# Text Clustering

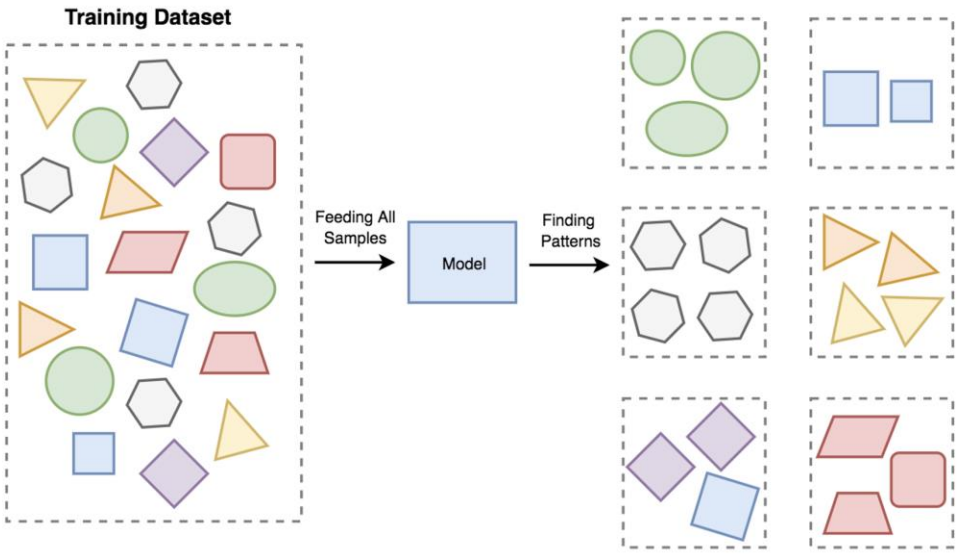
## Applied Text Mining

Ayoub Bagheri

## Lecture plan

1. What is text clustering?
2. What are the applications?
3. How to cluster text data?

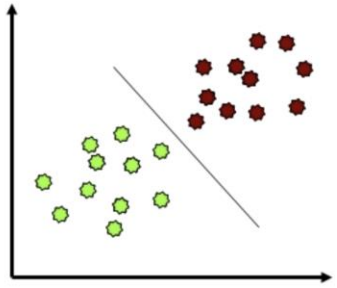
## Unsupervised learning



## Clustering versus classification

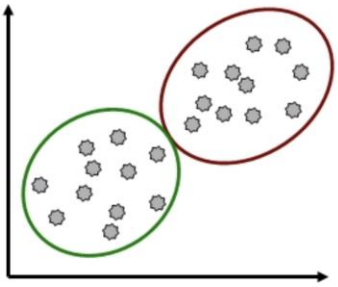
### CLASSIFICATION

- Labeled data points
- Want a "rule" that assigns labels to new points
- Supervised learning



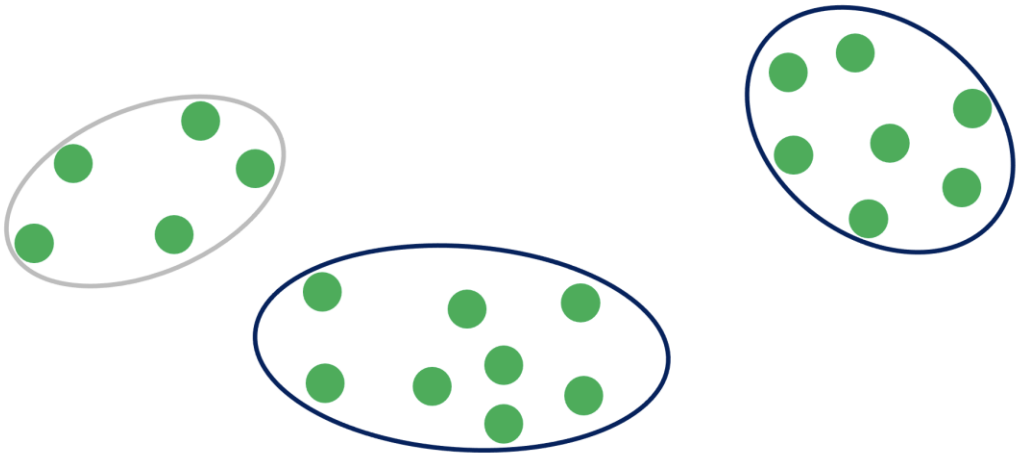
### CLUSTERING

- Data is not labeled
- Group points that are "close" to each other
- Identify structure or patterns in data
- Unsupervised learning



## Clustering

- Clustering: the process of grouping a set of objects into clusters of similar objects
- Discover "natural structure" of data
  - What is the criterion?
  - How to identify them?
  - How to evaluate the results?

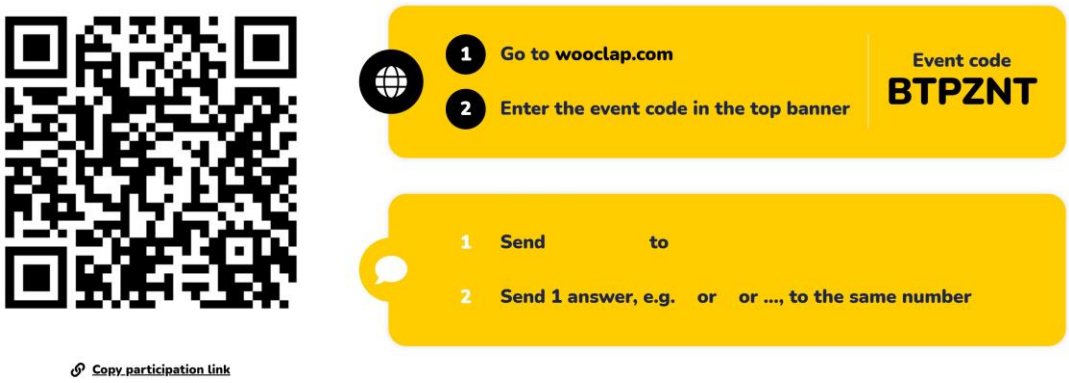


## Question

Which one is NOT a text clustering task?

