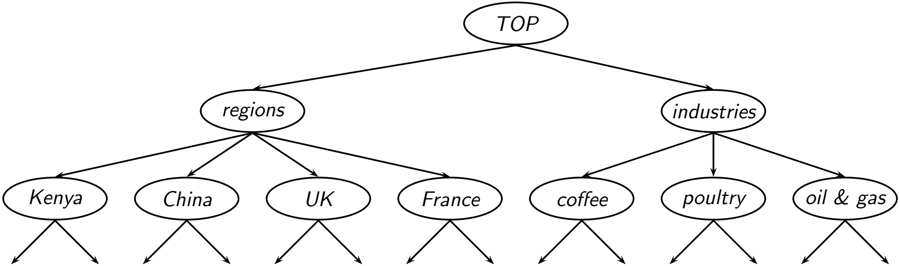
Our goal in hierarchical clustering is to create a hierarchy like the one we saw earlier in Reuters:



We want to create this hierarchy automatically.
We can do this either top-down or bottom-up.

Hierarchical clustering

Our goal in hierarchical clustering is to create a hierarchy like the one we saw earlier in Reuters:



We want to create this hierarchy automatically.

We can do this either top-down or bottom-up.

The best known bottom-up method is hierarchical agglomerative clustering.

Hierarchical agglomerative clustering (HAC)

- Assumes a similarity measure for determining the similarity of two clusters (up to now: similarity of documents).

Hierarchical agglomerative clustering (HAC)

- Assumes a similarity measure for determining the similarity of two clusters (up to now: similarity of documents).
- We will look at four different cluster similarity measures.

Hierarchical agglomerative clustering (HAC)

- Assumes a similarity measure for determining the similarity of two clusters (up to now: similarity of documents).
- We will look at four different cluster similarity measures.
- Start with each document in a separate cluster

Hierarchical agglomerative clustering (HAC)

- Assumes a similarity measure for determining the similarity of two clusters (up to now: similarity of documents).
- We will look at four different cluster similarity measures.
- Start with each document in a separate cluster
- Then repeatedly merge the two clusters that are most similar

Hierarchical agglomerative clustering (HAC)

- Assumes a similarity measure for determining the similarity of two clusters (up to now: similarity of documents).
- We will look at four different cluster similarity measures.
- Start with each document in a separate cluster
- Then repeatedly merge the two clusters that are most similar
- Until there is only one cluster

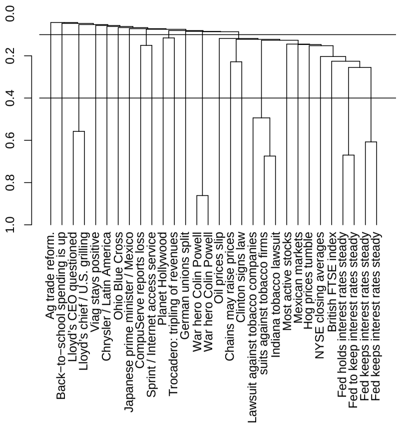
Hierarchical agglomerative clustering (HAC)

- Assumes a similarity measure for determining the similarity of two clusters (up to now: similarity of documents).
- We will look at four different cluster similarity measures.
- Start with each document in a separate cluster
- Then repeatedly merge the two clusters that are most similar
- Until there is only one cluster
- The history of merging forms a binary tree or hierarchy.

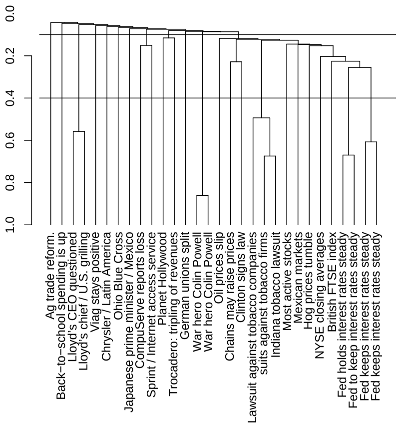
Hierarchical agglomerative clustering (HAC)

- Assumes a similarity measure for determining the similarity of two clusters (up to now: similarity of documents).
- We will look at four different cluster similarity measures.
- Start with each document in a separate cluster
- Then repeatedly merge the two clusters that are most similar
- Until there is only one cluster
- The history of merging forms a binary tree or hierarchy.
- The standard way of depicting this history is a dendrogram.

A dendrogram

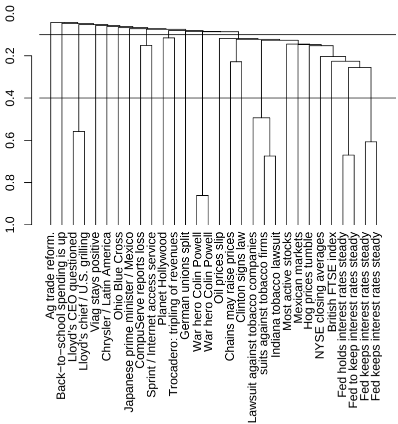


A dendrogram



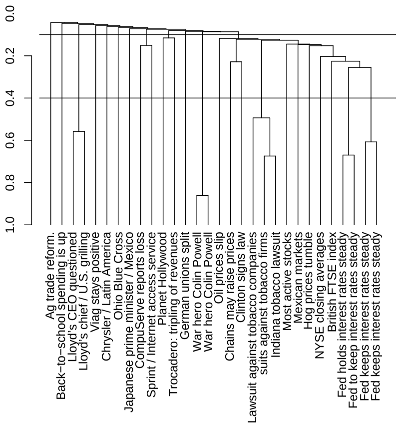
- The history of mergers can be read off from bottom to top.

A dendrogram



- The history of mergers can be read off from bottom to top.
- The horizontal line of each merger tells us what the similarity of the merger was.

A dendrogram



- The history of mergers can be read off from bottom to top.
- The horizontal line of each merger tells us what the similarity of the merger was.
- We can cut the dendrogram at a particular point (e.g., at 0.1 or 0.4) to get a flat clustering.

Divisive clustering

- Top-down (instead of bottom-up as in HAC)

Divisive clustering

- Top-down (instead of bottom-up as in HAC)
- Start with all docs in one big cluster

Divisive clustering

- Top-down (instead of bottom-up as in HAC)
- Start with all docs in one big cluster
- Then recursively split clusters

Divisive clustering

- Top-down (instead of bottom-up as in HAC)
- Start with all docs in one big cluster
- Then recursively split clusters
- Eventually each node forms a cluster on its own.

Divisive clustering

- Top-down (instead of bottom-up as in HAC)
- Start with all docs in one big cluster
- Then recursively split clusters
- Eventually each node forms a cluster on its own.
- $\rightarrow$ Bisecting K-means at the end

Divisive clustering

- Top-down (instead of bottom-up as in HAC)
- Start with all docs in one big cluster
- Then recursively split clusters
- Eventually each node forms a cluster on its own.
- → Bisecting K-means at the end
- For now: HAC

Naive HAC algorithm

$\text{SIMPLEHAC}(d_1, \ldots, d_N)$

for $n \leftarrow 1$ **to** N
do for $i \leftarrow 1$ **to** N
  do $C[n][i] \leftarrow \text{SIM}(d_n, d_i)$
  $I[n] \leftarrow 1$ *(keeps track of active clusters)*
$A \leftarrow []$ *(collects clustering as a sequence of merges)*
for $k \leftarrow 1$ **to** $N - 1$
do $\langle i, m \rangle \leftarrow \arg\max_{\{\langle i,m \rangle : i \neq m \wedge I[i]=1 \wedge I[m]=1\}} C[i][m]$
  $A.\text{APPEND}(\langle i, m \rangle)$ *(store merge)*
  for $j \leftarrow 1$ **to** N
  do $C[i][j] \leftarrow \text{SIM}(i, m, j)$
    $C[j][i] \leftarrow \text{SIM}(i, m, j)$
  $I[m] \leftarrow 0$ *(deactivate cluster)*
return A

Computational complexity of the naive algorithm

- First, we compute the similarity of all $N \times N$ pairs of documents.

