# UE 803 - Data Science for NLP

## Lecture 10: Clustering

Claire Gardent - CNRS / LORIA



# Outline

- Definition: what is clustering ?
- Motivations: why cluster text ?
- Clustering Algorithms
- K-Means
- Some Key Issues
  - Text Representation
  - Number of Clusters
  - Similarity Metrics
- Evaluation: what are good clusters ?

# What is Clustering ?

- Clustering is the process of grouping a set of objects into subsets called ***clusters***
- Objects can be images, numerical data, text (documents, terms, paragraphs, websites ... )
- Clustering groups together objects that are ***similar***
- It seeks to ***maximize intra-group similarity*** while ***minimizing inter-group similarity***
- It permits discovering structure in the input data
- It is ***Unsupervised*** i.e., it learns from raw data (no labels or classes are given)

# What is Text Clustering used for ?

Clustering is used for text mining and exploratory text analysis

- To structure the data (useful for browsing)
- To discover the major topics covered in the input documents
- To remove redundancy
- As additional features for classification (feature = cluster identifier)

# Example Applications

Customer Segmentation (e.g., Amazon)

- Grouping similar customers together
  Pet store
  Customers $\Rightarrow$ cat lovers / dog lovers / fish lovers

Recommendation Systems (e.g., Deezer)

- Grouping similar users together
  Users $\Rightarrow$ Jazz fans / Classics lovers / Rap people etc.

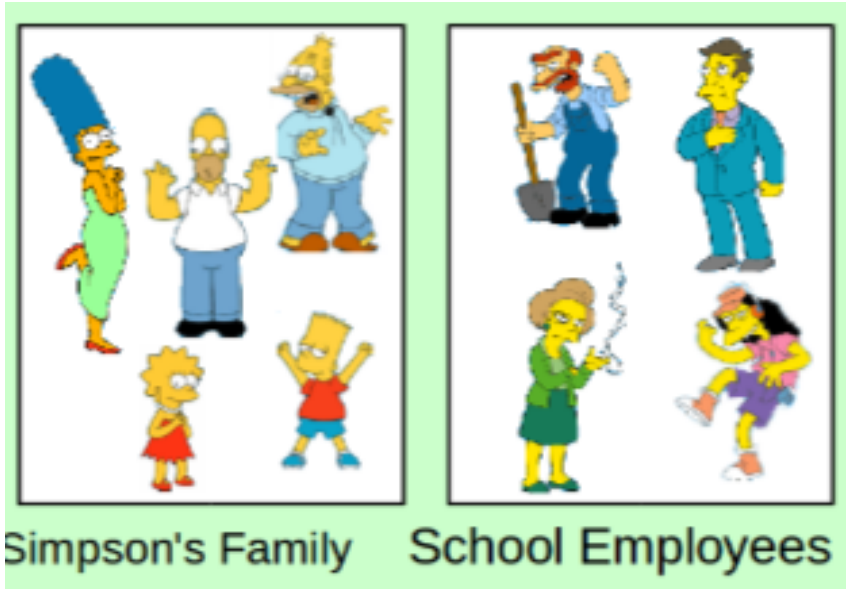
Search Engines (e.g., Google)

- Grouping similar documents together
  Retrieved documents $\Rightarrow$ Sport / Art / News / etc.

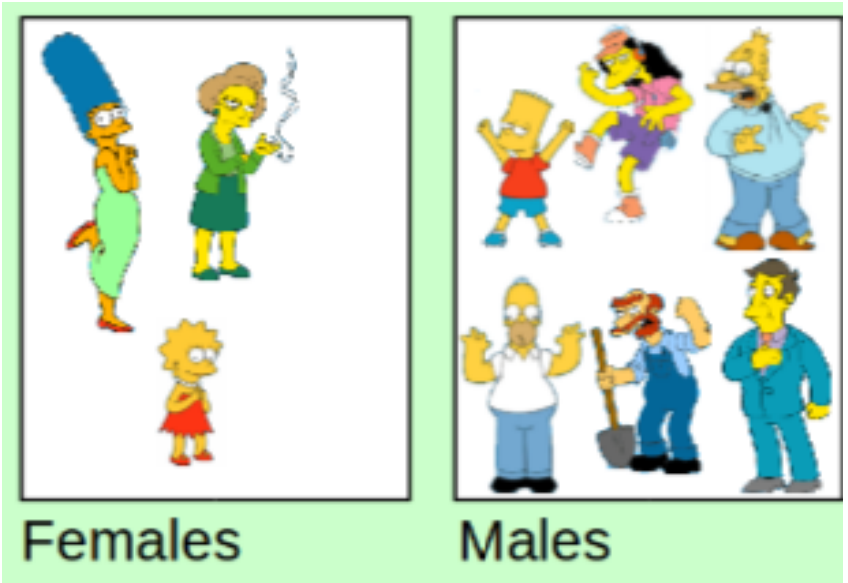
# Clustering Bias

- There can be several possible ways to cluster data
- The criteria (e.g., size vs shape) chosen to cluster data is called the ***bias***