- Finding similar patterns in customer reviews
- Grouping tweets and finding their unknown topics
- Cancer detection from patient notes
- Grouping scientific articles into similar clusters

## Question




## Clustering

- Basic criteria
  - high intra-cluster similarity
  - low inter-cluster similarity
- No (little) supervision signal about the underlying clustering structure
- Need similarity/distance as guidance to form clusters

# Clustering Algorithms

## Categories

- Hard versus soft clustering
- Partitional clustering
- Hierarchical clustering
- Topic modeling

## Hard versus Soft clustering

- Hard clustering: Each document belongs to exactly one cluster
  - More common and easier to do
- Soft clustering: A document can belong to more than one cluster.

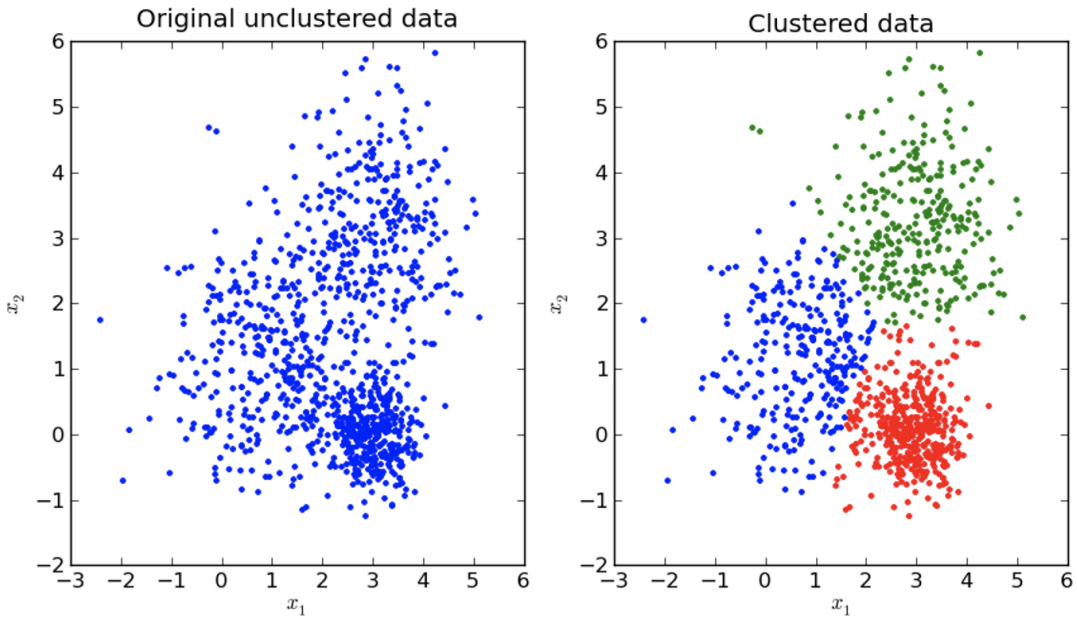
# Partitional Clustering

## Partitional clustering algorithms

- Partitional clustering method: Construct a partition of $n$ documents into a set of $K$ clusters
- Given: a set of documents and the number $K$
- Find: a partition of $K$ clusters that optimizes the chosen partitioning criterion
  - Globally optimal
    - Intractable for many objective functions
    - Ergo, exhaustively enumerate all partitions
  - Effective heuristic methods: K-means and K-medoids algorithms

## Partitional clustering algorithms

- Typical partitional clustering algorithms
  - k-means clustering
    - Partition data by its closest mean




## K-Means algorithm

- Assumes documents are real-valued vectors.
- Clusters based on centroids of points in a cluster, $c$:

$$\vec{\mu}(c) = \frac{1}{|c|}\sum_{\vec{d}\in c}\vec{x}$$

- Reassignment of instances to clusters is based on distance to the current cluster centroids.