Computational complexity of the naive algorithm

- First, we compute the similarity of all $N \times N$ pairs of documents.
- Then, in each iteration:

Computational complexity of the naive algorithm

- First, we compute the similarity of all $N \times N$ pairs of documents.
- Then, in each iteration:
  - We scan the $O(N \times N)$ similarities to find the maximum similarity.

Computational complexity of the naive algorithm

- First, we compute the similarity of all $N \times N$ pairs of documents.
- Then, in each iteration:
  - We scan the $O(N \times N)$ similarities to find the maximum similarity.
  - We merge the two clusters with maximum similarity.

Computational complexity of the naive algorithm

- First, we compute the similarity of all $N \times N$ pairs of documents.
- Then, in each iteration:
  - We scan the $O(N \times N)$ similarities to find the maximum similarity.
  - We merge the two clusters with maximum similarity.
  - We compute the similarity of the new cluster with all other (surviving) clusters.

Computational complexity of the naive algorithm

- First, we compute the similarity of all $N \times N$ pairs of documents.
- Then, in each iteration:
  - We scan the $O(N \times N)$ similarities to find the maximum similarity.
  - We merge the two clusters with maximum similarity.
  - We compute the similarity of the new cluster with all other (surviving) clusters.
- There are $O(N)$ iterations, each performing a $O(N \times N)$ "scan" operation.

Computational complexity of the naive algorithm

- First, we compute the similarity of all $N \times N$ pairs of documents.
- Then, in each iteration:
  - We scan the $O(N \times N)$ similarities to find the maximum similarity.
  - We merge the two clusters with maximum similarity.
  - We compute the similarity of the new cluster with all other (surviving) clusters.
- There are $O(N)$ iterations, each performing a $O(N \times N)$ "scan" operation.
- Overall complexity is $O(N^3)$.

Computational complexity of the naive algorithm

- First, we compute the similarity of all $N \times N$ pairs of documents.
- Then, in each iteration:
  - We scan the $O(N \times N)$ similarities to find the maximum similarity.
  - We merge the two clusters with maximum similarity.
  - We compute the similarity of the new cluster with all other (surviving) clusters.
- There are $O(N)$ iterations, each performing a $O(N \times N)$ "scan" operation.
- Overall complexity is $O(N^3)$.
- We'll look at more efficient algorithms later.

Key question: How to define cluster similarity

- Single-link: Maximum similarity

Key question: How to define cluster similarity

- Single-link: Maximum similarity
  - Maximum over all document pairs

Key question: How to define cluster similarity

- Single-link: Maximum similarity
  - Maximum over all document pairs
- Complete-link: Minimum similarity

Key question: How to define cluster similarity

- Single-link: Maximum similarity
  - Maximum over all document pairs
- Complete-link: Minimum similarity
  - Minimum over all document pairs

Key question: How to define cluster similarity

- Single-link: Maximum similarity
  - Maximum over all document pairs
- Complete-link: Minimum similarity
  - Minimum over all document pairs
- Centroid: Average "intersimilarity"

Key question: How to define cluster similarity

- Single-link: Maximum similarity
  - Maximum over all document pairs
- Complete-link: Minimum similarity
  - Minimum over all document pairs
- Centroid: Average "intersimilarity"
  - Average over all document pairs

Key question: How to define cluster similarity

- Single-link: Maximum similarity
  - Maximum over all document pairs
- Complete-link: Minimum similarity
  - Minimum over all document pairs
- Centroid: Average "intersimilarity"
  - Average over all document pairs
  - This is equivalent to the similarity of the centroids.

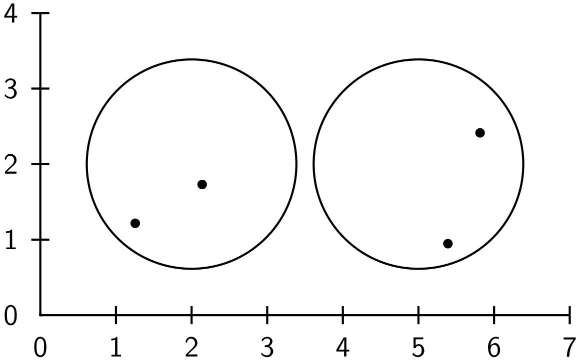
Key question: How to define cluster similarity

- Single-link: Maximum similarity
  - Maximum over all document pairs
- Complete-link: Minimum similarity
  - Minimum over all document pairs
- Centroid: Average "intersimilarity"
  - Average over all document pairs
  - This is equivalent to the similarity of the centroids.
- Group-average: Average "intrasimilarity"

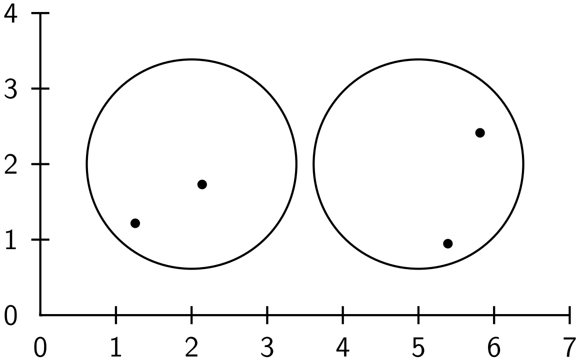
Key question: How to define cluster similarity

- Single-link: Maximum similarity
  - Maximum over all document pairs
- Complete-link: Minimum similarity
  - Minimum over all document pairs
- Centroid: Average "intersimilarity"
  - Average over all document pairs
  - This is equivalent to the similarity of the centroids.
- Group-average: Average "intrasimilarity"
  - Average over all document pairs, including pairs of docs in the same cluster