# Clustering Bias 1: family vs not family

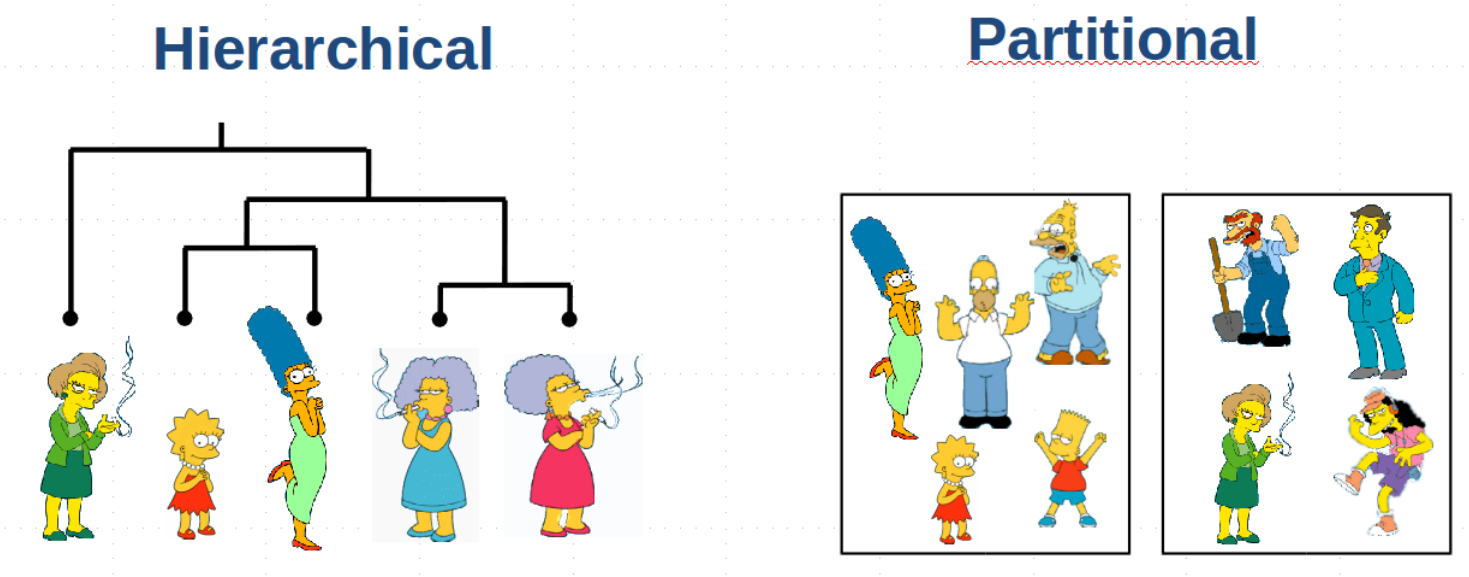


# Clustering Bias 1: female vs male



# Clustering Methods

# Flat vs Hierarchical CLustering




- Hierarchical Structure
- Larger clusters include smaller clusters

- Flat Structure
- No embedding of clusters within another cluster

# Clustering Methods

## Generative Probabilistic Models

- Not discussed in this course

## Similarity-Based Approaches

- Flat or Partitional (***K-Means*** )
  - Starts with a random partitioning into $n$ clusters
  - Refines the partitioning iteratively
- Hierarchical
  - Progressively constructs a hierarchy of clusters
  - Bottom-up (***agglomerative***)
  - Top-down (***divisive***)

# The K-Means Algorithm

# Reminder

- Machine Learning works with vectors
- A house can be represented by a vector whose dimensions indicate the size, the number of rooms etc. (cf. Lecture 09 on Linear Regression)
- A text can be represented by a vector whose dimensions indicates the words/tokens occurring in that text and their frequency

# Text as Vectors

| | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|---|---|---|---|---|
| **battle** | 1 | 0 | 7 | 13 |
| **good** | 114 | 80 | 62 | 89 |
| **fool** | 36 | 58 | 1 | 4 |
| **wit** | 20 | 15 | 2 | 3 |

**Figure 6.2** The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

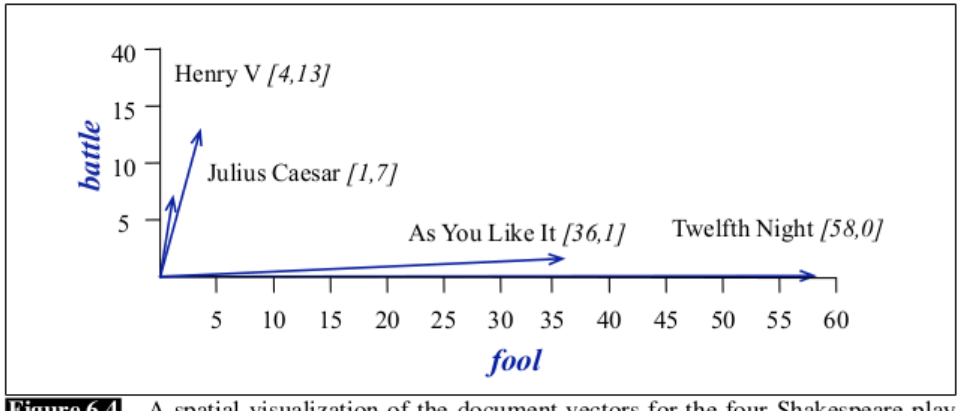



**Figure 6.4** A spatial visualization of the document vectors for the four Shakespeare play documents, showing just two of the dimensions, corresponding to the words *battle* and *fool*. The comedies have high values for the *fool* dimension and low values for the *battle* dimension.

- Julius Caesar and Henry V vectors are similar (similar values for battle and fool).
- "As you like it" and "Twelfth Night" documents are similar.

***Texts with similar content (shared tokens) have vectors which are close in space.***

# K-Means

An iterative flat/partitional clustering algorithm for *n* clusters

- Initialize
  Pick *n* random vectors as cluster centers
- Iterate
  1. Assign data points to ***closest*** cluster center
  2. ***Update the cluster center*** to the average of its newly assigned points
- Stop when no points assignments change

The ***center of a cluster*** is the average of all elements that belong to that cluster.

Cluster elements are vectors.

The center/centroid of $n$ vectors is the coordinate-wise mean of these vectors.

