1. Basics and Algorithms
2. K-means Clustering
3. Hierarchical Clustering
4. DBSCAN Clustering
5. Issues: Evaluation, Scalability, Comparison

1. Basics and Algorithms

Cluster: It is said to be "Collection of data objects in which the objects similar to one another within the same cluster and dissimilar to the objects in other clusters ".

E.g. *Amazon.com has different group of items i.e. Electronics, Footwear, Bags etc. where items are referred as Cluster*

Clustering is: -

- unsupervised (descriptive or undirected) classification - *describe hidden structure from "unlabelled" data (a classification or categorization is not included in the observations).*
- the process of grouping physical or abstract objects into classes of similar objects
- help in construct meaningful partitioning of a large set of objects

Cluster analysis is the process of finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters.

- A good clustering method will produce high quality clusters with
  - high intra-class similarity
  - low inter-class similarity
- The quality of a clustering result depends on both the similarity measure used by the method and its implementation.
- The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns.

Requirements of Clustering in Data Mining

The following points throw light on why clustering is required in data mining –

- **Scalability** – We need highly scalable clustering algorithms to deal with large databases.
- **Ability to deal with different kinds of attributes** – Algorithms should be capable to be applied on any kind of data such as interval-based (numerical) data, categorical, and binary data.
- **Discovery of clusters with attribute shape** – The clustering algorithm should be capable of detecting clusters of arbitrary shape. They should not be bounded to only distance measures that tend to find spherical cluster of small sizes.
- **High dimensionality** – The clustering algorithm should not only be able to handle low-dimensional data but also the high dimensional space.
- **Ability to deal with noisy data** – Databases contain noisy, missing or erroneous data. Some algorithms are sensitive to such data and may lead to poor quality clusters.
- **Interpretability** – The clustering results should be interpretable, comprehensible, and usable.

Types of similarities of clustering's are:

- Intra-class similarity - Objects are similar to objects in same cluster
- Inter-class dissimilarity - Objects are dissimilar to objects in other clusters

Types of Clustering Algorithms

I. Partitioning Method: Partition the database into k clusters which are represented by representative objects of them. Construct a partition of a database D of n objects into a set of k clusters, such that we have the minimum sum of squared distance
Given a k, find a partition of k clusters that optimizes the chosen partitioning criterion
Heuristic methods: k-means and k-medoids algorithms
   - **k-means**: Each cluster is represented by the center of the cluster
   - **k-medoids** or PAM (Partition around medoids): Each cluster is represented by one of the objects in the cluster

II. Hierarchical Method: Decompose the database into several levels of partitioning which are represented by dendrogram
The hierarchical clustering are of two types
   i. **Agglomerative or Bottom-up**: Agglomerative starts with as many clusters as there are records, with each cluster having only one record. Then pairs of clusters are successively merged until the number of clusters reduces to k. at each stage, the pair of clusters are merged which are nearest to each other. If the merging is continued, it terminates in the hierarchy of clusters which is built with just a single cluster containing all the records.
   ii. **Divisive or Top-down**: Divisive algorithm takes the opposite approach from the agglomerative techniques. These starts with all the records in one cluster, and then try to split tat clusters into smaller pieces.

Clustering Applications

- **Marketing**: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- **Land use:** Identification of areas of similar land use in an earth observation database
- **Insurance**: Identifying groups of motor insurance policy holders with a high average claim cost
- **City-planning**: Identifying groups of houses according to their house type, value, and geographical location
- **Earth-quake studies**: Observed earth quake epicenters should be clustered along continent faults
- **Psychiatry:** where the characterization of patients on the basis of clusters of symptoms can be useful in the identification of an appropriate form of therapy
- **Data Mining**: cluster analysis serves as a tool to gain insight into the distribution of data to observe characteristics of each cluster.
- **Pattern recognition**: outlier detection applications such as detection of credit card fraud.
- Image analysis
- **Bioinformatics**: used to derive plant and animal taxonomies, categorize genes with similar functionalities and gain insight into structures inherent to populations.
- Machine Learning, Voice mining, Image processing, Text mining
- **Web cluster engines**: classifying documents on the web for information discovery.
- Whether report analysis

