Clustering and Topic Modelling

Giovanni Colavizza

Text Mining
Amsterdam University College

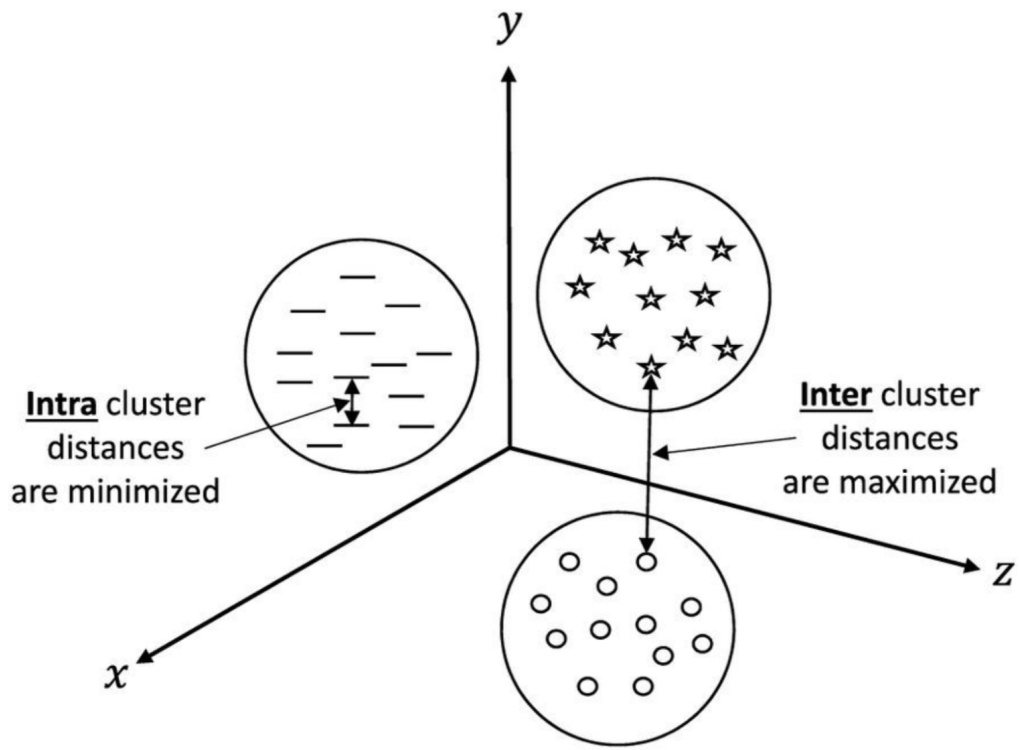
February 5, 2021

Overview

Clustering

Definition

- **Clustering** is the unsupervised assignment of data points into groups on the basis of their mutual similarity.
- Intuitively, in a good clustering solution, data points should be:
  - similar to those belonging to the same group;
  - dissimilar to those belonging to other groups.



Source: Galvan-Nunez & Attoh-Okine (2016)

Stages

- Task definition: how many clusters? What do we expect to find?
- Data representation: what features to use?
- Assessing similarity: which measure to use?
- Grouping: which clustering algorithm to use?
- Data abstraction: how can we represent the clusters? E.g., from a human perspective, how to intuitively understand groups? Possible solution: label them.
- Output assessment: how should the output of a given algorithm be evaluated?

Approaches

- Hard vs. Soft clustering:
  1. **Hard clustering** algorithms assign each object (pattern, document, user...) to exactly one group (e.g., K-means).
  2. **Soft clustering** algorithms assign each object to one or more groups with a certain probability (e.g., topic modeling).
- Flat vs. Hierarchical clustering:
  1. **Flat clustering** algorithms return a set of clusters, without any explicit structure.
  2. **Hierarchical clustering** algorithms create a hierarchy of clusters. Agglomerative approaches generates clusters in a bottom-up fashion, while divisive approaches generates clusters in a top-down fashion.

Evaluation metrics

- Clustering algorithms try to maximize:
  - Coherence: how similar are members of the same cluster?
  - Separation: how distant are members of different cluster?
  - Utility: how useful are the automatically discovered clusters?
- These can be evaluated by using:
  - Internal evaluation measures build on coherence and separation.
  - External evaluation measures: the clustering is evaluated against a gold standard or in an application of interest.