# Cluster Center

Three clusters

C1: {(2,2), (4,4), (6,6)}

C2: {(0,4), (4,0)}

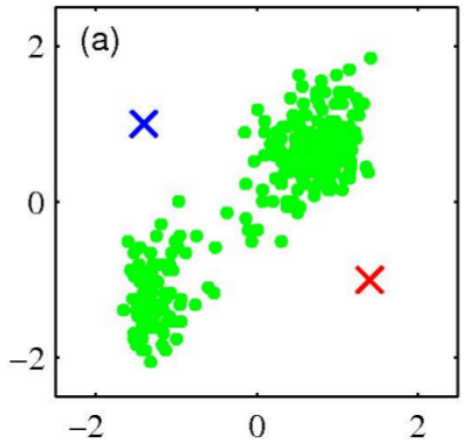
C3: {(5,5), (9,9)}

centroid C1
((2+4+6)/3, (2+4+6)/3) = (4, 4)

centroid C2
((0+4)/2, (4+0)/2) = (2, 2)

centroid cluster C3
((5+9)/2, (5+9)/2) = (7, 7)

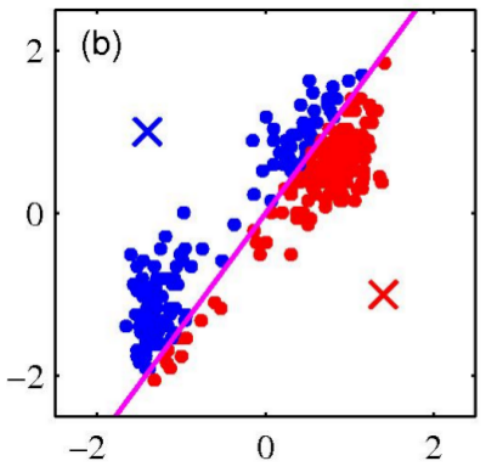
# K-Means Example




- Pick *K* random points as cluster centers (means)

Shown here for *K*=2

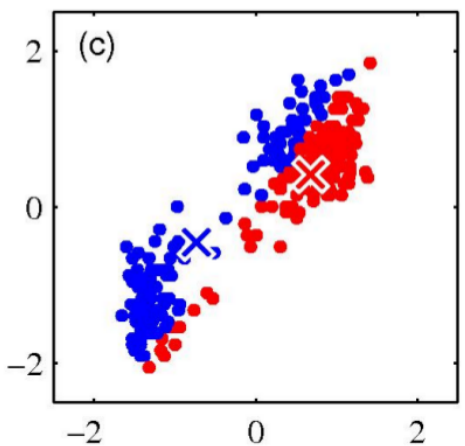
17




Iterative Step 1

- Assign data points to closest cluster center

18




Iterative Step 2

- Change the cluster center to the average of the assigned points

# Visualising the K-Means algorithm

# Expectation Maximisation

K-Means is a particularly simple application of the ***expectation–maximization*** (EM) algorithm

- Guess some cluster centers
- Repeat until convergence
  - *E-Step*: assign points to the nearest cluster center
  - *M-Step*: set the cluster centers to the mean

The *E-step* updates our ***expectation of which cluster each point belongs to***.

The *M-step* ***maximizes some fitness function*** that defines the location of the cluster centers —
In this case, maximization is accomplished by taking a simple mean of the data in each cluster.

Each repetition of the E-step and M-step results in a better estimate of the cluster characteristics.

# Clustering Text

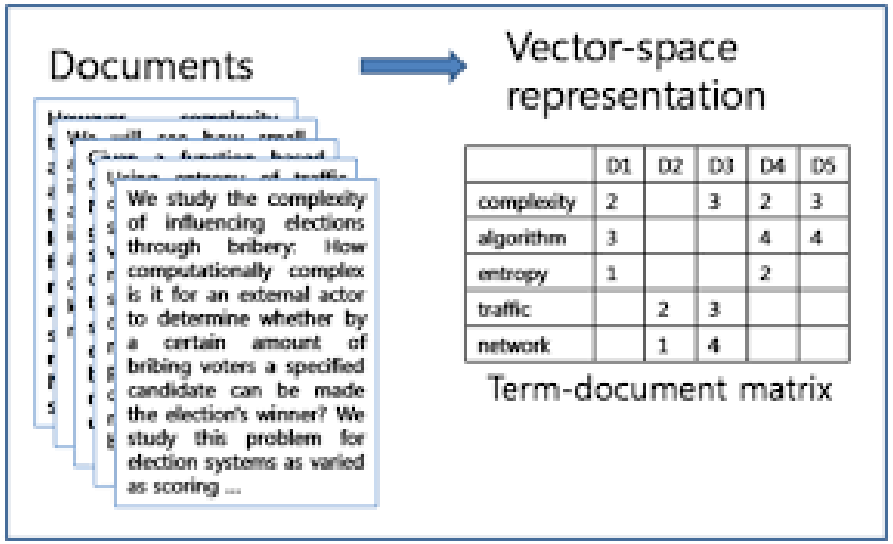
# Clustering Text

## Some key issues

- How to represent text?
- Which metrics for grouping objects ? (What does "closest" mean?)
- How many clusters ?

# Representing Text

- Texts are represented by vectors
  - vector of tokens, of token frequency, of tf×idf values etc.
  - normalised or not



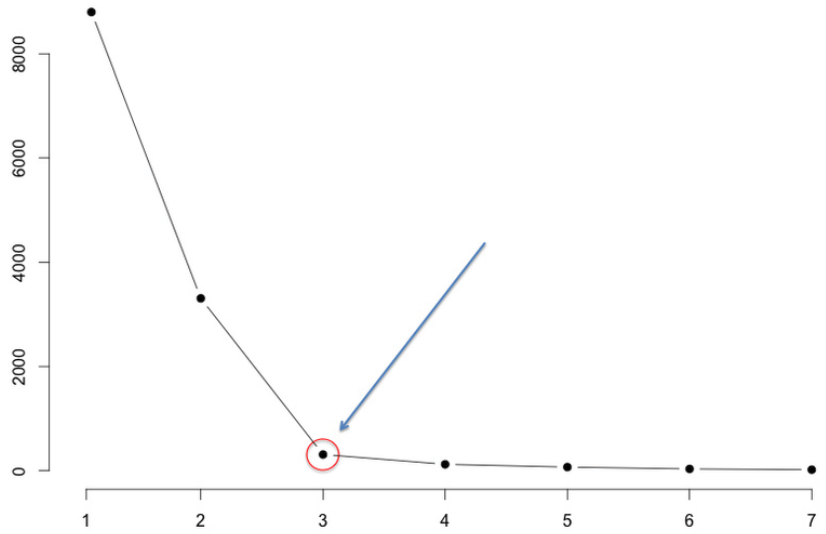

| | D1 | D2 | D3 | D4 | D5 |
|---|---|---|---|---|---|
| complexity | 2 | | 3 | 2 | 3 |
| algorithm | 3 | | | 4 | 4 |
| entropy | 1 | | | 2 | |
| traffic | | 2 | 3 | | |
| network | | 1 | 4 | | |

# Clustering Criterion

- Distance between two vectors ⇔ Similarity between two texts
- Cosine, Inner product, Manhattan distance, Euclidean distance etc.