2. K-means Clustering

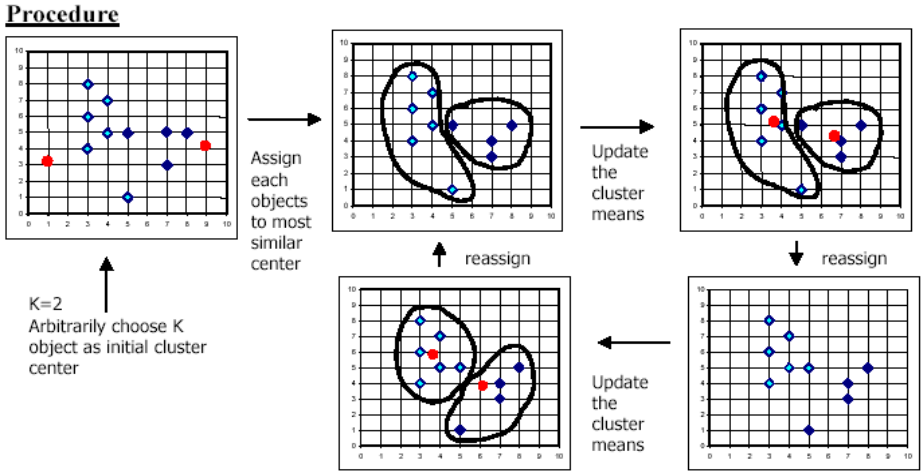
- K-Means algorithm is a type of partitioning method
- Group instances based on attributes into k groups
- High intra-cluster similarity; Low inter-cluster similarity
- Cluster similarity is measured in regards to the mean value of objects in the cluster.
  - First, select K random instances from the data – initial cluster centers
  - Second, each instance is assigned to its closest (most similar) cluster center
  - Third, each cluster center is updated to the mean of its constituent instances
  - Repeat steps two and three till there is no further change in assignment of instances to clusters

K-Means Algorithm Properties

- There are always K clusters.
- There is always at least one item in each cluster.
- The clusters are non-hierarchical and they do not overlap.
- Every member of a cluster is closer to its cluster than any other cluster because closeness does not always involve the 'center' of clusters.

The K-Means Algorithm Process

- Chose k number of clusters to be determined
- Chose k objects randomly as the initial cluster centers
- Repeat
  - Assign each object to their closest cluster center, using Euclidean distance, $d = \{(x2-x1)^2 + (y2-y1)^2\}^{1/2}$
  - Compute new cluster centers, calculate mean point
- Until
  - No change in cluster centers or
  - No object changes its clusters