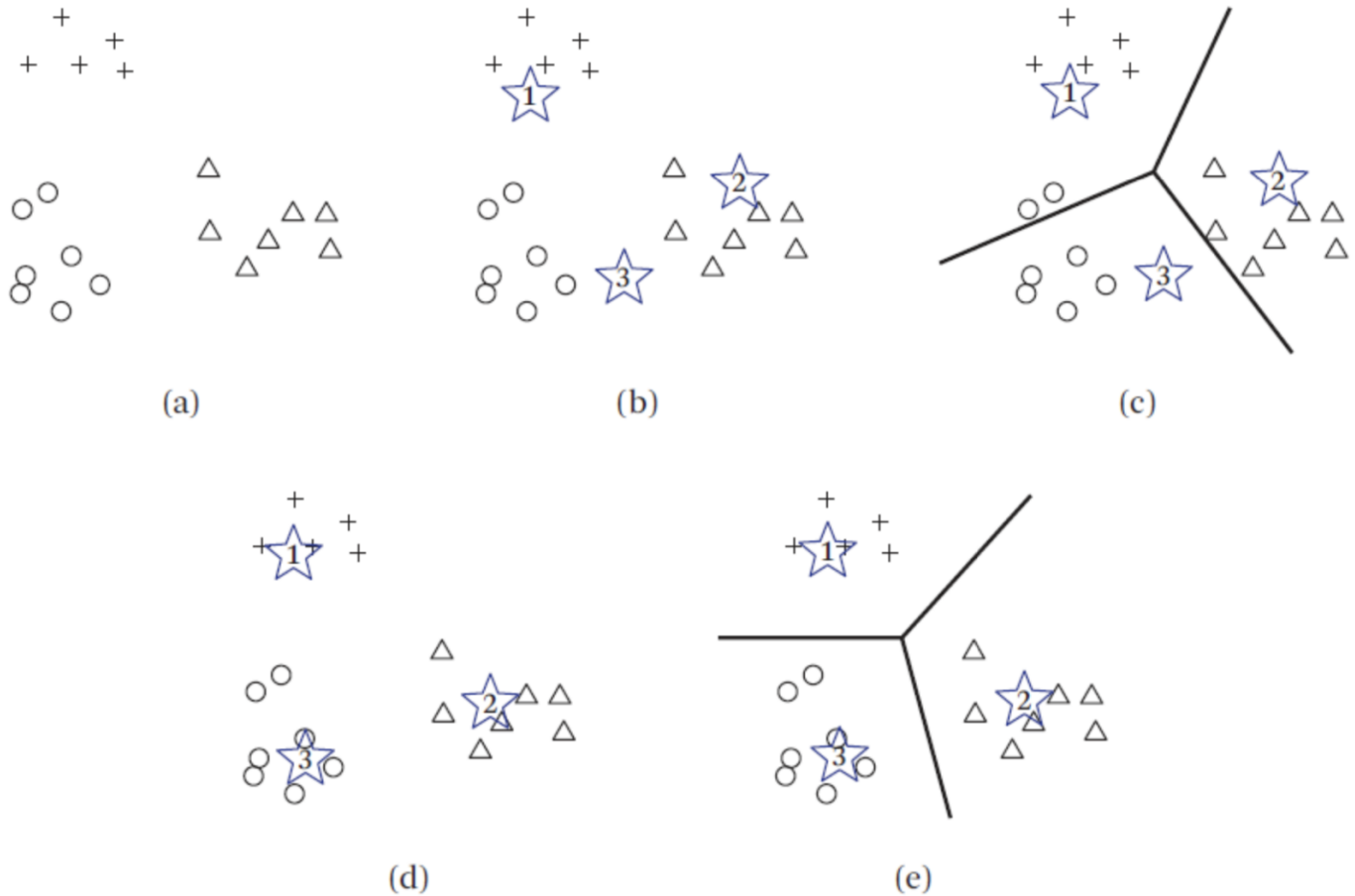
K-means

Description

- A well-known flat clustering algorithm.
- It requires the number (K) of desired clusters to be known in advance.
- Objective: to minimize the average squared Euclidean distance of the data points from the *centroid* of their cluster. centroid: a representation of all the documents of a cluster usually obtained by averaging all the members' features.
- Strength: easy to implement and computationally efficient.
- How does it work? Initialize K randomly selected centroids (*seeds*); *while not converge do:*
  1. *Assign each item to the cluster whose centroid is closest to it.*
  2. *Recompute centroids of the new cluster found from previous step.*