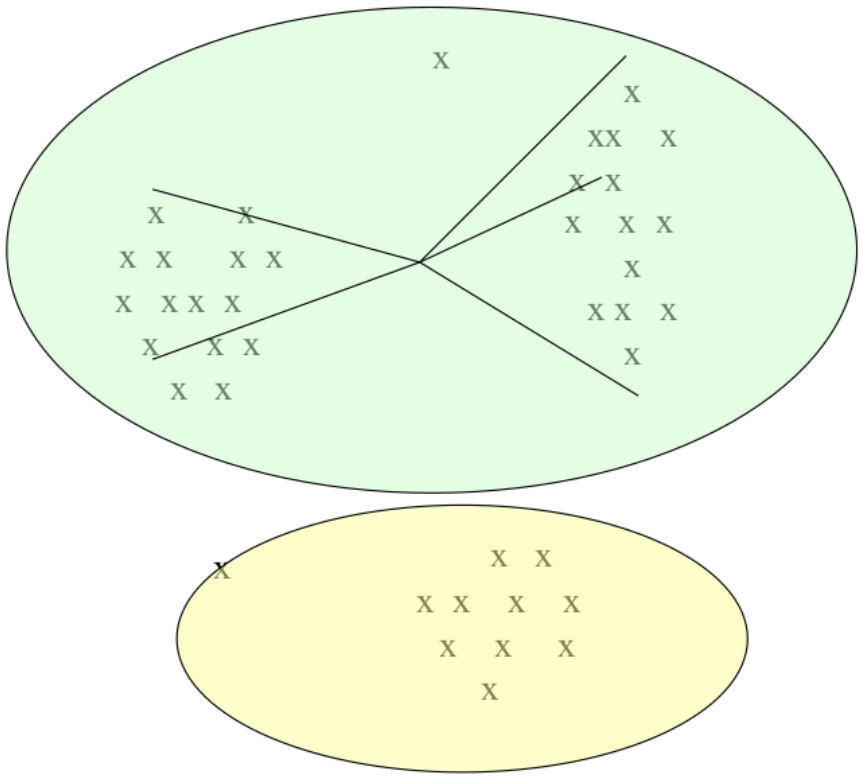
# Number of Clusters

- Fixed a priori
- Or optimized using the Elbow Method
  - Run the algorithm for different values of K
  - plot the K values against average distance to centroid
  - Select the value of K for the elbow point as shown in the figure.



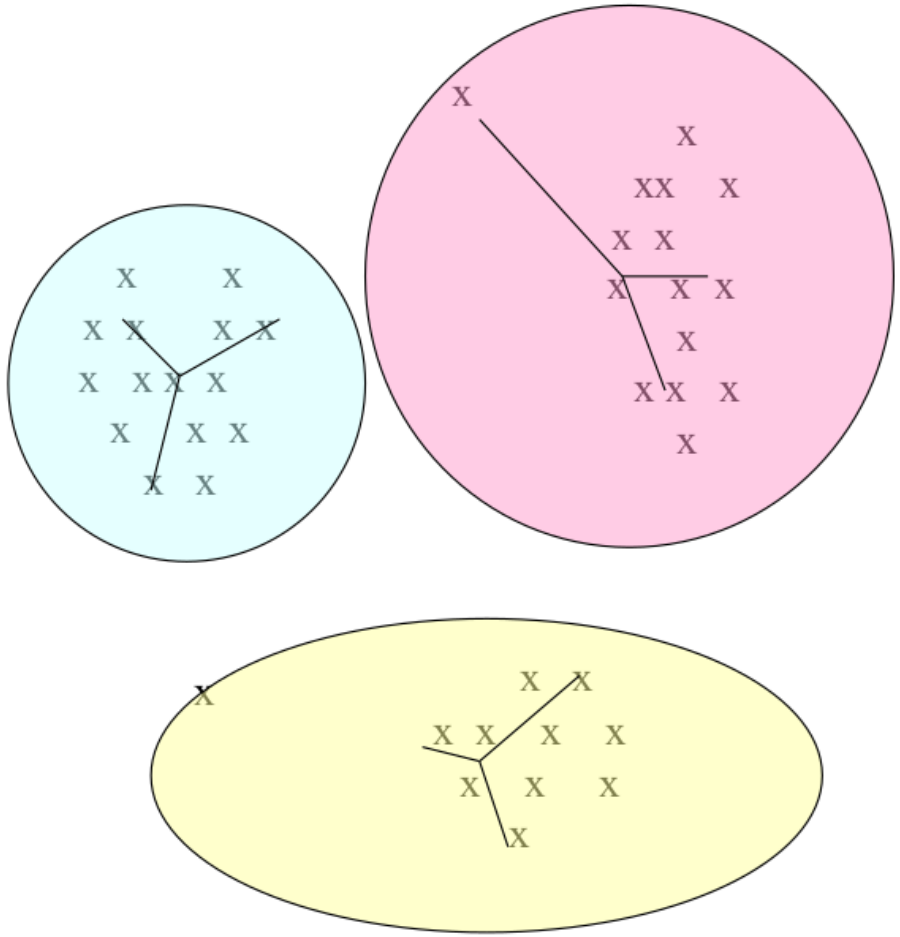

The elbow method looks at the percentage of variance explained as a function of the number of clusters: One should choose a number of clusters so that adding another cluster doesn’t give much better modeling of the data.