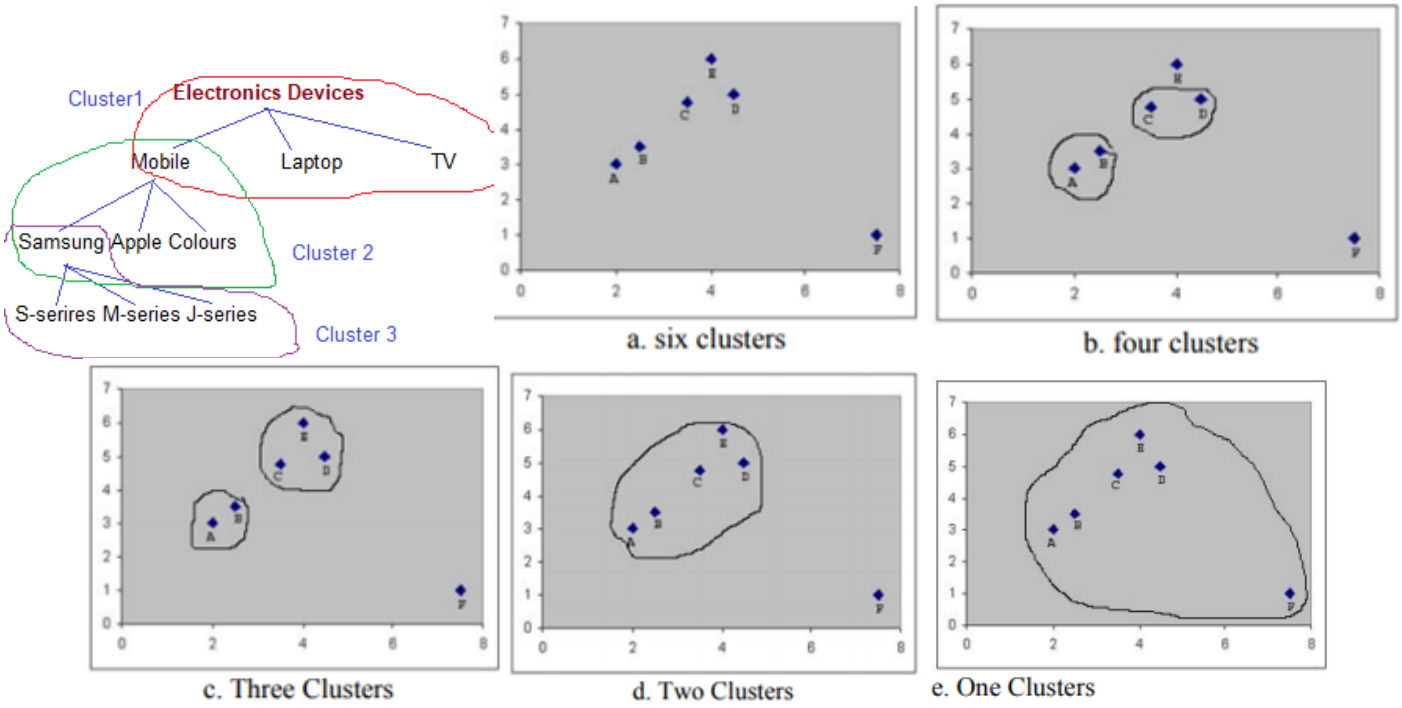
Drawbacks of K-Means

- Applicable only when *mean* is defined, then what about categorical data?
- Need to specify *K,* the *number* of clusters, in advance
  - run the algorithm with different K values
- Unable to handle noisy data and *outliers*
- Works best when clusters are of approximately of equal size

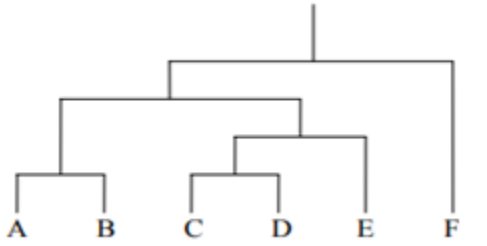
<... Example of K-means clustering: Follow in Class Note >

3. Hierarchical Clustering

This method creates a hierarchical decomposition or breakdown of the given set of data objects. We can classify hierarchical methods on the basis of how the hierarchical decomposition is formed.



In the above fig, in part **(a)** each cluster is viewed to consists of a single element. Part **(b)** illustrates four cluster. Here there are two sets of two-element clusters. These clusters are formed at this level because these two elements are close to each other than any of other elements. Part **(c)** shows the new cluster formed by adding a close element to one of the two-element clusters. In part **(d),** the two-element and three element clusters are merged to give a five-element clusters. This is done because these two clusters are closer to each other than to the remote element cluster. At the last stage, part **(e)** all the six elements are merged. The corresponding dendrogram is shown below



Dendrogram of above clusters a,b,c,d,e

Approaches

i. **Agglomerative Approach – Bottom Up**

This approach is also known as the bottom-up approach. In this, we start with each object forming a separate group. It keeps on merging the objects or groups that are close to one another. It keeps on doing so until all of the groups are merged into one or until the termination condition holds.

ii. **Divisive Approach – Top Bottom**

This approach is also known as the top-down approach. In this, we start with all of the objects in the same cluster. In the continuous iteration, a cluster is split up into smaller clusters. It is down until each object in one cluster or the termination condition holds. This method is rigid, i.e., once a merging or splitting is done, it can never be undone.

Approaches to Improve Quality of Hierarchical Clustering

Here are the two approaches that are used to improve the quality of hierarchical clustering –

- Perform careful analysis of object linkages at each hierarchical partitioning.
- Integrate hierarchical cluster by first using a hierarchical cluster algorithm to group objects into micro-clusters, and then performing macro-clustering on the micro-clusters.

Two Important things that you should know about hierarchical clustering are:

- This algorithm has been implemented above using bottom up approach. It is also possible to follow top-down approach starting with all data points assigned in the same cluster and recursively performing splits till each data point is assigned a separate cluster.