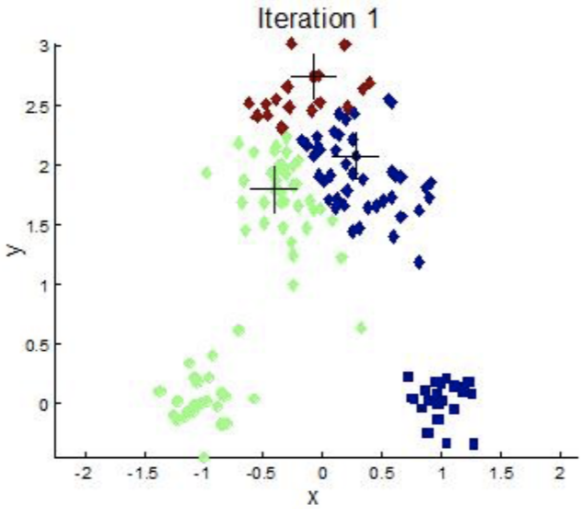
  end while.

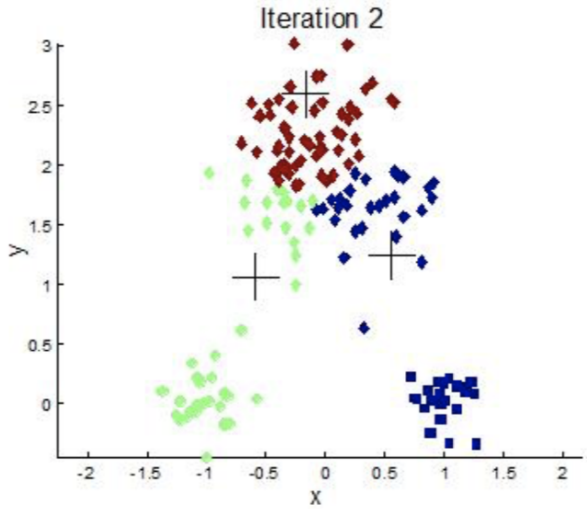
Illustration

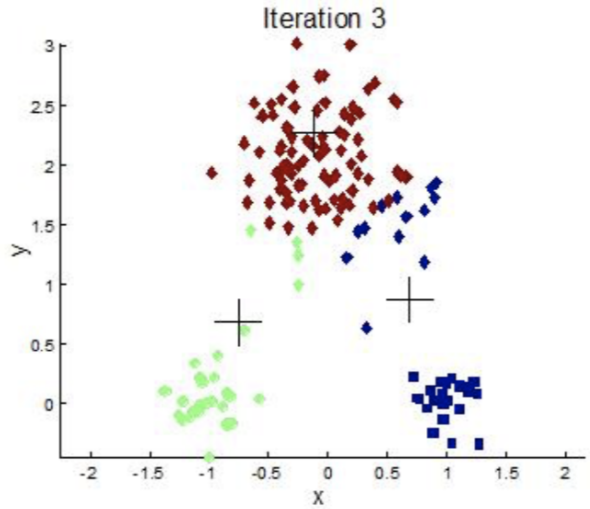


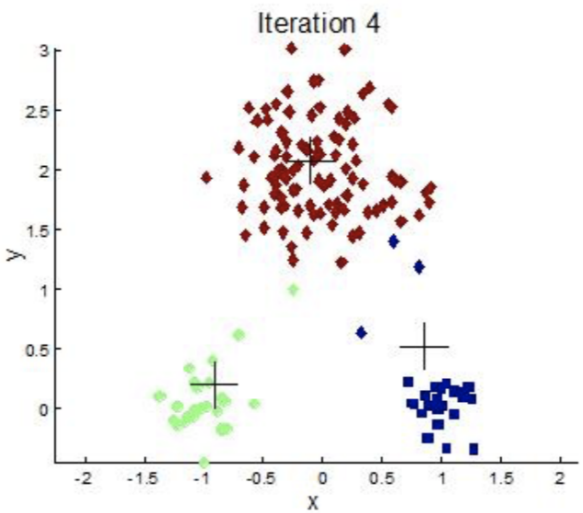
source: Zhai & Massung (2016: 283)