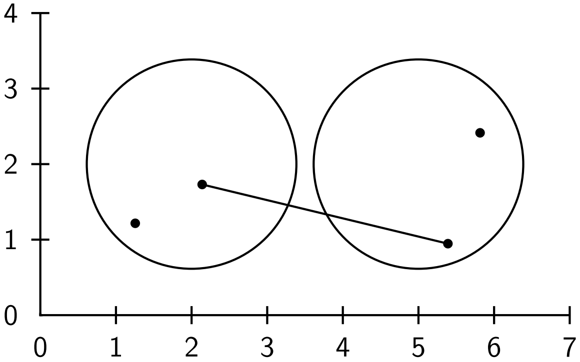
Cluster similarity: Example



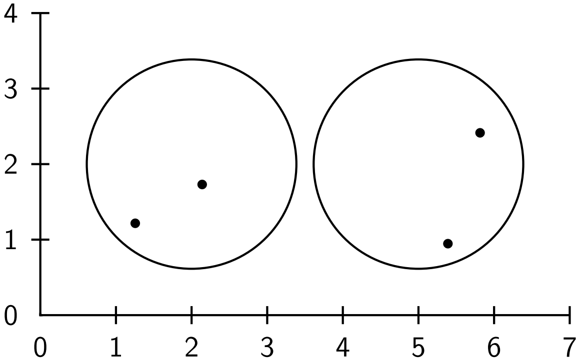
Single-link: Maximum similarity



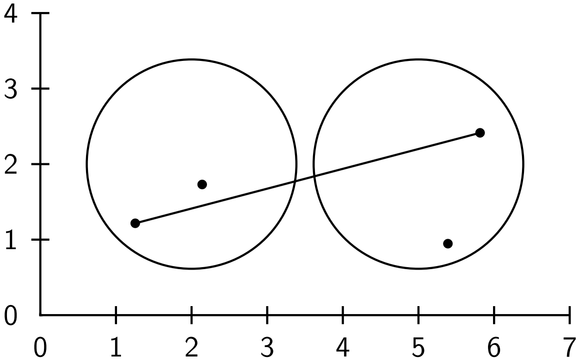
Single-link: Maximum similarity



Complete-link: Minimum similarity

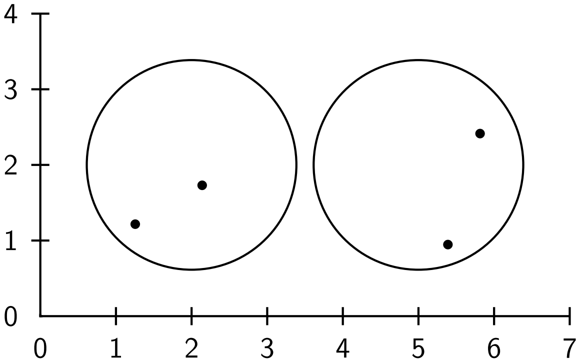


Complete-link: Minimum similarity



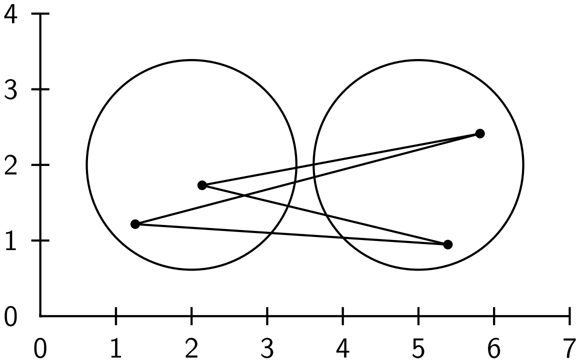
Centroid: Average intersimilarity

intersimilarity = similarity of two documents in different clusters



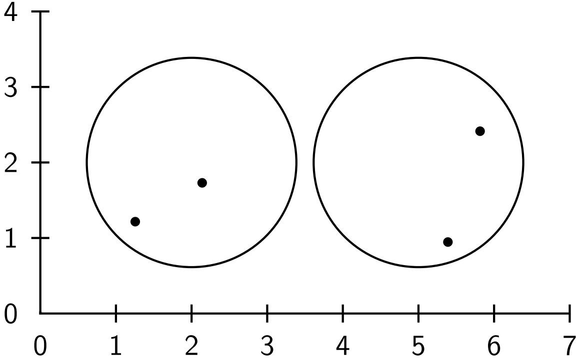
Centroid: Average intersimilarity

intersimilarity = similarity of two documents in different clusters



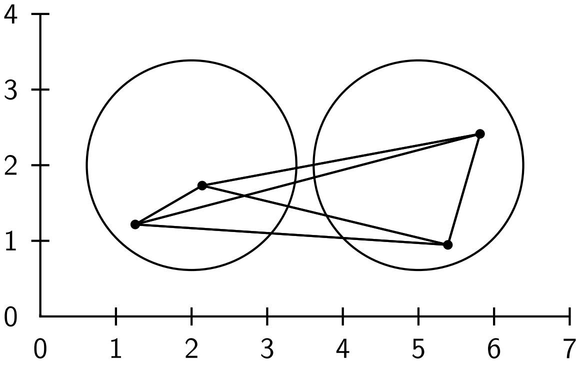
Group average: Average intrasimilarity

intrasimilarity = similarity of any pair, including those that are in cluster 1 and those that are in cluster 2

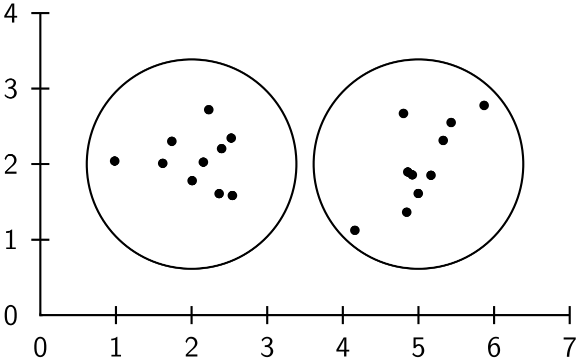


Group average: Average intrasimilarity

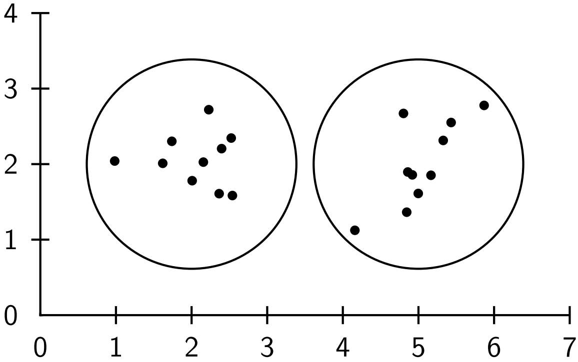
intrasimilarity = similarity of any pair, including those that are in cluster 1 and those that are in cluster 2