- The decision of merging two clusters is taken on the basis of closeness of these clusters. There are multiple metrics for deciding the closeness of two clusters:
  - Euclidean distance: $||a-b||_2 = \sqrt{(\Sigma(a_i-b_i))}$
  - Squared Euclidean distance: $||a-b||_2^2 = \Sigma((a_i-b_i)^2)$
  - Manhattan distance: $||a-b||_1 = \Sigma|a_i-b_i|$
  - Maximum distance: $||a-b||_{INFINITY} = max_i|a_i-b_i|$
  - Mahalanobis distance: $\sqrt{((a-b)^T S^{-1} (-b))}$ {where, s : covariance matrix}

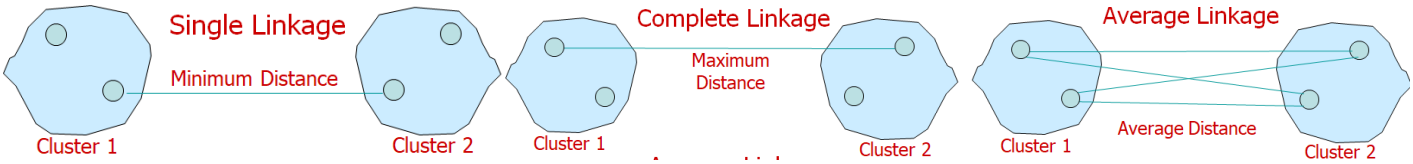
Steps in Hierarchical Clustering

Given a set of N items to be clustered, and an NxN distance (or similarity) matrix, the basic process of Johnson's (1967) hierarchical clustering is this:

i. Start by assigning each item to its own cluster, so that if you have N items, you now have N clusters, each containing just one item. Let the distances (similarities) between the clusters equal the distances (similarities) between the items they contain.
ii. Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one less cluster.
iii. Compute distances (similarities) between the new cluster and each of the old clusters.
iv. Repeat steps 2 and 3 until all items are clustered into a single cluster of size N.

Step 3 can be done in different ways, which is what distinguishes single-link from complete link and average-link clustering.

- In **single-link clustering** (also called the connectedness or minimum method)- Distance of the *"closest" points*, we consider the distance between one cluster and another cluster to be equal to the shortest distance from any member of one cluster to any member of the other cluster. If the data consist of similarities, we consider the similarity between one cluster and another cluster to be equal to the greatest similarity from any member of one cluster to any member of the other cluster.
- In **complete-link clustering** (also called the diameter or maximum method)- Distance of the "furthest" points, we consider the distance between one cluster and another cluster to be equal to the longest distance from any member of one cluster to any member of the other cluster.
- In **average-link clustering-** Average distance between pairs of elements, we consider the distance between one cluster and another cluster to be equal to the average distance from any member of one cluster to any member of the other cluster.

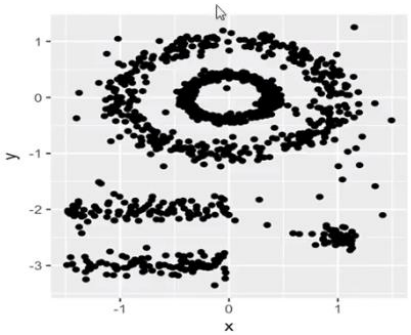


Difference between K Means and Hierarchical clustering

- Hierarchical clustering can't handle big data well but K Means clustering can. This is because the time complexity of K Means is linear i.e. O(n) while that of hierarchical clustering is quadratic i.e. $O(n^2)$.
- In K Means clustering, since we start with random choice of clusters, the results produced by running the algorithm multiple times might differ. While results are reproducible in Hierarchical clustering.
- K Means is found to work well when the shape of the clusters is hyper spherical (like circle in 2D, sphere in 3D).
- K Means clustering requires prior knowledge of K i.e. no. of clusters you want to divide your data into. But, you can stop at whatever number of clusters you find appropriate in hierarchical clustering by interpreting the dendrogram

4. DBSCAN Clustering: *density based algorithm*

- Partitioning methods and hierarchical clustering are suitable for finding spherical-shaped clusters i.e. they are well only for compact and well separated clusters.
- They are more affected by the presence of noise and outliers in the data.
- Unfortunately, real life data can contain
  - clusters of arbitrary shape e.g. oval, linear, "S" shape clusters
  - many outliers and noise



The plot above contains 5 clusters and outliers, including:
2 ovales clusters
2 linear clusters
1 compact cluster

DBSCAN Goal: to identify dense regions, which can be measured by the number of objects close to a given point.

- Two important parameters are required for DBSCAN: Minimum number of points (**MinPts**) and Epsilon "**Eps**"

- **Eps** is defined as the radius of neighborhood around a pint x. it is called ε-neighborhood of x.
- The parameter **MiPts** is the minimum number within "eps" radius.

