Introduction to

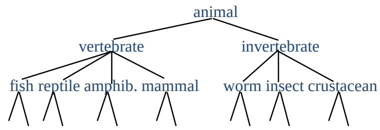
Information Retrieval

CS276: Information Retrieval and Web Search
Pandu Nayak and Prabhakar Raghavan

IR Book, Chapter 17: Hierarchical Clustering
(Slides modified by Scott Sanner)

Hierarchical Clustering

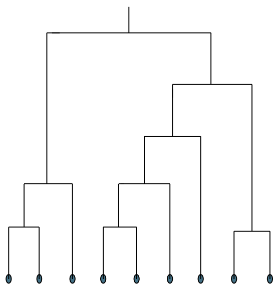
- Build a tree-based hierarchical taxonomy (*dendrogram*) from a set of documents.



- One approach: recursive application of a partitional clustering algorithm.

Dendrogram: Hierarchical Clustering

- Clustering obtained by cutting the dendrogram at a desired level: each connected component forms a cluster.



Hierarchical Agglomerative Clustering (HAC)

- Starts with each doc in a separate cluster
  - then repeatedly joins the <u>*closest pair*</u> of clusters, until there is only one cluster.
- The history of merging forms a binary tree or hierarchy.

Note: the resulting clusters are still "hard" and induce a partition

Closest pair of clusters

- Many variants to defining closest pair of clusters
- **Single-link**
  - Similarity of the *most* cosine-similar (single-link)
- **Complete-link**
  - Similarity of the "furthest" points, the *least* cosine-similar
- **Centroid**
  - Clusters whose centroids (centers of gravity) are the most cosine-similar
- **Average-link**
  - Average cosine between pairs of elements

Single Link Agglomerative Clustering

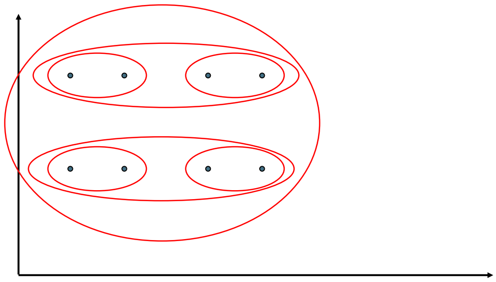
- Use maximum similarity of pairs:

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

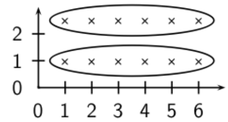
- Can result in "straggly" (long and thin) clusters due to chaining effect.
- After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:

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

Single Link Example

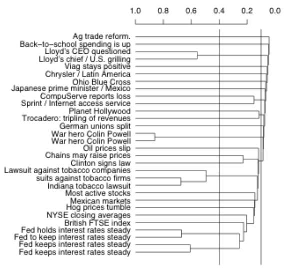


Single-link: Chaining



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

This dendogram 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.

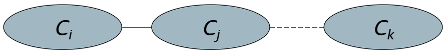
Complete Link

- Use minimum similarity of pairs:

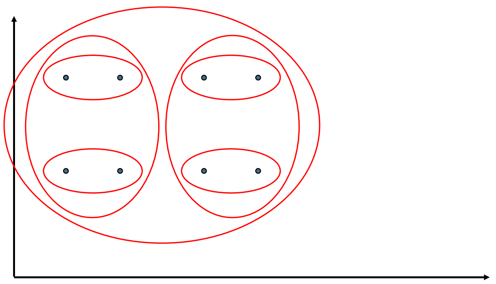
$$sim(c_i, c_j) = \min_{x \in c_i, y \in c_j} sim(x, y)$$

- Makes "tighter," spherical clusters that are typically preferable.
- After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:

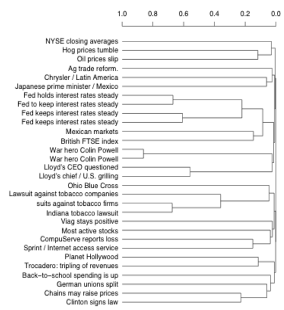
$$sim((c_i \cup c_j), c_k) = \min(sim(c_i, c_k), sim(c_j, c_k))$$



Complete Link Example



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.