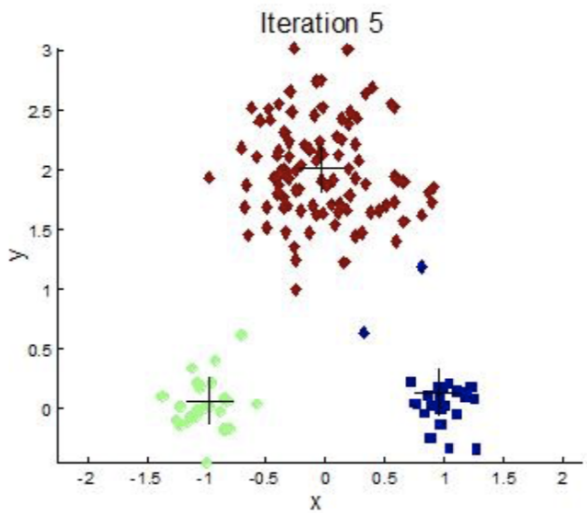
Illustration

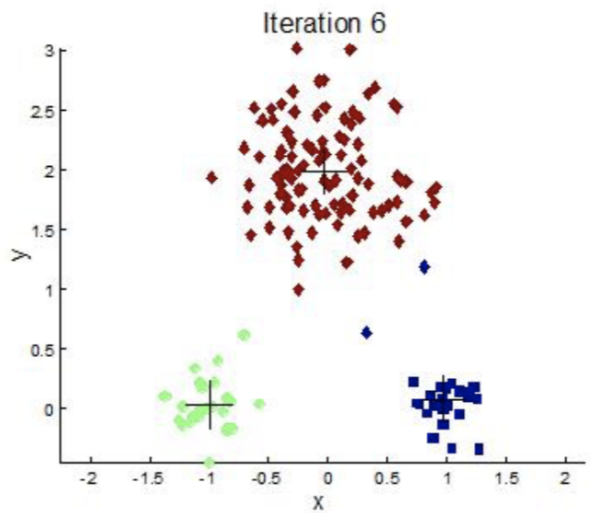








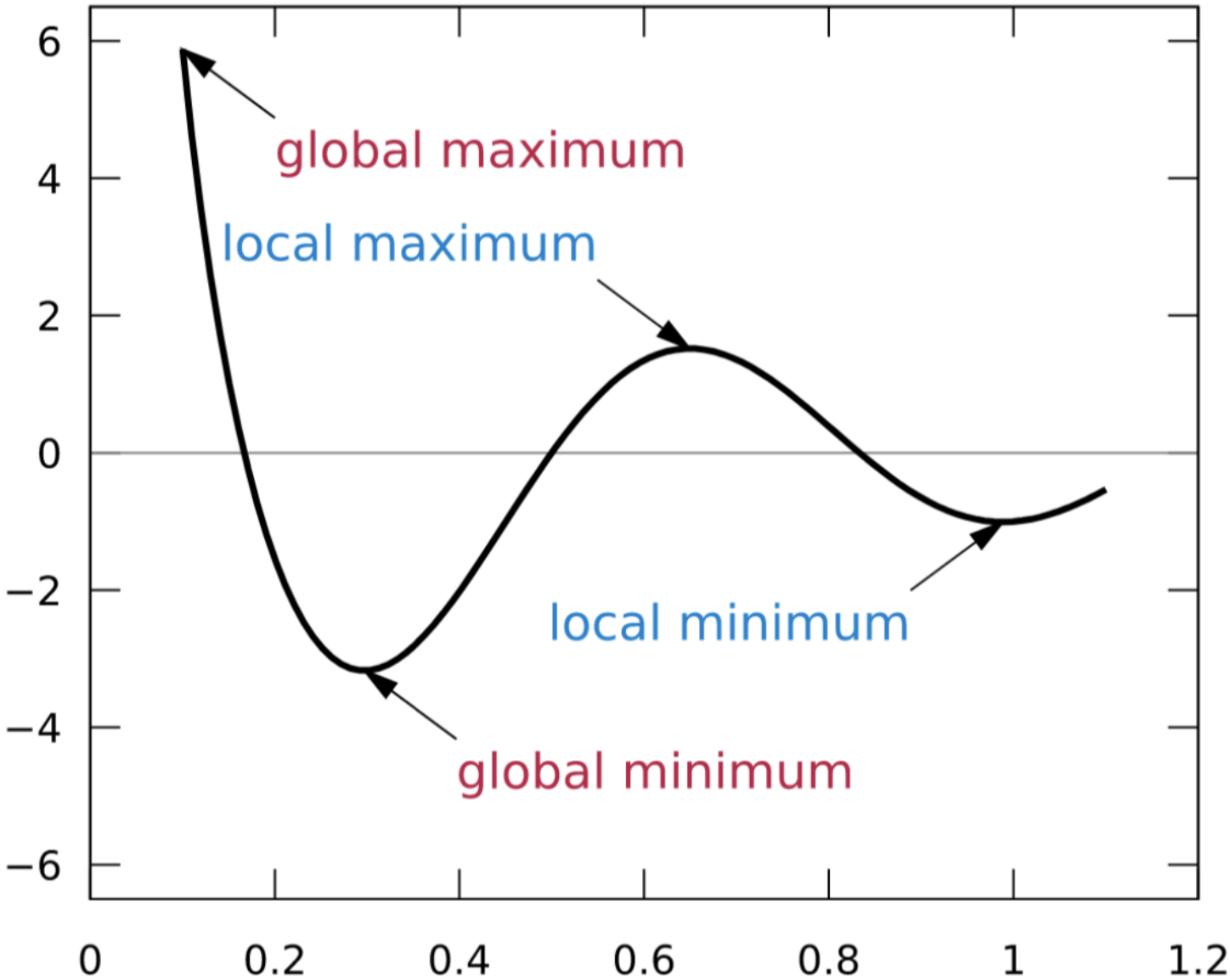




Termination conditions

- Stop after a given number of iterations: efficient but quality may suffer.
- Stop when cluster composition or centroids do not change between iterations: better results but may be unacceptably long or may be stuck in local minima.
- Residual Sum of Squares (RSS) below a given threshold. RSS: sum of the squared distances between each vector-centroid pairs; useful to specify a given cluster quality, but may be unacceptably long.

Minima


global maximum
local maximum
local minimum
global minimum
6
4
2
0
−2
−4
−6
0
0.2
0.4
0.6
0.8
1
1.2

Issues

- There is no guarantee for K-means to converge to a global minimum: influence of outliers, different initializations can have very different results, if we chose an outlier as initial centroid we might end up with a singleton cluster.
- Proposed solutions: remove outliers from data, try multiple iterations and choose the one with lower cost, obtain seeds from other methods (e.g. hierarchical clustering), select more than K clusters and choose the K that minimized the cost.