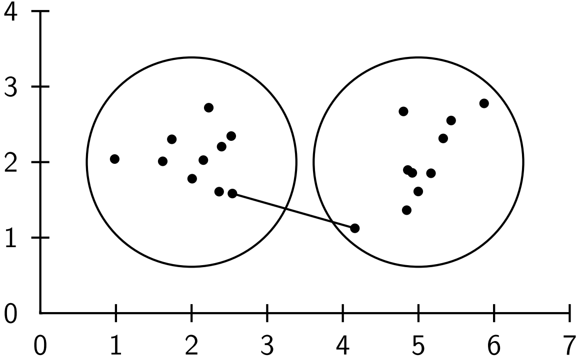
Cluster similarity: Larger example



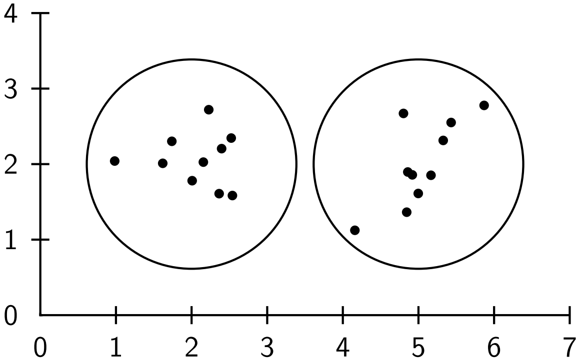
Single-link: Maximum similarity



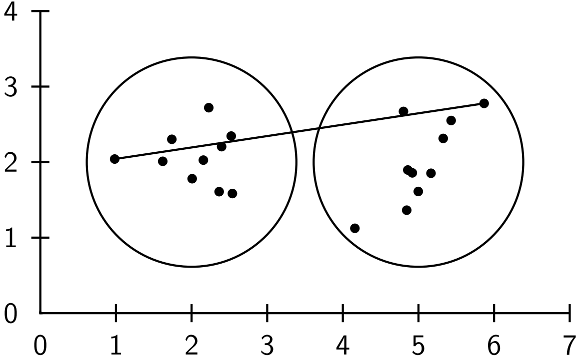
Single-link: Maximum similarity



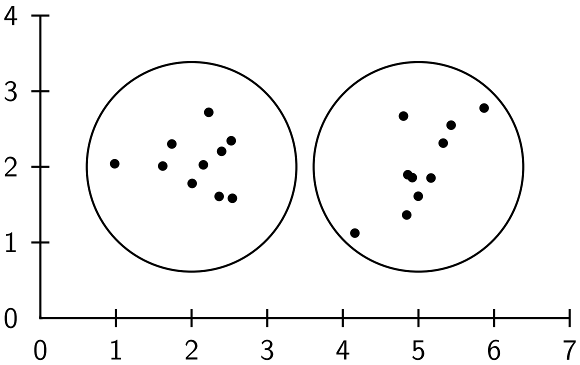
Complete-link: Minimum similarity



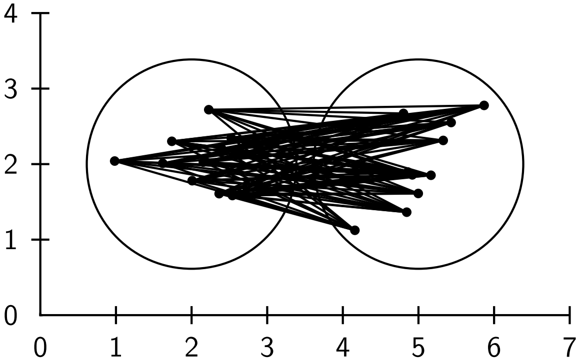
Complete-link: Minimum similarity



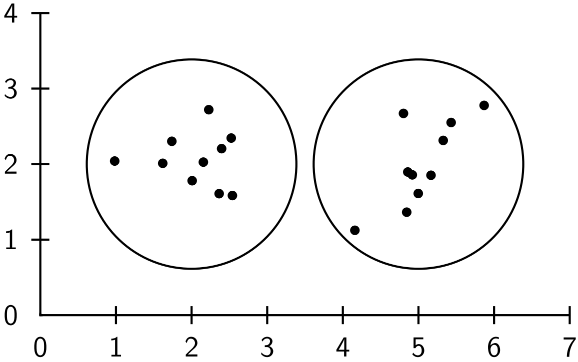
Centroid: Average intersimilarity



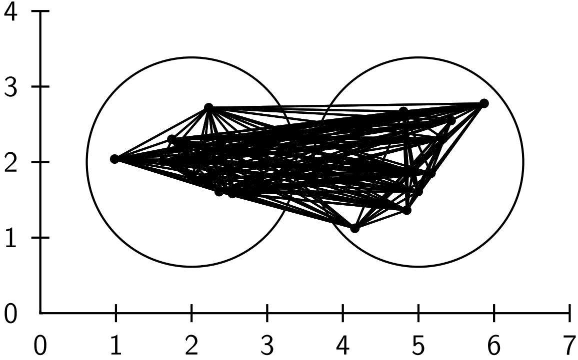
Centroid: Average intersimilarity



Group average: Average intrasimilarity



Group average: Average intrasimilarity



Outline

Single link HAC

- The similarity of two clusters is the maximum intersimilarity – the maximum similarity of a document from the first cluster and a document from the second cluster.

Single link HAC

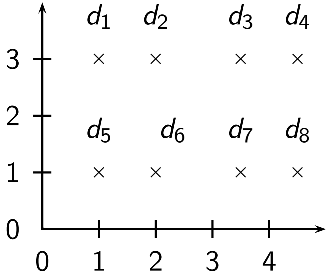
- The similarity of two clusters is the maximum intersimilarity – the maximum similarity of a document from the first cluster and a document from the second cluster.
- Once we have merged two clusters, how do we update the similarity matrix?

Single link HAC

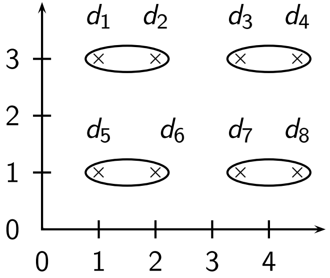
- The similarity of two clusters is the maximum intersimilarity – the maximum similarity of a document from the first cluster and a document from the second cluster.
- Once we have merged two clusters, how do we update the similarity matrix?
- This is simple for single link:

$$\text{SIM}(\omega_i, (\omega_{k_1} \cup \omega_{k_2})) = \max(\text{SIM}(\omega_i, \omega_{k_1}), \text{SIM}(\omega_i, \omega_{k_2}))$$

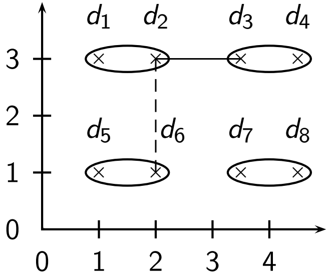
Single-link clustering: Example



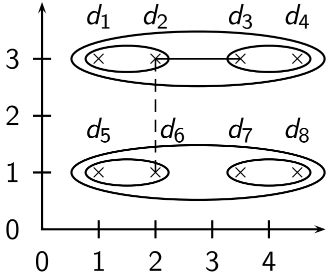
Single-link clustering: Example



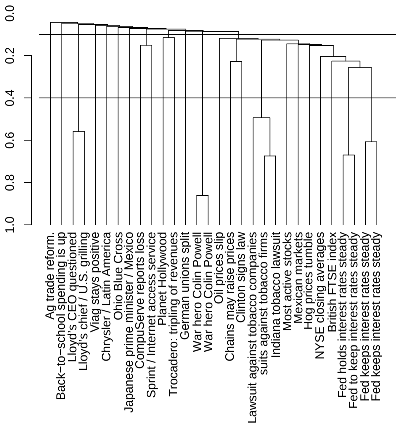
Single-link clustering: Example



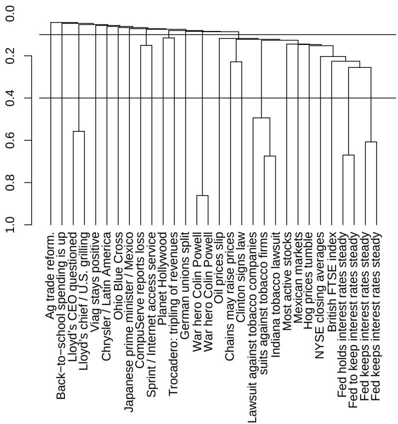
Single-link clustering: Example



This dendrogram was produced by single-link

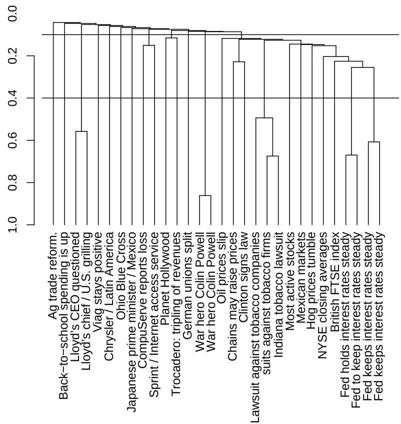


This dendrogram was produced by single-link



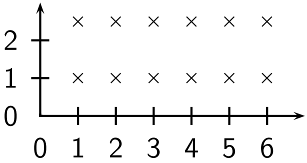
- Notice: many small clusters (1 or 2 members) being added to the main cluster

This dendrogram was produced by single-link

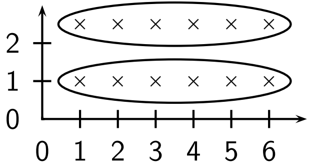


- Notice: many small clusters (1 or 2 members) being added to the main cluster
- There is no balanced 2-cluster or 3-cluster clustering that can be derived by cutting the dendrogram.