### K-Means algorithm

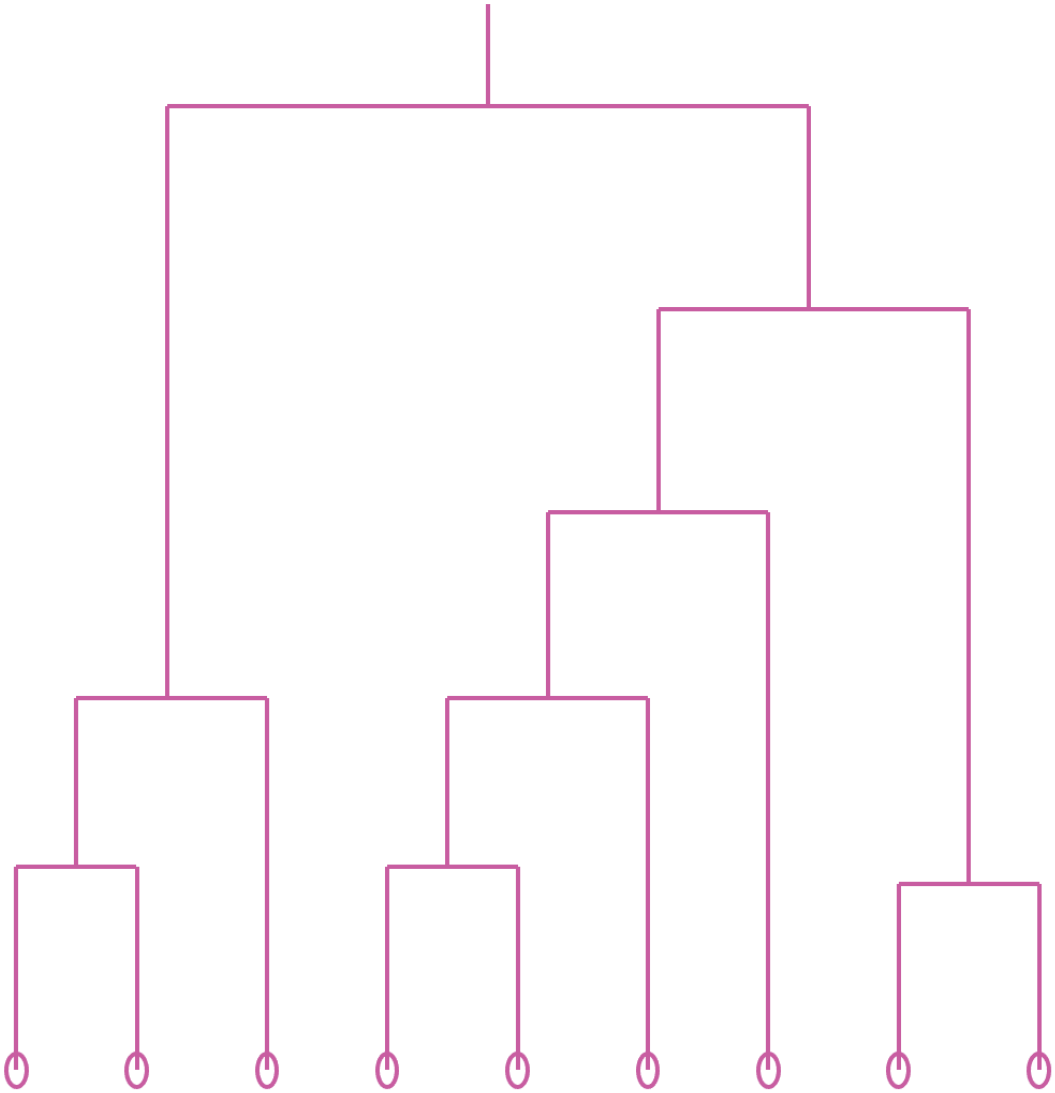
- Select $K$ random docs $\{s_1, s_2, \dots s_K\}$ as seeds.
- Until clustering converges (or other stopping criterion):
  - For each document $d_i$:
    - Assign $d_i$ to the cluster $c_j$ such that $dist(x_i, s_j)$ is minimal.
  - (Next, update the seeds to the centroid of each cluster)
  - For each cluster cj
    - $s_j = \mu(c_j)$

### K-Means example (K=2)

## Hierarchical Clustering

### Dendrogram: Hierarchical clustering

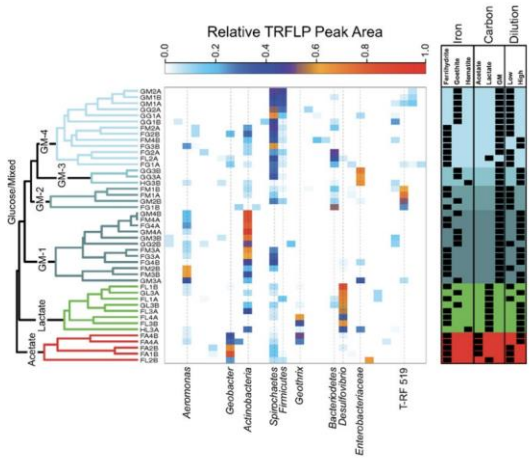
- Build a tree-based hierarchical taxonomy (dendrogram) from a set of documents.
- Clustering obtained by cutting the dendrogram at a desired level: each connected component forms a cluster.



## Clustering algorithms

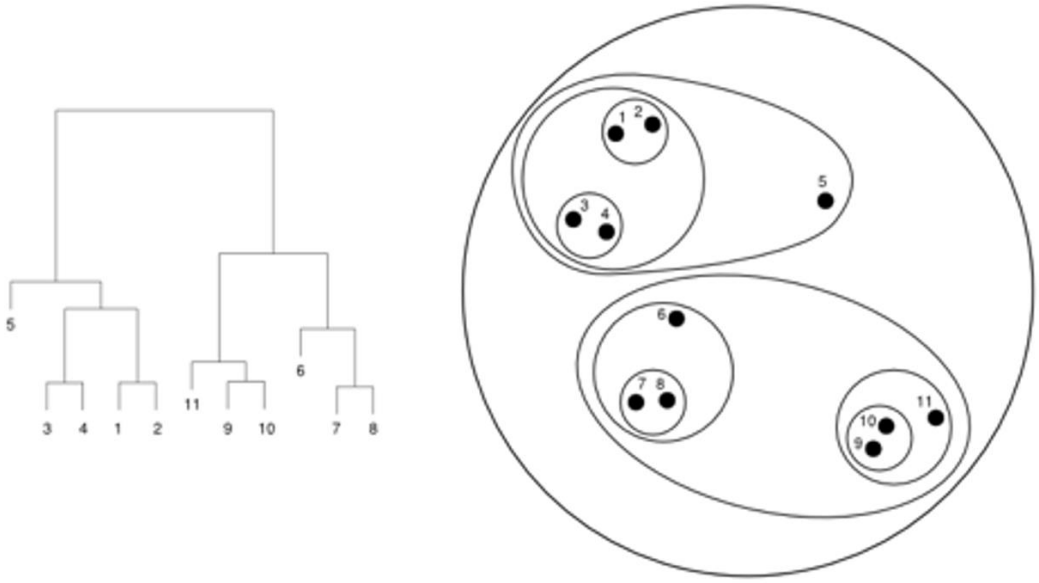
- Typical hierarchical clustering algorithms
  - Bottom-up agglomerative clustering
    - Start with individual objects as separated clusters
    - Repeatedly merge closest pair of clusters