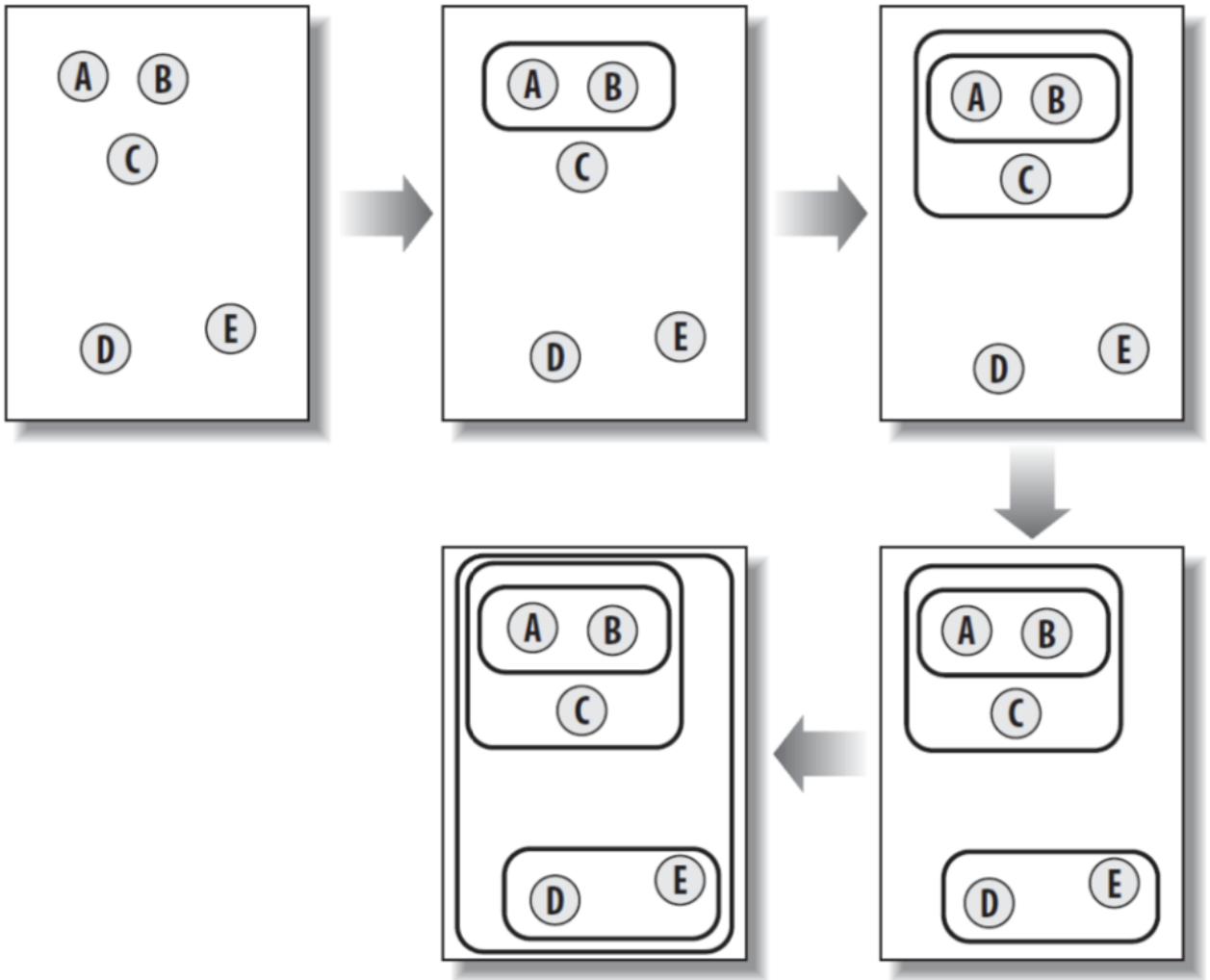
Hierarchical clustering

Description

- The output of the clustering algorithm is a structure of nested clusters that can be visualized as a **hierarchical tree**.
- No pre-determined number of clusters must be specified.
- Two families of algorithms:
  - **Agglomerative**: you start with a cluster for each data point and progressively merge the two most similar pairs of clusters.