# K too small



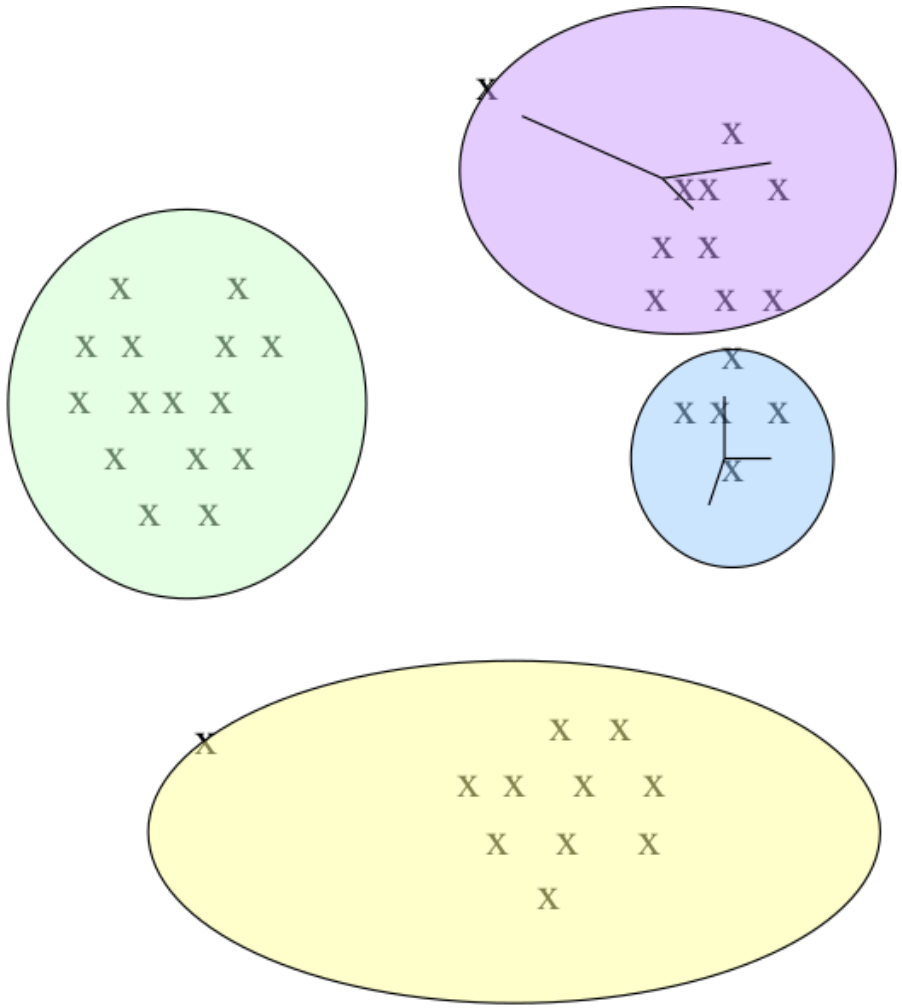
Many points far away from centroid

# K just right



Most points close to center

# K too large



More clusters, not much improvment

# Hierarchical Clustering

# Hierarchical Clustering

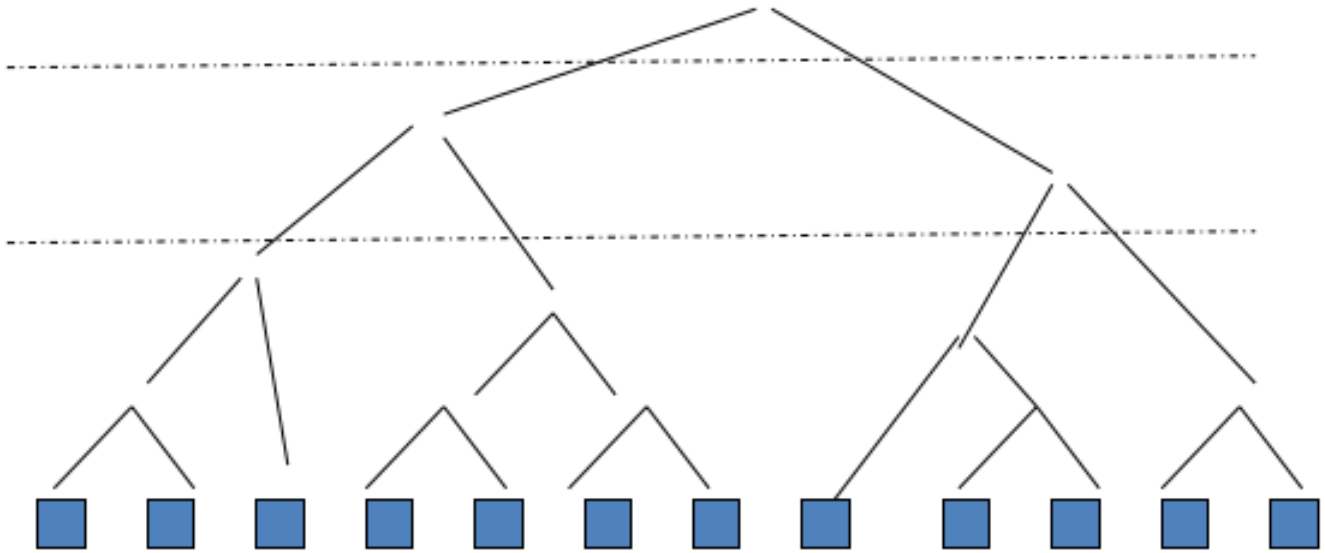
- Divisive (top down) clustering
- Agglomerative (bottom up) clustering

# Divisive (top down) clustering

Starts with all data points in one cluster, the root, then

- Splits the root into a set of children clusters e.g., using k-means.
- Each child cluster is recursively divided further
- stops when only singleton clusters of individual data points remain, i.e., each cluster with only a single point

# Divisive Top-Down Clustering



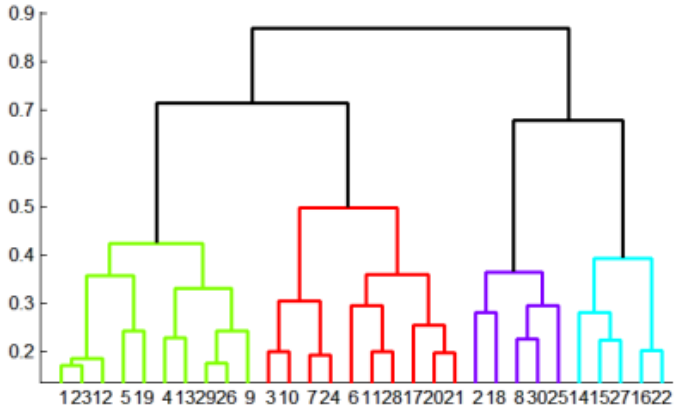
# Agglomerative (bottom up) clustering

The clusters are built from the bottom level by

- merging the most similar (or nearest) pair of clusters
- stopping when all the data points are merged into a single cluster (i.e., the root cluster).