Computational Complexity

- In the first iteration, all HAC methods need to compute similarity of all pairs of N initial instances, 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.
- In order to maintain an overall $O(N^2)$ performance, computing similarity to each other cluster must be done in constant time.
  - Often $O(N^3)$ if done naively or $O(N^2 \log N)$ if done more cleverly

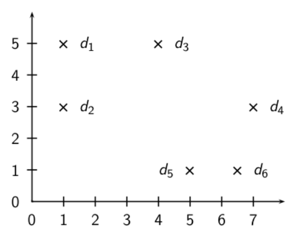
Centroid HAC

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

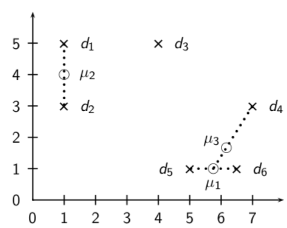
$$\text{SIM-CENT}(\omega_i, \omega_j) = \vec{\mu}(\omega_i) \cdot \vec{\mu}(\omega_j)$$

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

Exercise: Compute centroid clustering

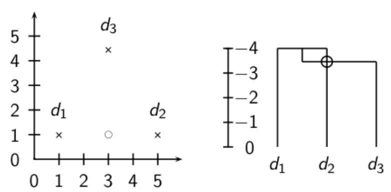


Centroid clustering



The 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.
- The rationale for hierarchical clustering is that at any given point, we've found the most coherent clustering of a given size.
- Intuitively: smaller clusterings should be more coherent than larger clusterings.
- 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
- Similarity of two clusters = average similarity of all pairs within merged cluster.

$$sim(c_i, c_j) = \frac{1}{|c_i \cup c_j|(|c_i \cup c_j| - 1)} \sum_{\vec{x} \in (c_i \cup c_j)} \sum_{\vec{y} \in (c_i \cup c_j): \vec{y} \neq \vec{x}} sim(\vec{x}, \vec{y})$$

- Compromise between single and complete link.
- Two options:
  - Averaged across all ordered pairs in the merged cluster
  - Averaged over all pairs *between* the two original clusters
- No clear difference in efficacy

Computing Group Average Similarity

- Always maintain sum of vectors in each cluster.

$$\vec{s}(c_j) = \sum_{\vec{x} \in c_j} \vec{x}$$

- Can compute similarity of clusters in **constant time**!

$$sim(c_i, c_j) = \frac{(\vec{s}(c_i) + \vec{s}(c_j)) \bullet (\vec{s}(c_i) + \vec{s}(c_j)) - (|c_i| + |c_j|)}{(|c_i| + |c_j|)(|c_i| + |c_j| - 1)}$$

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)
- 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

1. Recap
2. Introduction
3. Single-link/ Complete-link
4. Centroid/ GAAC
5. Variants
6. Labeling clusters

Efficient single link clustering

```
SingleLinkClustering(d_1, ..., d_N)
for n ← 1 to N
do for i ← 1 to N
    do C[n][i].sim ← SIM(d_n, d_i)
        C[n][i].index ← i
    I[n] ← n
    NBM[n] ← arg max_{X∈{C[n][i]:n≠i}} X.sim
A ← []
for n ← 1 to N − 1
do i_1 ← arg max_{i:I[i]=i} NBM[i].sim
    i_2 ← I[NBM[i_1].index]
    A.Append(⟨i_1, i_2⟩)
    for i ← 1 to N
    do if I[i] = i ∧ i ≠ i_1 ∧ i ≠ i_2
          then C[i_1][i].sim ← C[i][i_1].sim ← max(C[i_1][i].sim, C[i_2][i].sim)
       if I[i] = i_2
          then I[i] ← i_1
    NBM[i_1] ← arg max_{X∈{C[i_1][i]:I[i]=i∧i≠i_1}} X.sim
return A
```

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 $O(N^2)$.

 25

Combination similarities of the four algorithms

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

 26

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

 27

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
  - Ignores hierarchy below and above cutting line.

 28

Divisive Clustering: 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

 29

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 $\omega_k \leftarrow \textsc{PickClusterFrom}(\textit{leaves})$

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

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

return *leaves*

 30

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.
- There are deterministic versions of bisecting K-means (see resources at the end), 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?
- We need a pithy label for each cluster.
- For example, in search result clustering for "jaguar", The labels of the three clusters could be "animal", "car", and "operating system".
- Topic of this section: How can we automatically find good labels for clusters?

Exercise

- Come up with an algorithm for labeling clusters
- Input: a set of documents, partitioned into K clusters (flat clustering)
- Output: A label for each cluster
- Part of the exercise: What types of labels should we consider? Words?

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
- 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.
- 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
- All three methods do a pretty good job.

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)