What cluster structure after 10 mergers?



Single-link: Chaining



Single-link clustering often produces long, straggly clusters. For most applications, these are undesirable.

Complete link HAC

- The similarity of two clusters is the minimum intersimilarity – the minimum similarity of a document from the first cluster and a document from the second cluster.

Complete link HAC

- The similarity of two clusters is the minimum intersimilarity – the minimum similarity of a document from the first cluster and a document from the second cluster.
- Once we have merged two clusters, how do we update the similarity matrix?

Complete link HAC

- The similarity of two clusters is the minimum intersimilarity – the minimum similarity of a document from the first cluster and a document from the second cluster.
- Once we have merged two clusters, how do we update the similarity matrix?
- Again, this is simple:

$$\text{SIM}(\omega_i, (\omega_{k_1} \cup \omega_{k_2})) = \min(\text{SIM}(\omega_i, \omega_{k_1}), \text{SIM}(\omega_i, \omega_{k_2}))$$

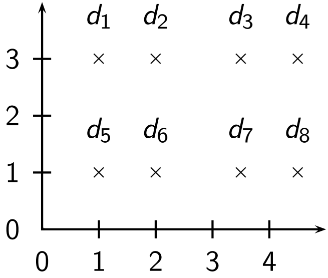
Complete link HAC

- The similarity of two clusters is the minimum intersimilarity – the minimum similarity of a document from the first cluster and a document from the second cluster.
- Once we have merged two clusters, how do we update the similarity matrix?
- Again, this is simple:

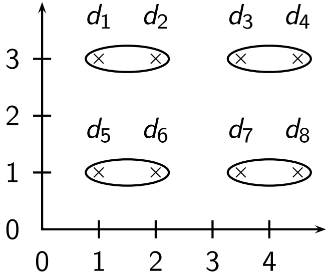
$$\text{SIM}(\omega_i, (\omega_{k_1} \cup \omega_{k_2})) = \min(\text{SIM}(\omega_i, \omega_{k_1}), \text{SIM}(\omega_i, \omega_{k_2}))$$

- We measure the similarity of two clusters by computing the radius of the cluster that we would get if we merged them.

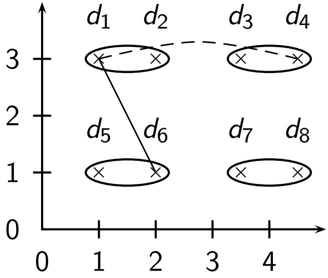
Complete link clustering: Example



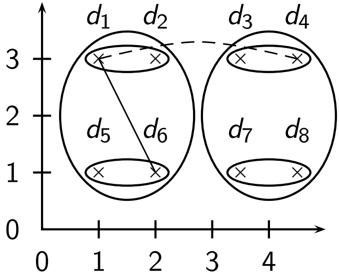
Complete link clustering: Example



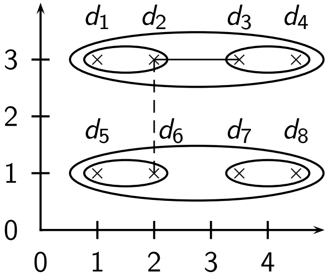
Complete link clustering: Example

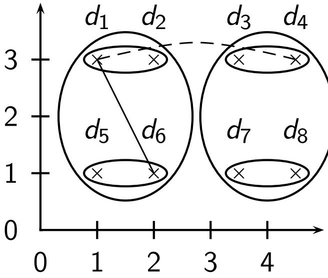


Complete link clustering: Example

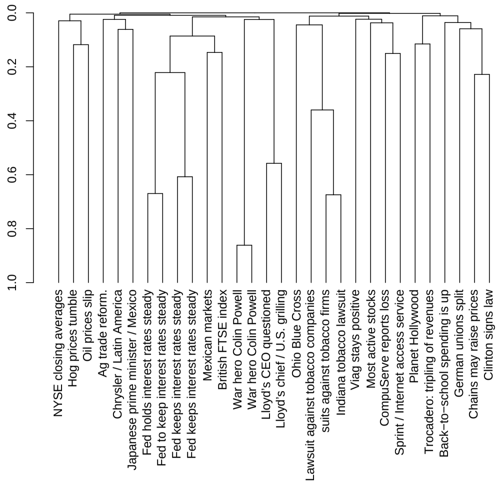


Single-link vs. Complete link clustering

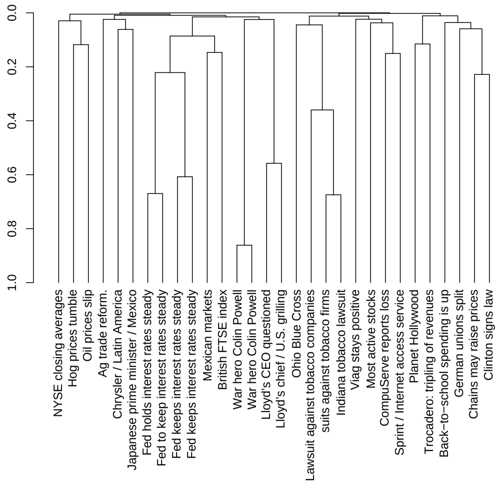




Complete-link dendrogram

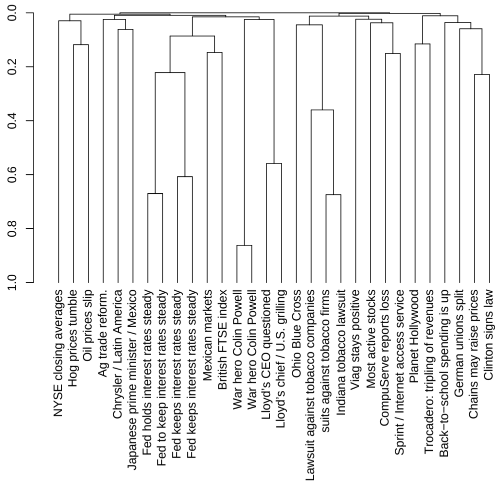


Complete-link dendrogram



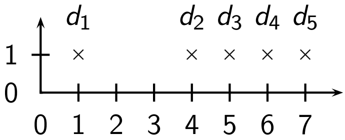
- Notice that this dendrogram is much more balanced than the single-link one.

Complete-link dendrogram



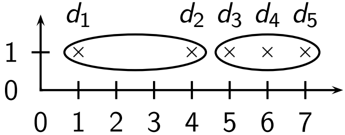
- Notice that this dendrogram is much more balanced than the single-link one.
- We can create a 2-cluster clustering with two clusters of about the same size.

Complete-link: Sensitivity to outliers



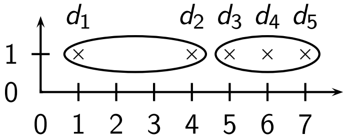
What is the intuitively best 2-cluster clustering here?

Complete-link: Sensitivity to outliers



The complete-link clustering of this set. It's not intuitive.