Density = number of points within a specified radius r (Eps)

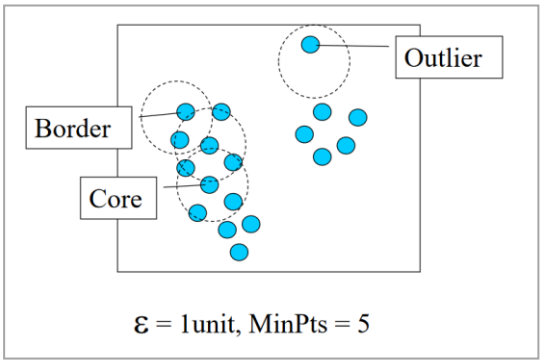
Major features:

- Can handle clusters of different shapes and sizes
- Handle noise
- One scan
- Need density parameters
- Cannot handle variable densities
- sensitive to parameters hard to determine the correct set of parameters

Core, Border & Outlier

Given ε and MinPts categorize the objects into three exclusive groups.

1. A point is a **core point** if it has more than or equal a specified minimum number of points (MinPts) within epsilon "Eps"
   - These are points that are at the interior of a cluster.
2. **A border point** has fewer than MinPts within Eps, but is in the neighborhood of a core point.
3. A **noise point** is any point that is not a core point nor a border point.



Basic concept:

For any cluster we have:

- A central point (p) i.e. core point
- A distance from the core point(Eps)
- Minimum number of points within the specified distance (MinPts)

Basic idea

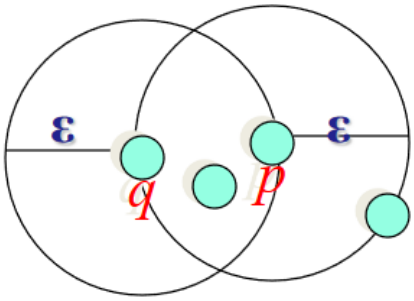
- Clusters are dense regions in the data space, separated by regions of lower object density
- A cluster is defined as a maximal set of density-connected points
- Discovers clusters of random shape
- DBSCAN requires two parameters: epsilon (Eps) and minimum points (MinPts).It starts with an arbitrary starting point that has not been visited .It then finds all the neighbour points within distance Eps of the starting point.
- If the number of neighbours is greater than or equal to MinPts, a cluster is formed. The starting point and its neighbours are added to this cluster and the starting point is marked as visited. The algorithm then repeats the evaluation process for all the neighbours recursively.
- If the number of neighbours is less than MinPts, the point is marked as **noise**.
- If a cluster is fully expanded (all points within reach are visited) then the algorithm proceeds to iterate through the remaining unvisited points in the dataset.

Density Definition

- ε-Neighborhood –Objects within a radius of ε from an object.

$$N_\varepsilon(p) : \{q \mid d(p,q) \le \varepsilon\}$$

- "High density" -ε-Neighborhood of an object contains at least MinPts of objects



ε-Neighborhood of *p*

ε-Neighborhood of *q*

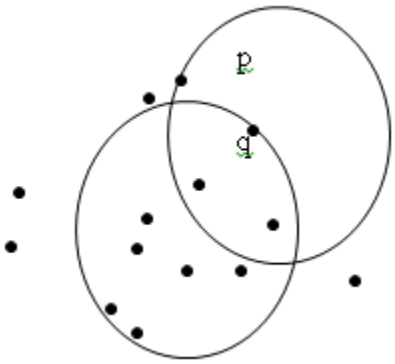
Density of p is "high" (MinPts = 4)

Density of q is "low" (MinPts = 4)

DBSCAN Algorithm

- For each point xi, compute the distance between xi and the other points
- Find all neighbor points within distance eps of the starting point xi
- Each point, with a neighbor count greater than or equal to MinPts, is marked as core point or visited.
- For each core point, if it's not already assigned to a cluster, create new cluster

- Find recursively all its density connected points and assign them to the same cluster as the core point.
- Iterate through the remaining unvisited points in the data set. Those points that don not belong to any cluster are noise.