*Most typical usage: gene sequence analysis*




## Clustering algorithms

- Typical hierarchical clustering algorithms
  - Top-down divisive clustering
    - Start with all data as one cluster
    - Repeatedly splitting the remaining clusters into two




## Hierarchical Agglomerative Clustering (HAC)

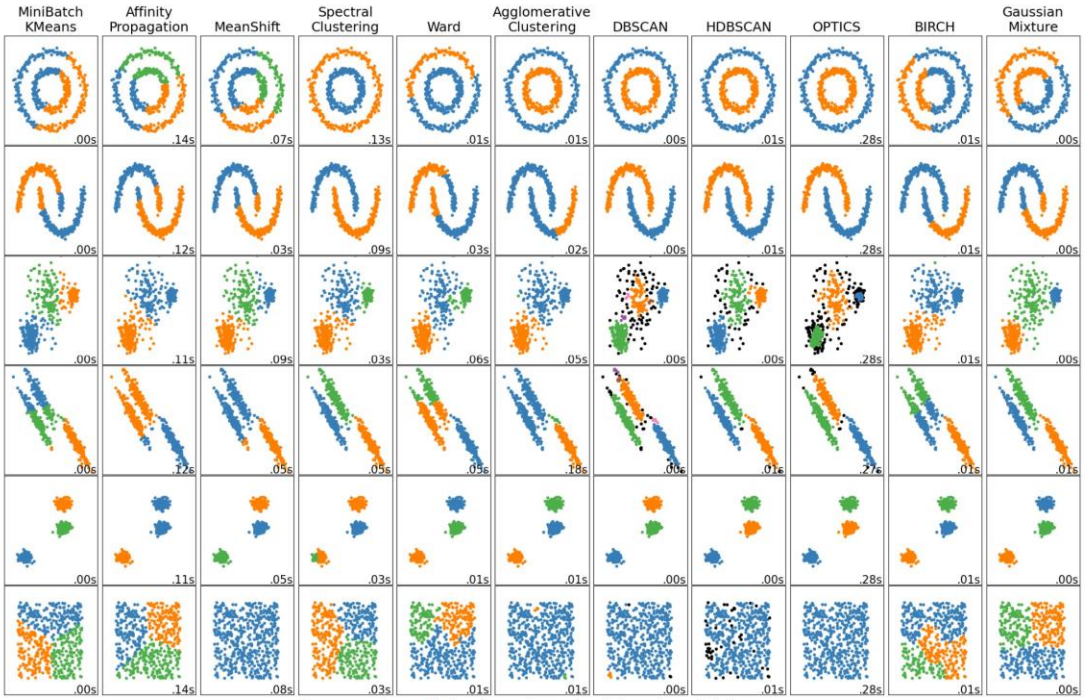
- Starts with each document in a separate cluster

  - then repeatedly joins the closest pair of clusters, until there is only one cluster.
- The history of merging forms a binary tree or hierarchy.

## Closest pair of clusters

- Many variants to defining closest pair of clusters (linkage methods):
  - Single-link
    - Similarity of the most cosine-similar
  - 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
  - Ward's linkage
    - Ward's minimum variance method, much in common with analysis of variance (ANOVA)
    - The distance between two clusters is computed as the increase in the "error sum of squares" (ESS) after fusing two clusters into a single cluster.