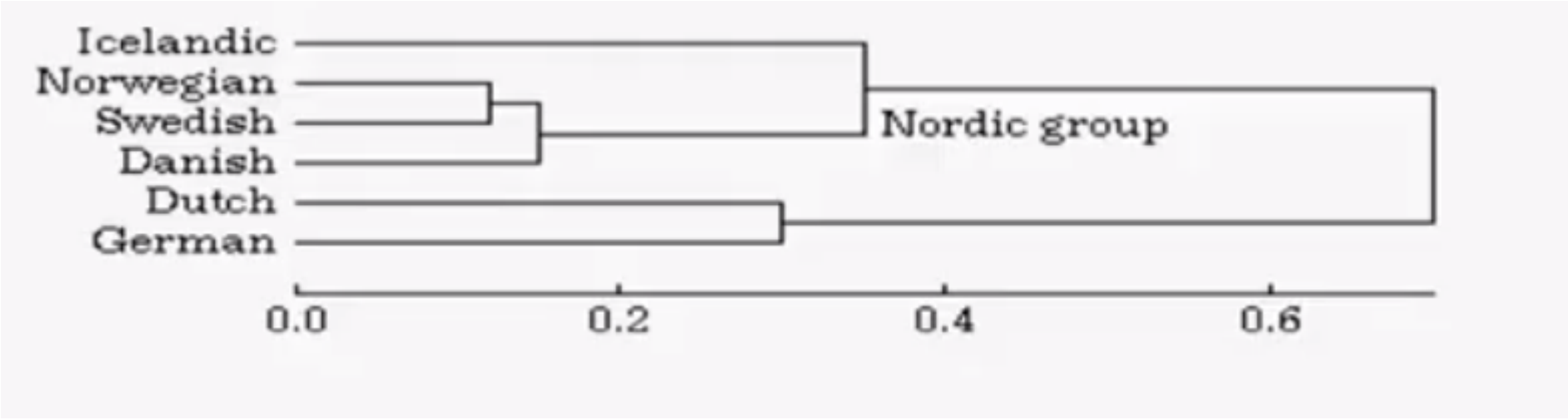
Creates a ***dendogram***

# Dendogram




- The height in the dendrogram at which two clusters are merged represents the distance between two clusters in the data space
- When drawing a horizontal line across the dendogram, the number of vertical lines crossed yields the number of clusters

# Clustering Languages



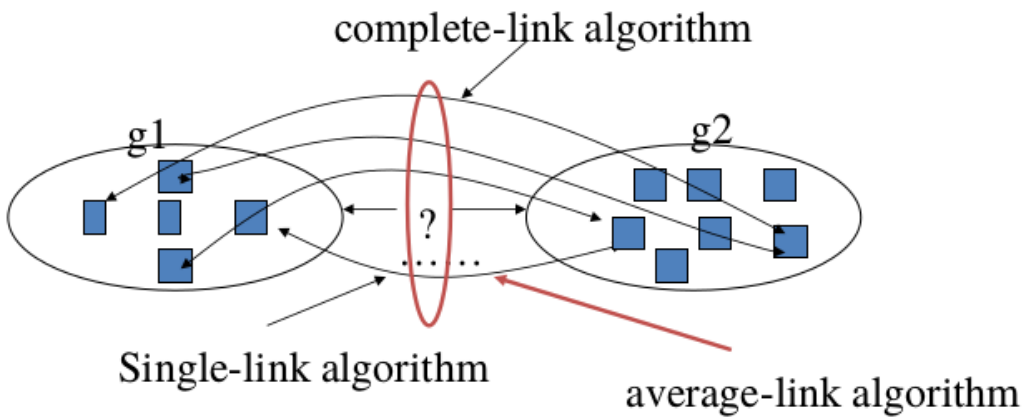

src: http://www.let.rug.nl/kleiweg/clustering/clustering.html

# Several ways of Computing Group Similarity

- Similarity of the closest pair (*Single-link algorithm*)
  Merge two clusters if ***one item pair*** has highest similarity

- Similarity of the farthest pair (*Complete-link algorithm*)
  Merge two clusters if ***all item pairs*** in clusters are similar enough

- ***Average*** similarity of all pairs (*Average-link algorithm*)



# Evaluating Clustering Results

# Evaluating Clusters

## Extrinsic (supervised)

- Compare a clustering against the ground truth
- ***Homogeneity, Completeness and V-measure, Rand index***, Adjusted Rand index, Fowlkes-Mallows scores, Mutual information based scores ...

## Intrinsic (unsupervised)

- Evaluate the goodness of a clustering by considering how well the clusters are separated (***maximise intra-group similarity*** ), and how compact they are ( ***minimizing inter-group similarity*** )
- ***Silhouette coefficient***