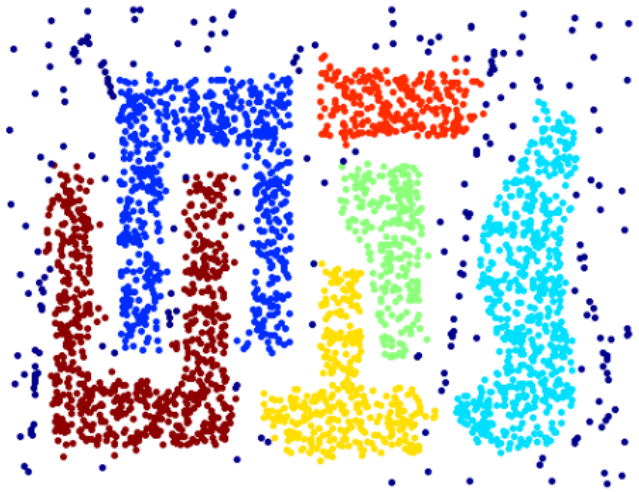
MinPts= 5
Eps= 1cm

- MinPts: Minimum number of points in any cluster
- ϵ: For each point in cluster there must be another point in it less than this distance away.
- ε-neighborhood: Points within ε distance of appoint
- N ϵ (p) :{q belongs to D |dist(p,q) <=ϵ)
- Core point: ϵ - neighborhood dense enough (MinPts)
- Conditions: p belongs to N ϵ(q)
  |Nϵ(q)| > = MinPts

When DBSCAN Works Well

- Can handle clusters of different shapes and sizes
- Resistant to Noise

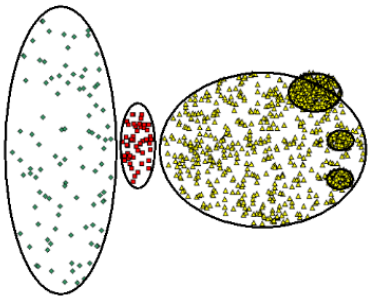
Original Points



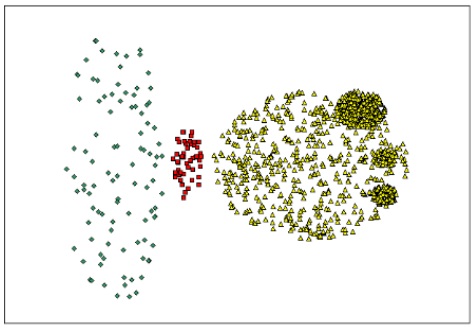
Clusters

When DBSCAN Does NOT Work Well

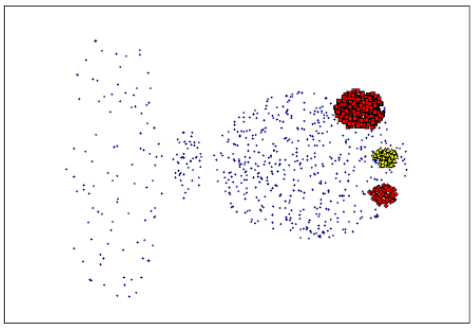
- Cannot handle variable densities
- sensitive to parameters hard to determine the correct set of parameters



Original Points



(MinPts=4, Eps=9.92).



(MinPts=4, Eps=9.75)

5. Issues : Evaluation, Scalability, Comparison

- In the first iteration, all HAC methods need to compute similarity of all pairs of *n* individual instances which is O(mn^2).
- In each of the subsequent *n*–2 merging iterations, compute the distance between the most recently created cluster and all other existing clusters.
- Maintaining of heap of distances allows this to be O($mn^2 \log n$)

Major issue – labeling

- After clustering algorithm finds clusters - how can they be useful to the end user?
- Need pithy label for each cluster
  - In search results, say "Animal" or "Car" in the ***jaguar*** example.
  - In topic trees, need navigational cues.
    - Often done by hand, a posteriori.

How would you do this?

How to Label Clusters?

- Show titles of typical documents
  - Titles are easy to scan
  - Authors create them for quick scanning!
  - But you can only show a few titles which may not fully represent cluster
- Show words/phrases prominent in cluster
  - More likely to fully represent cluster
  - Use distinguishing words/phrases
    - Differential labeling
  - But harder to scan

Labeling

- Common heuristics - list 5-10 most frequent terms in the centroid vector.
  - Drop stop-words; stem.
- Differential labeling by frequent terms
  - Within a collection "Computers", clusters all have the word ***computer*** as frequent term.
  - Discriminant analysis of centroids.
  - Perhaps better: distinctive noun phrase