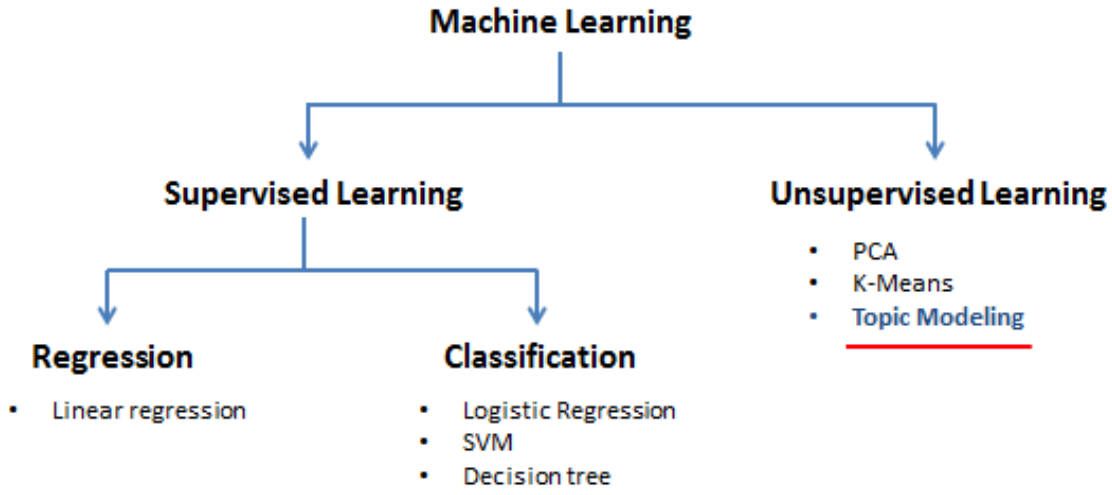
## Clustering in SKlrean



A comparison of the clustering algorithms in scikit-learn

# Topic Modeling

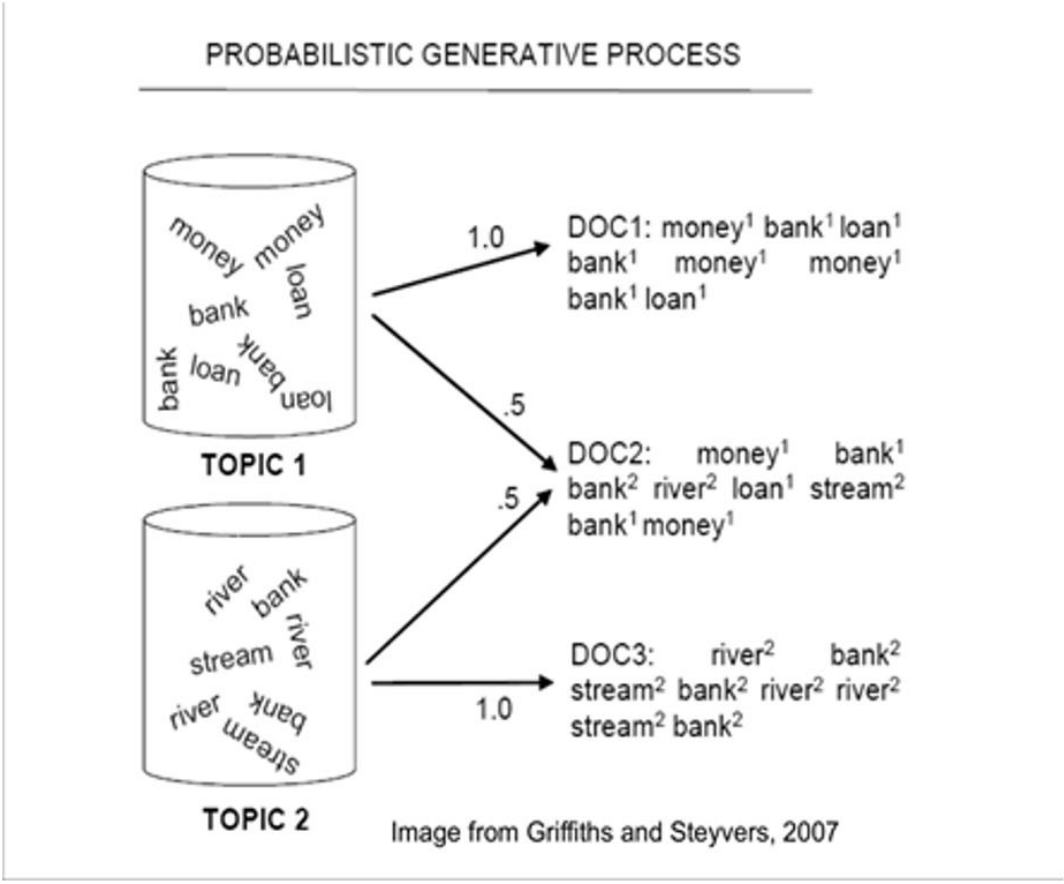
## Topic modeling



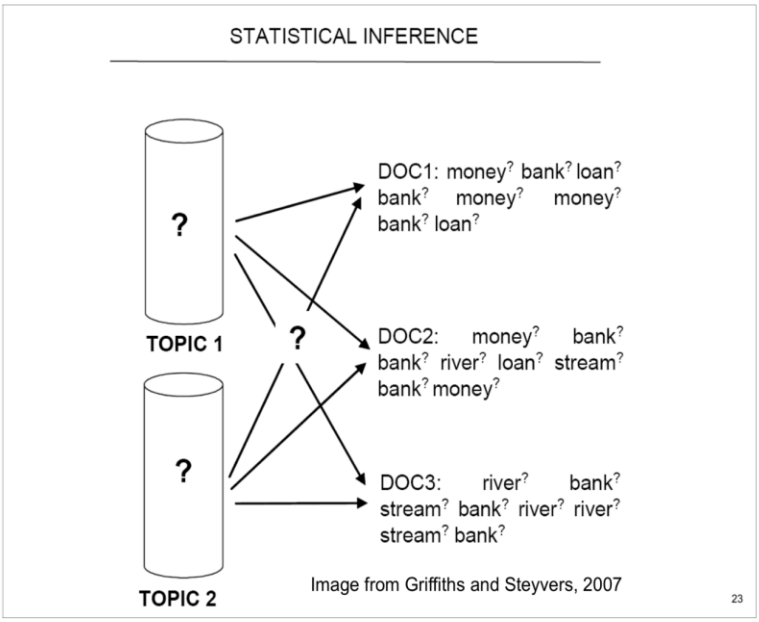

https://thinkinfi.com/

## Topic models

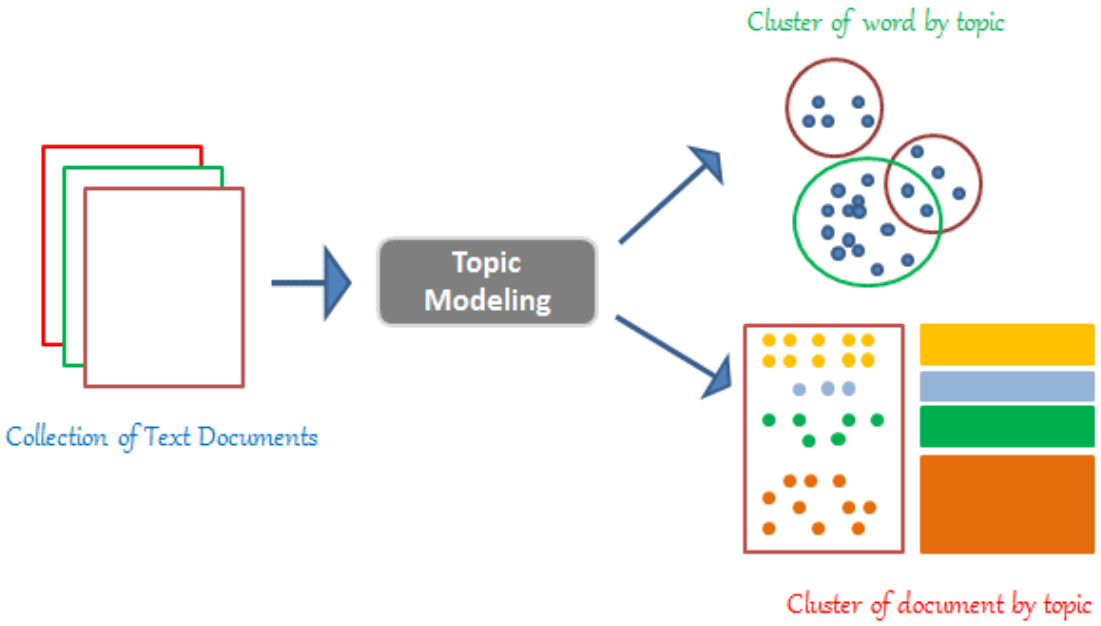
- Three concepts: words, topics, and documents
- Documents are a collection of words and have a probability distribution over topics
- Topics have a probability distribution over words
- Model:
  - Topics made up of words used to generate documents