# Extrinsic Evaluation (Supervised)

**Homogeneity**

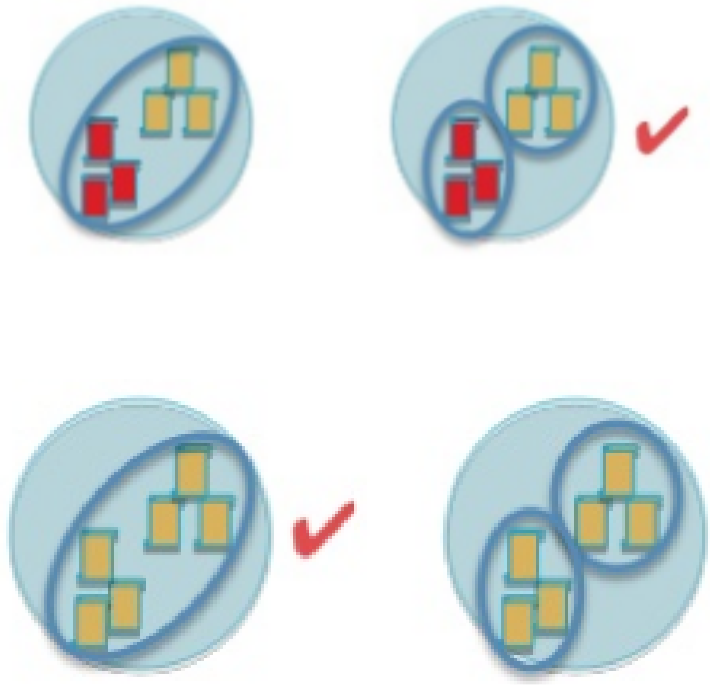
$H = 1.0$
Each cluster contains only members of a single class

**Completeness***

$C = 1.0$
All members of a class are assigned to the same cluster

**V-measure**

Harmonic mean of H and C



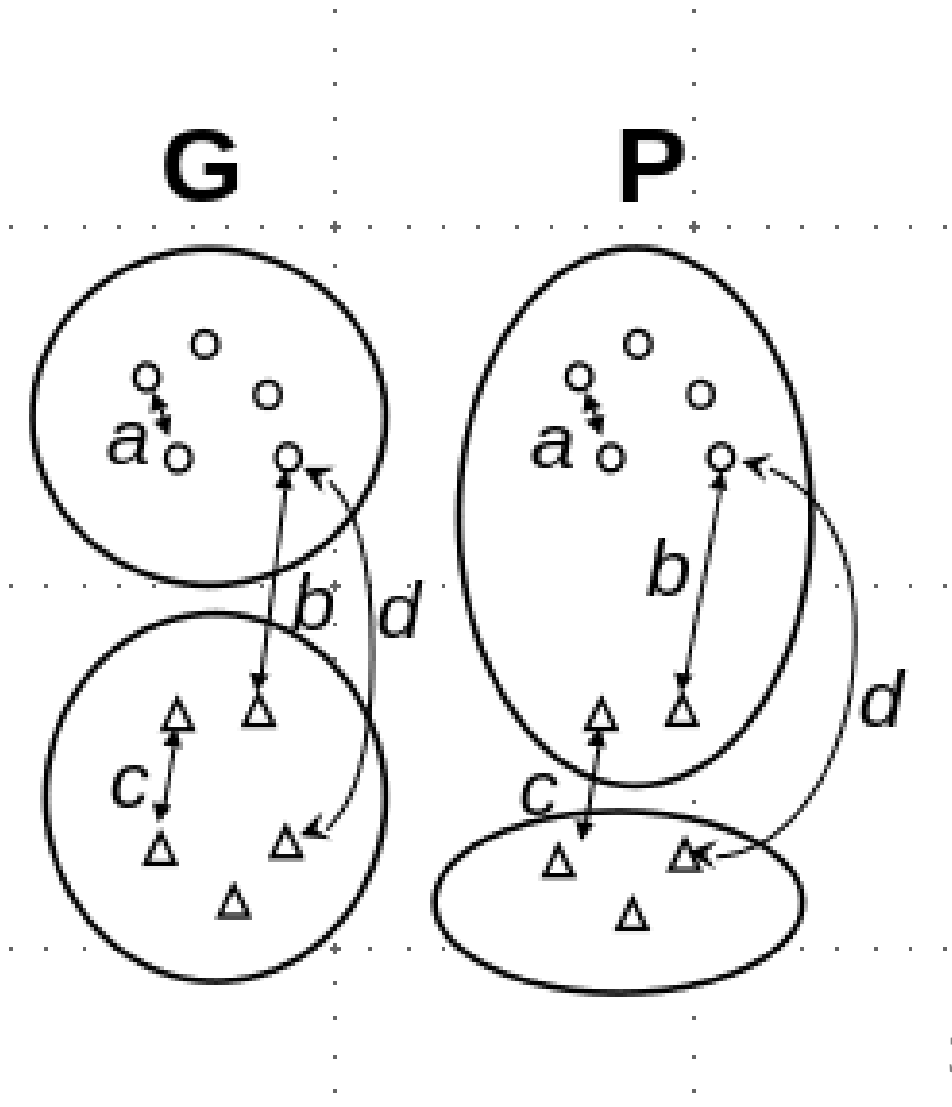
# Rand Index

Computes how similar the clusters returned by the clustering algorithm are to the ground truth clusters.

$$RI = \frac{TP + TN}{TP + TN + FP + FN}$$

The instances being counted here are the number of *correct pairwise assignments*.

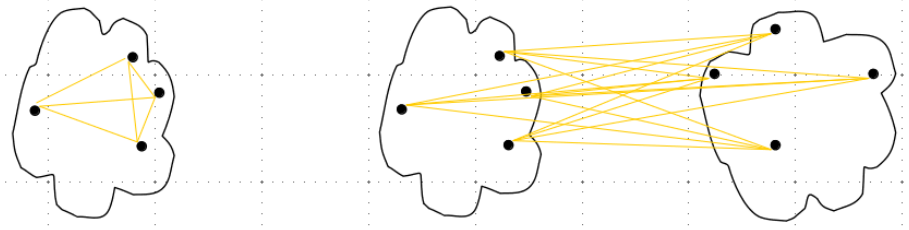
- TP (a): same class in P (prediction) and G (Ground Truth)
- TN (d): different class in both P and G
- FP (b): same class in P but different class in G
- FN (c): different class in P but same class in G