  - **Divisive**: a flat clustering algorithm is repeatedly applied to partition the data into smaller and smaller clusters.
- Initialize each cluster to be a singleton;
  while more than one clusters exist do:
  1. *Find the two most similar clusters.*
  2. *Merge these two clusters.*

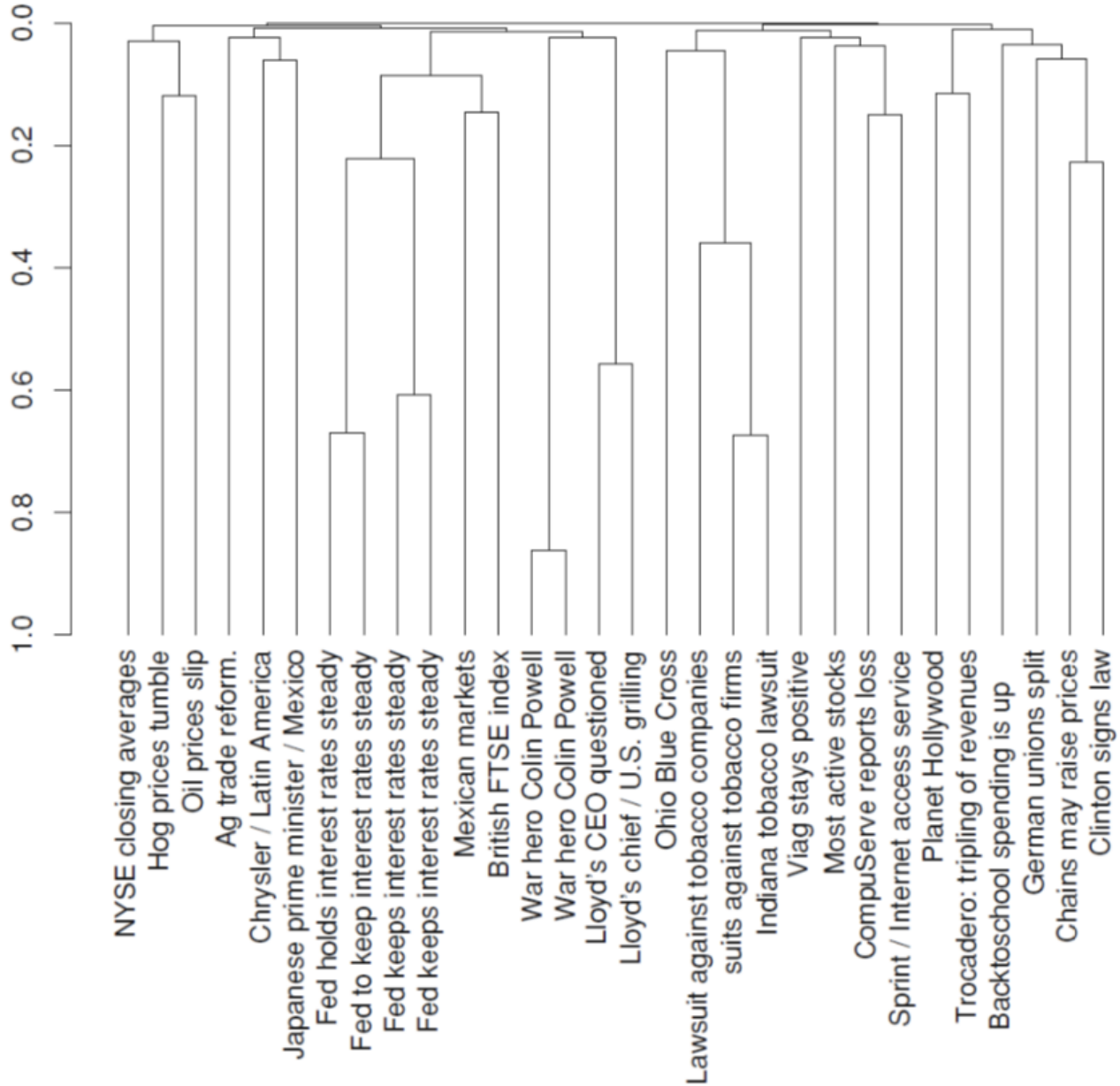
  end while.