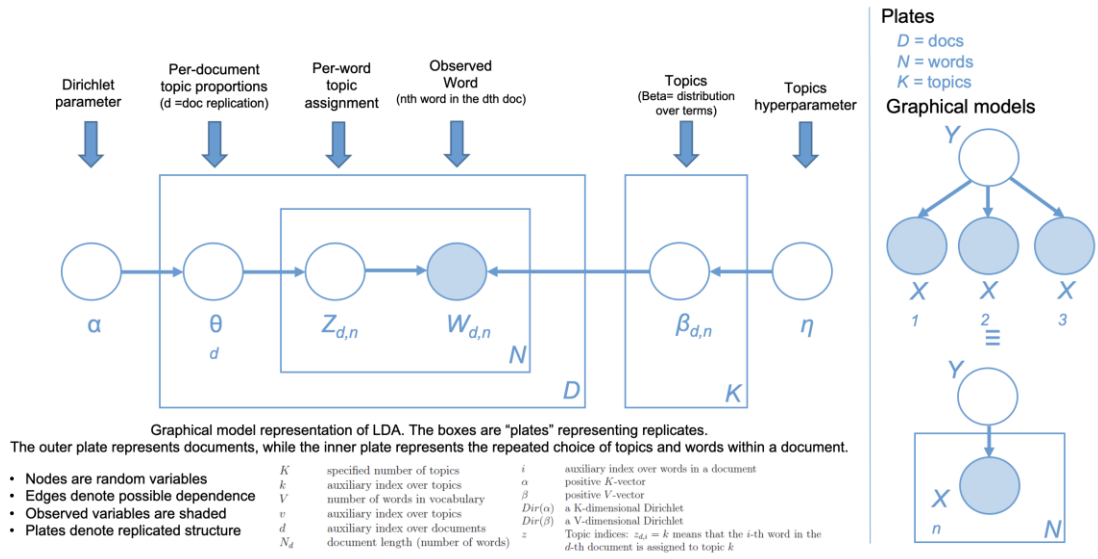

## Topic models | Reality: Documents observed, infer topics



## Topic models




## LDA graphical model

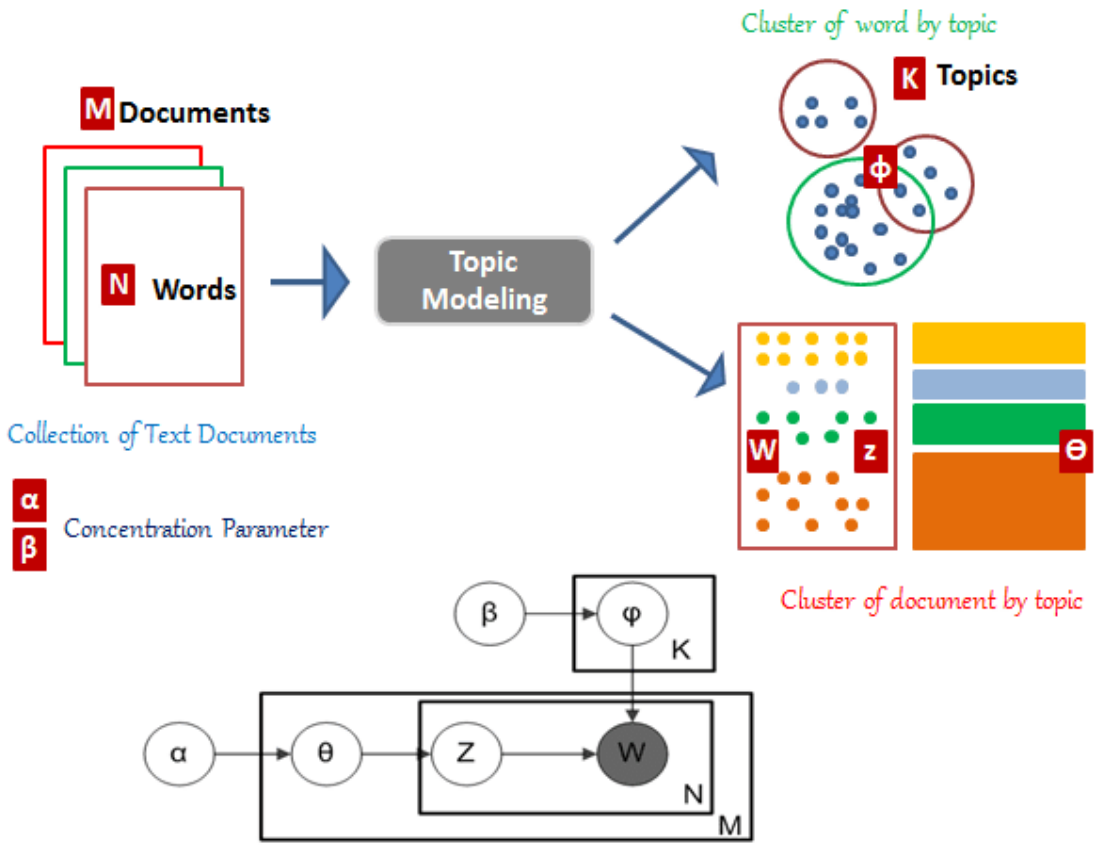



Graphical model representation of LDA. The boxes are "plates" representing replicates.
The outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document.

- Nodes are random variables
- Edges denote possible dependence
- Observed variables are shaded
- Plates denote replicated structure

| | | | |
|---|---|---|---|
| $K$ | specified number of topics | $i$ | auxiliary index over words in a document |
| $k$ | auxiliary index over topics | $\alpha$ | positive $K$-vector |
| $V$ | number of words in vocabulary | $\beta$ | positive $V$-vector |
| $v$ | auxiliary index over topics | $Dir(\alpha)$ | a K-dimensional Dirichlet |
| $d$ | auxiliary index over documents | $Dir(\beta)$ | a V-dimensional Dirichlet |
| $N_d$ | document length (number of words) | $z$ | Topic indices: $z_{d,i} = k$ means that the $i$-th word in the $d$-th document is assigned to topic $k$ |

## LDA




## Probabilistic modeling

1. Treat data as observations that arise from a generative probabilistic process that includes hidden variables: For documents, the hidden variables reflect the thematic structure of the collection.

2. Infer the hidden structure using posterior inference: What are the topics that describe this collection?

3. Situate new data into the estimated model: How does this query or new document fit into the estimated topic structure?

## Example

What is latent Dirichlet allocation? It's a way of automatically discovering topics that these sentences contain.

Suppose you have the following set of sentences:

- I like to eat broccoli and bananas.