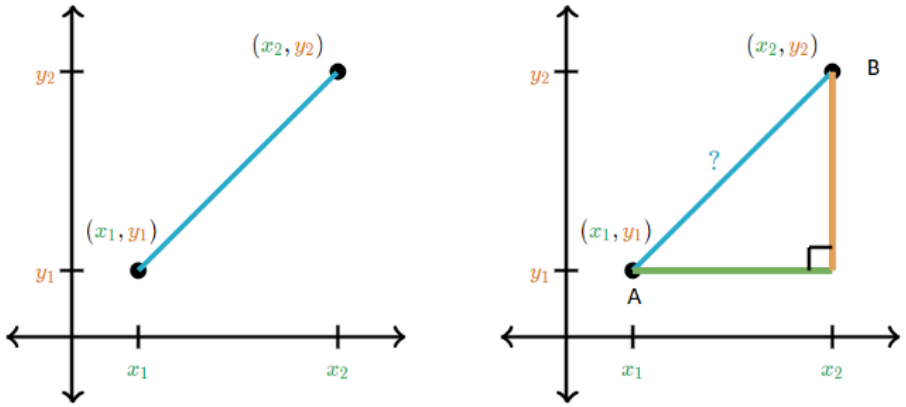
# Intrinsic Evaluation: the Silhouette Coefficient

- measures how similar an object is to its own cluster (cohesion) and how dissimilar it is compared to datapoints in other clusters (separation)
- ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.
- can be calculated with any distance metric, such as the Euclidean distance
- the mean Silhouette coefficient over all data in the entire dataset is a measure of how appropriately the data have been clustered

- Cohesion: measures how closely related are objects in a cluster
- Separation: measure how distinct or well-separated a cluster is from other clusters



# Euclidean Distance




We have two vectors/points: $A = (x1, y1)$ and $B = (x2, y2)$

Euclidean distance between A and B ?

Let's define $C = (x2, y1)$

Pythagore theorem: $AB^2 = AC^2 + CB^2$

$$AB = \sqrt{AC^2 + CB^2}$$

$$AC = (x_2 - x_1)$$

$$CB = (y_2 - y_1)$$

$$AB = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

# Silhouette Coefficient

- Cohesion $a(x)$: average distance of $x$ to all other vectors in the same cluster
- Separation $b(x)$: average distance of $x$ to the vectors in other clusters.
- *silhouette(x)*

$$s(x) = \frac{b(x) - a(x)}{max(a(x), b(x))}$$

- $s(x) \in [-1, +1]$, 1 = bad, 0 = indifferent, 1 = good
- Silhouette Coefficient (SC)

$$SC = \frac{1}{N} \sum s(x)$$

# Clustering in Python

# Loading Data into a Pandas dataframe

```
!ls ../data/bbc/
```

```
business  entertainment  politics  sport  tech
```

```
import pandas as pd
from sklearn.datasets import load_files
DATA_DIR = "../data/bbc/"
data = load_files(DATA_DIR, encoding="utf-8", decode_error="replace")
df = pd.DataFrame(list(zip(data['data'], data['target_names'])), columns=['text', 'label'])
df.head()
```

```
text    label
0    Tate & Lyle boss bags top award\n\nTate & Lyle...    business
1    Halo 2 sells five million copies\n\nMicrosoft ...    entertainment
2    MSPs hear renewed climate warning\n\nClimate c...    politics
3    Pavey focuses on indoor success\n\nJo Pavey wi...    sport
4    Tories reject rethink on axed MP\n\nSacked MP ...    tech
```

# Converting Text into vectors of TF-IDF scores

```
import nltk
from nltk import word_tokenize

# Create a TFIDF vectorizer to convert convert words to vectors
tfidf_vectorizer = TfidfVectorizer(max_features=8000,
                                   use_idf=True,
                                   stop_words='english',
                                   tokenizer=nltk.word_tokenize,
                                   ngram_range=(1, 3))

# Apply the vectorizer to the input texts (X)
# X is a list of strings/texts
tfidf_matrix = tfidf_vectorizer.fit_transform(X)
```

# Evaluating the Clustering

```
from sklearn.cluster import KMeans

# Create a KMeans clustering model
km = KMeans(n_clusters=5, init='k-means++', max_iter=300, n_init=5, verbose=0, random_state=3425)

# Apply the clustering model on the tf-idf matrix (the features)
km.fit(tfidf_matrix)

# Print out the predicted labels
predicted_labels = km.labels_
clusters = predicted_labels.tolist()
```

# Inspect the model predictions

Computing the various metrics

```
from sklearn import metrics

# labels = truth labels
# km.labels = predicted labels

# Compute and show evaluation scores
print("Homogeneity: %0.3f" % metrics.homogeneity_score(labels, km.labels_))
print("Completeness: %0.3f" % metrics.completeness_score(labels, km.labels_))
print("V-measure: %0.3f" % metrics.v_measure_score(labels, km.labels_))
print("Adjusted Rand-Index: %.3f"
      % metrics.adjusted_rand_score(labels, km.labels_))
print("Silhouette Coefficient: %0.3f"
      % metrics.silhouette_score(tfidf_matrix, km.labels_, sample_size=1000))

print()
```

# Lab Session

In this lab session, we will use clustering to group Wikipedia articles into 16 clusters. The dataset consists of 160 documents and 16 categories: 'Transport', 'Astronomical_objects', 'Monuments_and_memorials', 'Astronauts', 'Politicians', 'Sportspeople', 'Foods', 'Companies', 'Building', 'Artists', 'Comics_characters', 'Written_communication', 'Airports', 'Universities_and_colleges', 'City', 'Sports_teams' .

Since we are working on clustering (unsupervised machine learning), we will ignore the labels when training. We will use them however to evaluate the trained model (intrinsic evaluation).

The exercises cover the following points:

- Storing the data into an pandas dataframe and inspecting the data
- Learning a clustering model from the data
- Inspecting the output clusters

# Useful Links

- Scikit-learn Recipe notebook for clustering
- Python Data Science Handbook Chapter on Clustering: https://jakevdp.github.io/PythonDataScienceHandbook/05.11-k-means.html
- Recipe: Text Clustering using NLTK and scikit-learn
- Visualizing K-Means: http://shabal.in/visuals/kmeans/1.html
- Annotated Jupyter Notebook on how to cluster text corpora
- http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/
- http://cgm.cs.mcgill.ca/~godfried/student_projects/bonnef_k-means
- http://www.cs.washington.edu/research/imagedatabase/demo/kmcluster
- Video on hierarchical clustering