Complete-link: Sensitivity to outliers



The complete-link clustering of this set. It's not intuitive. This shows that a single outlier can have a large effect on the final outcome of complete-link clustering. Coordinates: $1 + 2 \times \epsilon, 4, 5 + 2 \times \epsilon, 6, 7 - \epsilon$.

Outline

Centroid HAC

- The similarity of two clusters is the average intersimilarity – the average similarity of documents from the first cluster with documents from the second cluster.

Centroid HAC

- The similarity of two clusters is the average intersimilarity – the average similarity of documents from the first cluster with documents from the second cluster.
- The above definition is inefficient ($O(N^2)$), but the definition is equivalent to computing the similarity of the centroids:

$$\text{SIM-CENT}(\omega_i, \omega_j) = \vec{\mu}(\omega_i) \cdot \vec{\mu}(\omega_j)$$

Centroid HAC

- The similarity of two clusters is the average intersimilarity – the average similarity of documents from the first cluster with documents from the second cluster.
- The above definition is inefficient ($O(N^2)$), but the definition is equivalent to computing the similarity of the centroids:

$$\text{SIM-CENT}(\omega_i, \omega_j) = \vec{\mu}(\omega_i) \cdot \vec{\mu}(\omega_j)$$

- Hence the name: centroid HAC

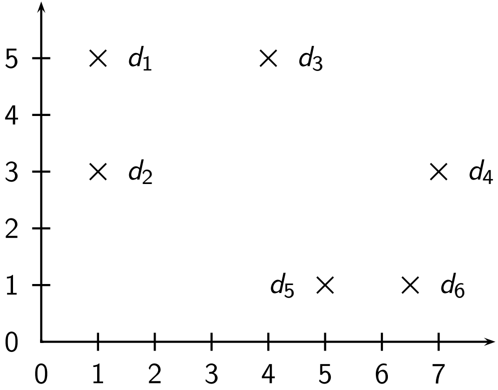
Centroid HAC

- The similarity of two clusters is the average intersimilarity – the average similarity of documents from the first cluster with documents from the second cluster.
- The above definition is inefficient ($O(N^2)$), but the definition is equivalent to computing the similarity of the centroids:

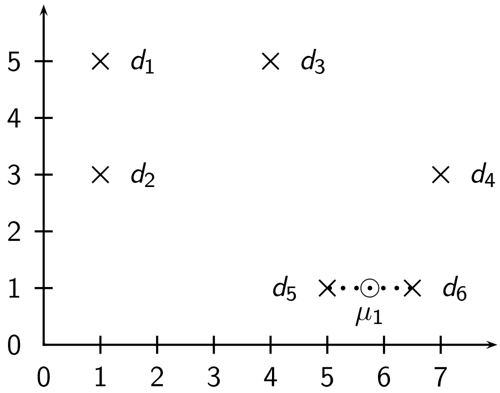
$$\text{SIM-CENT}(\omega_i, \omega_j) = \vec{\mu}(\omega_i) \cdot \vec{\mu}(\omega_j)$$

- Hence the name: centroid HAC
- Note: this is the dot product, not cosine similarity!

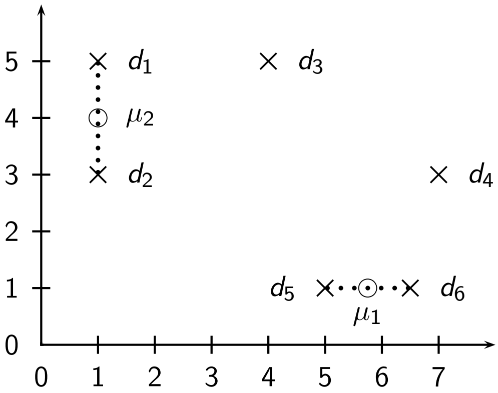
Centroid clustering: Example



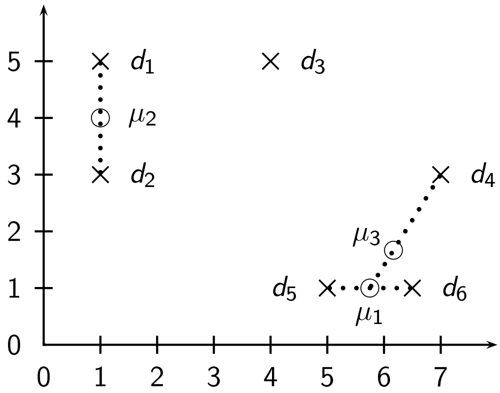
Centroid clustering: Example



Centroid clustering: Example

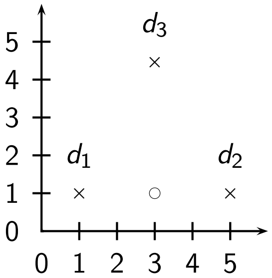


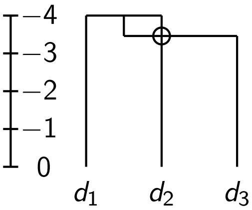
Centroid clustering: Example



Inversion in centroid clustering

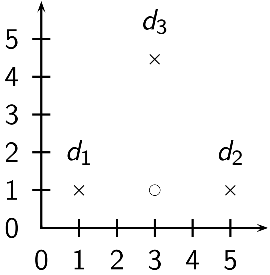
- In an inversion, the similarity increases during a merge sequence. Results in an "inverted" dendrogram.

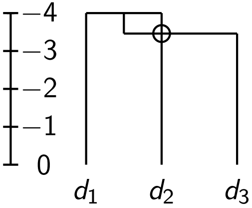




Inversion in centroid clustering

- In an inversion, the similarity increases during a merge sequence. Results in an "inverted" dendrogram.
- Below: Similarity of the first merger ($d_1 \cup d_2$) is -4.0, similarity of second merger ($(d_1 \cup d_2) \cup d_3$) is ≈ -3.5.





Inversions

- Hierarchical clustering algorithms that allow inversions are inferior.

Inversions

- Hierarchical clustering algorithms that allow inversions are inferior.
- The rationale for hierarchical clustering is that at any given point, we've found the most coherent cluster of a given size.

Inversions

- Hierarchical clustering algorithms that allow inversions are inferior.
- The rationale for hierarchical clustering is that at any given point, we've found the most coherent cluster of a given size.
- Intuitively: smaller clusters should be more coherent than larger clusters.

Inversions

- Hierarchical clustering algorithms that allow inversions are inferior.
- The rationale for hierarchical clustering is that at any given point, we've found the most coherent cluster of a given size.
- Intuitively: smaller clusters should be more coherent than larger clusters.
- An inversion contradicts this intuition: we have a large cluster that is more coherent than one of its subclusters.

Group-average agglomerative clustering (GAAC)

- GAAC also has an “average-similarity” criterion, but does not have inversions.

Group-average agglomerative clustering (GAAC)

- GAAC also has an "average-similarity" criterion, but does not have inversions.
- The similarity of two clusters is the average intrasimilarity – the average similarity of all document pairs (including those from the same cluster).

Group-average agglomerative clustering (GAAC)

- GAAC also has an “average-similarity” criterion, but does not have inversions.
- The similarity of two clusters is the average intrasimilarity – the average similarity of all document pairs (including those from the same cluster).
- But we exclude self-similarities.