- I ate a banana and spinach smoothie for breakfast.
- Chinchillas and kittens are cute.

- My sister adopted a kitten yesterday.
- Look at this cute hamster munching on a piece of broccoli.

## Example

Given these sentences and asked for 2 topics, LDA might produce something like:

- Sentences 1 and 2: 100% Topic A
- Sentences 3 and 4: 100% Topic B
- Sentence 5: 60% Topic A, 40% Topic B
- Topic A: 30% broccoli, 15% bananas, 10% breakfast, 10% munching, … (at which point, you could interpret topic A to be about food)
- Topic B: 20% chinchillas, 20% kittens, 20% cute, 15% hamster, … (at which point, you could interpret topic B to be about cute animals)

How does LDA perform this discovery?

## LDA training

- Go through each document, and randomly assign each word in the document to one of the K topics.
- Notice that this random assignment already gives you both topic representations of all the documents and word distributions of all the topics (albeit not very good ones).
- So to improve on them, for each document d…
- Go through each word w in d…

## LDA training

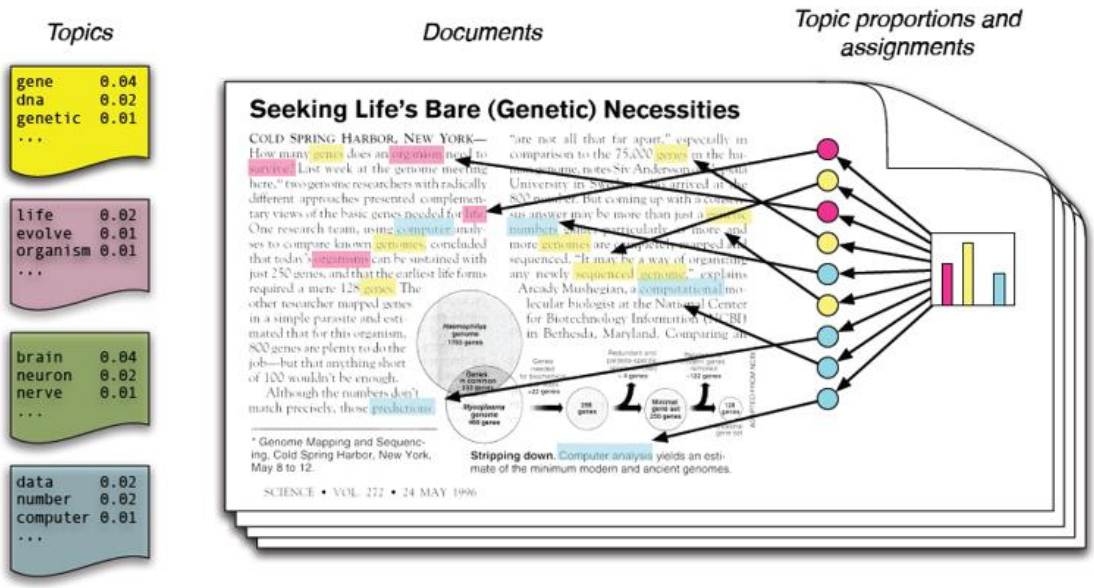
- And for each topic t, compute two things:
  - p(topic t | document d) = the proportion of words in document d that are currently assigned to topic t, and
  - p(word w | topic t) = the proportion of assignments to topic t over all documents that come from this word w.
- Reassign w a new topic, where we choose topic t with probability p(topic t | document d) * p(word w | topic t)
- In other words, in this step, we're assuming that all topic assignments except for the current word in question are correct, and then updating the assignment of the current word using our model of how documents are generated.