Illustration

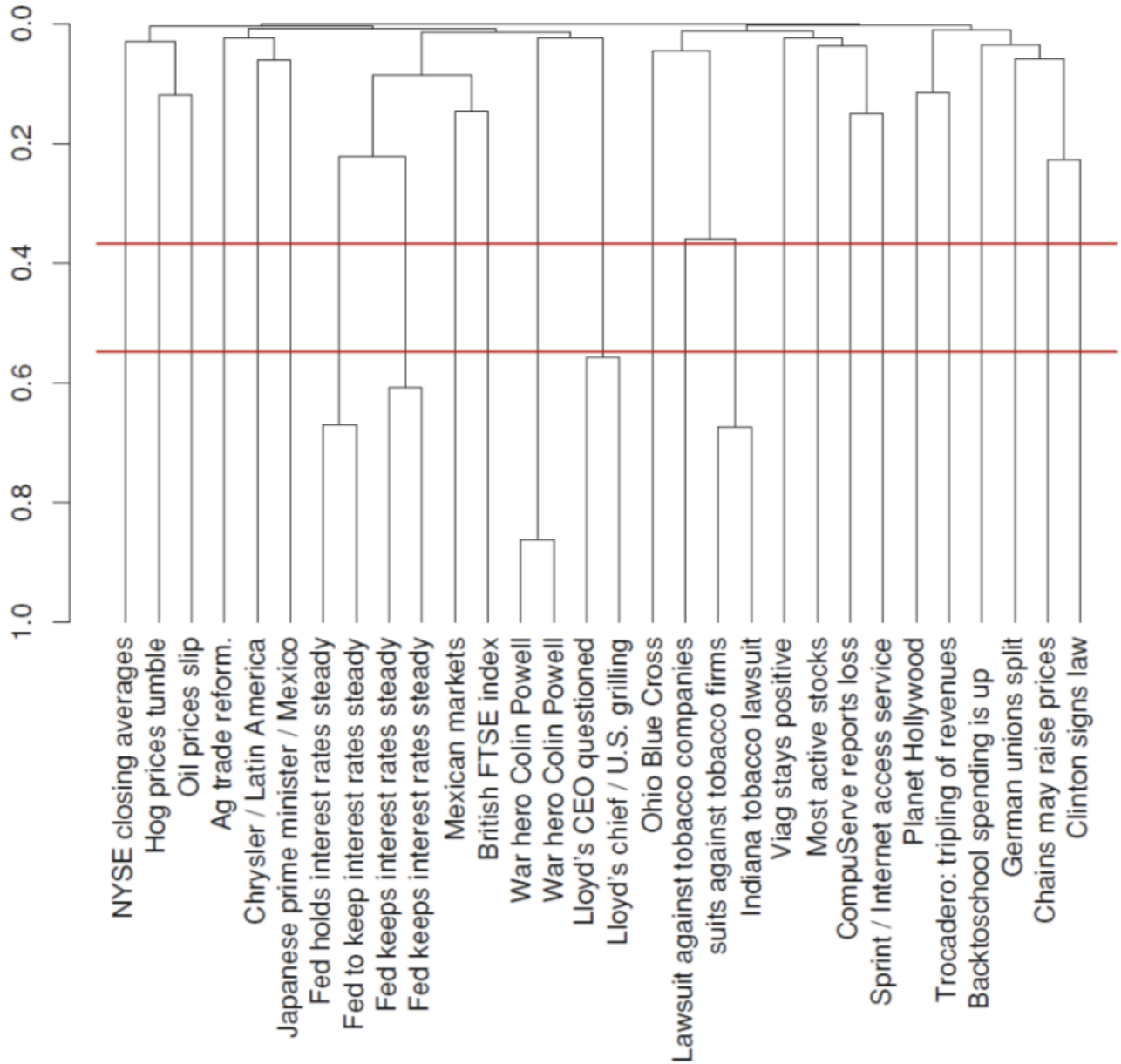


source: Segaran (2007: 33)

Illustration