Group-average agglomerative clustering (GAAC)

- Again, the above definition is inefficient ($O(N^2)$) and there is an equivalent, more efficient, centroid-based definition:

$$\text{SIM-GA}(\omega_i, \omega_j) =$$

$$\frac{1}{(N_i + N_j)(N_i + N_j - 1)}[(\sum_{d_m \in \omega_i \cup \omega_j} \vec{d}_m)^2 - (N_i + N_j)]$$

Group-average agglomerative clustering (GAAC)

- Again, the above definition is inefficient ($O(N^2)$) and there is an equivalent, more efficient, centroid-based definition:

$$\text{SIM-GA}(\omega_i, \omega_j) =$$

$$\frac{1}{(N_i + N_j)(N_i + N_j - 1)}[(\sum_{d_m \in \omega_i \cup \omega_j} \vec{d}_m)^2 - (N_i + N_j)]$$

- Again, this is the dot product, not cosine similarity.

Which HAC clustering should I use?

- Don’t use centroid HAC because of inversions.

Which HAC clustering should I use?

- Don't use centroid HAC because of inversions.
- In most cases: GAAC is best since it isn't subject to chaining and sensitivity to outliers.

Which HAC clustering should I use?

- Don't use centroid HAC because of inversions.
- In most cases: GAAC is best since it isn't subject to chaining and sensitivity to outliers.
- However, we can only use GAAC for vector representations.

Which HAC clustering should I use?

- Don't use centroid HAC because of inversions.
- In most cases: GAAC is best since it isn't subject to chaining and sensitivity to outliers.
- However, we can only use GAAC for vector representations.
- For other types of document representations (or if only pairwise similarities for document are available): use complete-link.

Which HAC clustering should I use?

- Don't use centroid HAC because of inversions.
- In most cases: GAAC is best since it isn't subject to chaining and sensitivity to outliers.
- However, we can only use GAAC for vector representations.
- For other types of document representations (or if only pairwise similarities for document are available): use complete-link.
- There are also some applications for single-link (e.g., duplicate detection in web search).

Flat or hierarchical clustering?

- For high efficiency, use flat clustering (or perhaps bisecting k-means)

Flat or hierarchical clustering?

- For high efficiency, use flat clustering (or perhaps bisecting *k*-means)
- For deterministic results: HAC

Flat or hierarchical clustering?

- For high efficiency, use flat clustering (or perhaps bisecting *k*-means)
- For deterministic results: HAC
- When a hierarchical structure is desired: hierarchical algorithm

Flat or hierarchical clustering?

- For high efficiency, use flat clustering (or perhaps bisecting k-means)
- For deterministic results: HAC
- When a hierarchical structure is desired: hierarchical algorithm
- HAC also can be applied if K cannot be predetermined (can start without knowing K)

Outline

Efficient single link clustering

$\textsc{SingleLinkClustering}(d_1, \ldots, d_N)$

for $n \leftarrow 1$ **to** N
do for $i \leftarrow 1$ **to** N
    do $C[n][i].\text{sim} \leftarrow \text{SIM}(d_n, d_i)$
        $C[n][i].\text{index} \leftarrow i$
    $I[n] \leftarrow n$
    $NBM[n] \leftarrow \arg\max_{X \in \{C[n][i]:n \neq i\}} X.\text{sim}$
$A \leftarrow []$
for $n \leftarrow 1$ **to** $N - 1$
do $i_1 \leftarrow \arg\max_{\{i:I[i]=i\}} NBM[i].\text{sim}$
    $i_2 \leftarrow I[NBM[i_1].\text{index}]$
    $A.\textsc{Append}(\langle i_1, i_2 \rangle)$
    for $i \leftarrow 1$ **to** N
    do if $I[i] = i \wedge i \neq i_1 \wedge i \neq i_2$
            then $C[i_1][i].\text{sim} \leftarrow C[i][i_1].\text{sim} \leftarrow \max(C[i_1][i].\text{sim}, C[i_2][i].\text{sim})$
        if $I[i] = i_2$
            then $I[i] \leftarrow i_1$
    $NBM[i_1] \leftarrow \arg\max_{X \in \{C[i_1][i]:I[i]=i \wedge i \neq i_1\}} X.\text{sim}$
return A

Time complexity of HAC

- The single-link algorithm we just saw is $O(N^2)$.

Time complexity of HAC

- The single-link algorithm we just saw is $O(N^2)$.
- Much more efficient than the $O(N^3)$ algorithm we looked at earlier!

Time complexity of HAC

- The single-link algorithm we just saw is $O(N^2)$.
- Much more efficient than the $O(N^3)$ algorithm we looked at earlier!
- There is no known $O(N^2)$ algorithm for complete-link, centroid and GAAC.

Time complexity of HAC

- The single-link algorithm we just saw is $O(N^2)$.
- Much more efficient than the $O(N^3)$ algorithm we looked at earlier!
- There is no known $O(N^2)$ algorithm for complete-link, centroid and GAAC.
- Best time complexity for these three is $O(N^2 \log N)$: See book.

Time complexity of HAC

- The single-link algorithm we just saw is $O(N^2)$.
- Much more efficient than the $O(N^3)$ algorithm we looked at earlier!
- There is no known $O(N^2)$ algorithm for complete-link, centroid and GAAC.
- Best time complexity for these three is $O(N^2 \log N)$: See book.
- In practice: little difference between $O(N^2 \log N)$ and $O(N^2)$.

Combination similarities of the four algorithms

clustering algorithm	$\text{sim}(\ell, k_1, k_2)$
single-link	$\max(\text{sim}(\ell, k_1), \text{sim}(\ell, k_2))$
complete-link	$\min(\text{sim}(\ell, k_1), \text{sim}(\ell, k_2))$
centroid	$(\frac{1}{N_m}\vec{v}_m) \cdot (\frac{1}{N_\ell}\vec{v}_\ell)$
group-average	$\frac{1}{(N_m+N_\ell)(N_m+N_\ell-1)}[(\vec{v}_m + \vec{v}_\ell)^2 - (N_m + N_\ell)]$

Comparison of HAC algorithms

method	combination similarity	time compl.	optimal?	comment
single-link	max intersimilarity of any 2 docs	$\Theta(N^2)$	yes	chaining effect
complete-link	min intersimilarity of any 2 docs	$\Theta(N^2 \log N)$	no	sensitive to outliers
group-average	average of all sims	$\Theta(N^2 \log N)$	no	best choice for most applications
centroid	average intersimilarity	$\Theta(N^2 \log N)$	no	inversions can occur

What to do with the hierarchy?

- Use as is (e.g., for browsing as in Yahoo hierarchy)

What to do with the hierarchy?

- Use as is (e.g., for browsing as in Yahoo hierarchy)
- Cut at a predetermined threshold

What to do with the hierarchy?

- Use as is (e.g., for browsing as in Yahoo hierarchy)
- Cut at a predetermined threshold
- Cut to get a predetermined number of clusters K

What to do with the hierarchy?

- Use as is (e.g., for browsing as in Yahoo hierarchy)
- Cut at a predetermined threshold
- Cut to get a predetermined number of clusters K
- Hierarchical clustering is often used to get K flat clusters. The hierarchy is then ignored.

K-means vs. HAC

- Consider running 2-means clustering on a corpus, each doc of which is from one of two different languages.

K-means vs. HAC

- Consider running 2-means clustering on a corpus, each doc of which is from one of two different languages.
- What are the two clusters we would expect to see?

K-means vs. HAC

- Consider running 2-means clustering on a corpus, each doc of which is from one of two different languages.
- What are the two clusters we would expect to see?
- Is HAC likely to produce results different from the above?