## LDA training

- After repeating the previous step a large number of times, you'll eventually reach a roughly steady state where your assignments are pretty good.

- Use these assignments to estimate the topic mixtures of each document (by counting the proportion of words assigned to each topic within that document) and the words associated to each topic (by counting the proportion of words assigned to each topic overall).

## LDA: Identifying structure in text




## Variations of LDA

- Hierarchical LDA (hLDA): automatically mine the hierarchical dimension of topics
- Supervised LDA (sLDA): learn topics that are inline with the class label
- Hybrid LDA: extracting topics and other information
- LDA & BERT: we cover deep learning and BERT later

## BERTtopic

- https://github.com/MaartenGr/BERTopic
- BERTopic is a topic modeling technique that leverages 🤗 transformers
- creates dense clusters
- allowing for easily interpretable topics
- https://colab.research.google.com/drive/1BoQ_vakEVtojsd2x_U6-_x52OOuqruj2?usp=sharing

## LDA in Python

### sklearn.decomposition.LatentDirichletAllocation

*class* sklearn.decomposition.**LatentDirichletAllocation**(*n_components=10, *, doc_topic_prior=None, topic_word_prior=None, learning_method='batch', learning_decay=0.7, learning_offset=10.0, max_iter=10, batch_size=128, evaluate_every=-1, total_samples=1000000.0, perp_tol=0.1, mean_change_tol=0.001, max_doc_update_iter=100, n_jobs=None, verbose=0, random_state=None*) [source]

**Examples**

```
>>> from sklearn.decomposition import LatentDirichletAllocation
>>> from sklearn.datasets import make_multilabel_classification
>>> # This produces a feature matrix of token counts, similar to what
>>> # CountVectorizer would produce on text.
>>> X, _ = make_multilabel_classification(random_state=0)
>>> lda = LatentDirichletAllocation(n_components=5,
...     random_state=0)
>>> lda.fit(X)
LatentDirichletAllocation(...)
>>> # get topics for some given samples:
>>> lda.transform(X[-2:])
array([[0.00360392, 0.25499205, 0.0036211 , 0.64236448, 0.09541846],
       [0.15297572, 0.00362644, 0.44412786, 0.39568399, 0.003586  ]])
```

## BERTtopic in Python

```
from bertopic import BERTopic
from sklearn.datasets import fetch_20newsgroups

docs = fetch_20newsgroups(subset='all',  remove=('headers', 'footers', 'quotes'))['data']

topic_model = BERTopic()
topics, probs = topic_model.fit_transform(docs)
```

```
>>> topic_model.get_topic_info()

Topic   Count   Name
-1      4630    -1_can_your_will_any
0       693     49_windows_drive_dos_file
1       466     32_jesus_bible_christian_faith
2       441     2_space_launch_orbit_lunar
3       381     22_key_encryption_keys_encrypted
...
```

# Cluster Validation

## Desirable properties of clustering

- Scalability
  - Both in time and space
- Ability to deal with various types of data
  - No/less assumption about input data

  - Minimal requirement about domain knowledge

- Interpretability and usability

## What is a good clustering?

- Internal criterion: A good clustering will produce high quality clusters in which:
  - The intra-class (that is, intra-cluster) similarity is high
  - The inter-class similarity is low
  - The measured quality of a clustering depends on both the document representation and the similarity measure used

## Cluster validation

- Criteria to determine whether the clusters are meaningful
  - Internal validation
    - Stability and coherence
  - External validation
    - Match with known categories

## Internal validation

- Coherence
  - Inter-cluster similarity v.s. intra-cluster similarity
  - Davies–Bouldin index
    - $DB = \frac{1}{k}\sum_{i=1}^{k} \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right)$ ← Evaluate every pair of clusters
      - where $k$ is total number of clusters, $\sigma_i$ is average distance of all elements in cluster $i$ from the cluster center, $d(c_i, c_j)$ is the distance between cluster centroid $c_i$ and $c_j$.

We prefer smaller DB-index!

## External criteria for clustering quality

- Quality measured by its ability to discover some or all of the hidden patterns or latent classes in gold standard data
- Assesses a clustering with respect to ground truth … requires labeled data
- Assume documents with $C$ gold standard classes, while our clustering algorithms produce $K$ clusters, $\omega_1, \omega_2, \ldots, \omega_K$ with $n_i$ members.

### Clustering performance evaluation in SKlearn

- https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation
- Rand index
- Mutual Information based scores
- Homogeneity, completeness and V-measure
- Fowlkes-Mallows scores
- Silhouette Coefficient
- Calinski-Harabasz Index
- Davies-Bouldin Index
- Contingency Matrix
- Pair Confusion Matrix

## Summary

### Summary

- Text clustering
- In clustering, clusters are inferred from the data without human input (unsupervised learning)
- Many ways of influencing the outcome of clustering: number of clusters, similarity measure, representation of documents
- Evaluation

## Practical 4