Bisecting K-means: A top-down algorithm

- Start with all documents in one cluster

Bisecting K-means: A top-down algorithm

- Start with all documents in one cluster
- Split the cluster into 2 using K-means

Bisecting K-means: A top-down algorithm

- Start with all documents in one cluster
- Split the cluster into 2 using K-means
- Of the clusters produced so far, select one to split (e.g. select the largest one)

Bisecting K-means: A top-down algorithm

- Start with all documents in one cluster
- Split the cluster into 2 using K-means
- Of the clusters produced so far, select one to split (e.g. select the largest one)
- Repeat until we have produced the desired number of clusters

Bisecting K-means

$\textsc{BisectingKMeans}(d_1, \ldots, d_N)$

$\omega_0 \leftarrow \{\vec{d}_1, \ldots, \vec{d}_N\}$

leaves $\leftarrow \{\omega_0\}$

for $k \leftarrow 1$ **to** $K - 1$

do $\omega_k \leftarrow \textsc{PickClusterFrom}(\mathit{leaves})$

$\quad \{\omega_i, \omega_j\} \leftarrow \textsc{KMeans}(\omega_k, 2)$

$\quad \mathit{leaves} \leftarrow \mathit{leaves} \setminus \{\omega_k\} \cup \{\omega_i, \omega_j\}$

return *leaves*

Bisecting K-means

- If we don't generate a complete hierarchy, then a top-down algorithm like bisecting K-means is much more efficient than HAC algorithms.

Bisecting K-means

- If we don't generate a complete hierarchy, then a top-down algorithm like bisecting K-means is much more efficient than HAC algorithms.
- But bisecting K-means is not deterministic.

Bisecting K-means

- If we don't generate a complete hierarchy, then a top-down algorithm like bisecting K-means is much more efficient than HAC algorithms.
- But bisecting K-means is not deterministic.
- Why?

Bisecting K-means

- If we don't generate a complete hierarchy, then a top-down algorithm like bisecting K-means is much more efficient than HAC algorithms.
- But bisecting K-means is not deterministic.
- Why?
- There are deterministic versions, see below – but they are much less efficient.

Outline

Major issue in clustering – labeling

- After a clustering algorithm finds a set of clusters: how can they be useful to the end user?

Major issue in clustering – labeling

- After a clustering algorithm finds a set of clusters: how can they be useful to the end user?
- We need a pithy label for each cluster.

Major issue in clustering – labeling

- After a clustering algorithm finds a set of clusters: how can they be useful to the end user?
- We need a pithy label for each cluster.
- For example, in search result clustering for "jaguar": "animal", "car", "operating system"

Major issue in clustering – labeling

- After a clustering algorithm finds a set of clusters: how can they be useful to the end user?
- We need a pithy label for each cluster.
- For example, in search result clustering for "jaguar": "animal", "car", "operating system"
- How can we do this?

Discriminative labeling

- To label cluster ω, compare ω with all other clusters

Discriminative labeling

- To label cluster ω, compare ω with all other clusters
- Find terms or phrases that distinguish ω from the other clusters

Discriminative labeling

- To label cluster ω, compare ω with all other clusters
- Find terms or phrases that distinguish ω from the other clusters
- We can use any of the feature selection criteria we introduced in text classification to identify discriminating terms: mutual information, χ^2 and frequency.

Discriminative labeling

- To label cluster ω, compare ω with all other clusters
- Find terms or phrases that distinguish ω from the other clusters
- We can use any of the feature selection criteria we introduced in text classification to identify discriminating terms: mutual information, χ^2 and frequency.
- (but the latter is actually not discriminative)

Non-discriminative labeling

- Select terms or phrases based solely on information from the cluster itself

Non-discriminative labeling

- Select terms or phrases based solely on information from the cluster itself
- Terms with high weights in the centroid (if we are using a vector space model)

Non-discriminative labeling

- Select terms or phrases based solely on information from the cluster itself
- Terms with high weights in the centroid (if we are using a vector space model)
- Non-discriminative methods sometimes select frequent terms that do not distinguish clusters.

Non-discriminative labeling

- Select terms or phrases based solely on information from the cluster itself
- Terms with high weights in the centroid (if we are using a vector space model)
- Non-discriminative methods sometimes select frequent terms that do not distinguish clusters.
- For example, MONDAY, TUESDAY, . . . in newspaper text

Using titles for labeling clusters

- Terms and phrases are hard to scan and condense into a holistic idea of what the cluster is about.

Using titles for labeling clusters

- Terms and phrases are hard to scan and condense into a holistic idea of what the cluster is about.
- Alternative: titles

Using titles for labeling clusters

- Terms and phrases are hard to scan and condense into a holistic idea of what the cluster is about.
- Alternative: titles
- For example, the titles of two or three documents that are closest to the centroid.

Using titles for labeling clusters

- Terms and phrases are hard to scan and condense into a holistic idea of what the cluster is about.
- Alternative: titles
- For example, the titles of two or three documents that are closest to the centroid.
- Titles are easier to scan than a list of phrases.

Cluster labeling: Example

		labeling method		
	# docs	centroid	mutual information	title
4	622	oil plant mexico production crude **power 000 refinery gas** bpd	plant oil production **barrels** crude bpd mexico **dolly capacity petroleum**	MEXICO: Hurricane Dolly heads for Mexico coast
9	1017	police security **russian** people military peace killed told **grozny court**	police killed military security peace told **troops forces rebels** people	RUSSIA: Russia's Lebed meets rebel chief in Chechnya
10	1259	00 000 tonnes traders futures wheat prices **cents september** tonne	**delivery** traders futures tonne tonnes **desk** wheat prices 000 00	USA: Export Business - Grain/oilseeds complex

- Three methods: most prominent terms in centroid, differential labeling using MI, title of doc closest to centroid

Cluster labeling: Example

	# docs	labeling method		
		centroid	mutual information	title
4	622	oil plant mexico production crude **power 000 refinery gas** bpd	plant oil production **barrels** crude bpd mexico **dolly capacity petroleum**	MEXICO: Hurricane Dolly heads for Mexico coast
9	1017	police security **russian** people military peace killed told **grozny court**	police killed military security peace told **troops forces rebels** people	RUSSIA: Russia's Lebed meets rebel chief in Chechnya
10	1259	00 000 tonnes traders futures wheat prices **cents september** tonne	**delivery** traders futures tonne tonnes **desk** wheat prices 000 00	USA: Export Business - Grain/oilseeds complex

- Three methods: most prominent terms in centroid, differential labeling using MI, title of doc closest to centroid
- All three methods do a pretty good job.

Resources

- Chapter 17 of IIR

Resources

- Chapter 17 of IIR
- Resources at `http://ifnlp.org/ir`

Resources

- Chapter 17 of IIR
- Resources at `http://ifnlp.org/ir`
- Columbia Newsblaster (a precursor of Google News): McKeown et al. (2002)

Resources

- Chapter 17 of IIR
- Resources at `http://ifnlp.org/ir`
- Columbia Newsblaster (a precursor of Google News): McKeown et al. (2002)
- Bisecting K-means clustering: Steinbach et al. (2000)

Resources

- Chapter 17 of IIR
- Resources at `http://ifnlp.org/ir`
- Columbia Newsblaster (a precursor of Google News): McKeown et al. (2002)
- Bisecting K-means clustering: Steinbach et al. (2000)
- PDDP (similar to bisecting K-means; deterministic, but also less efficient): Saravesi and Boley (2004)