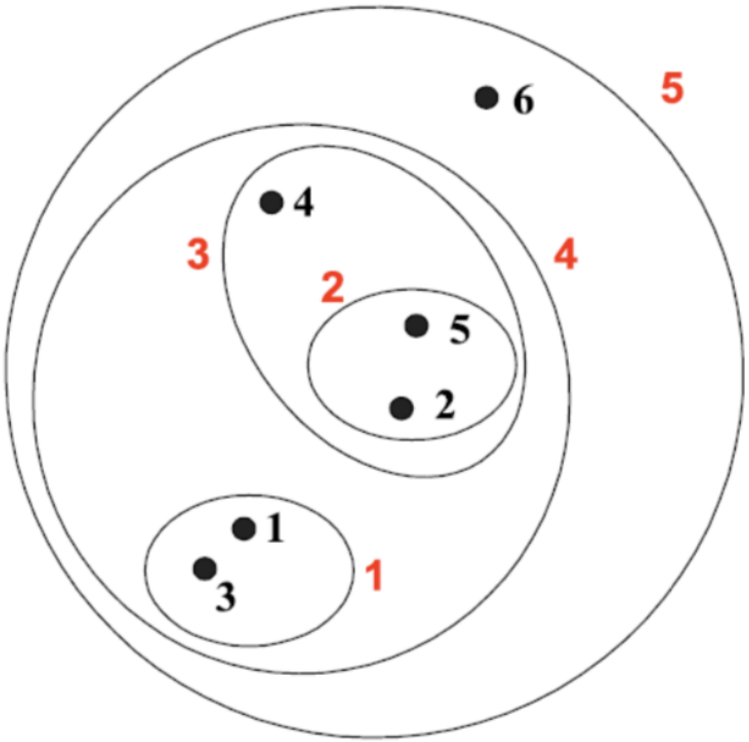


Illustration



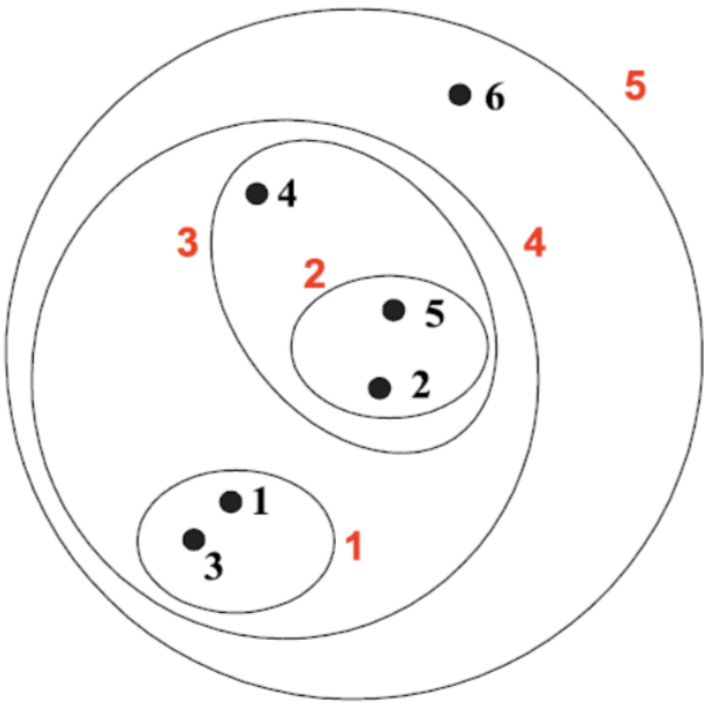
Illustration

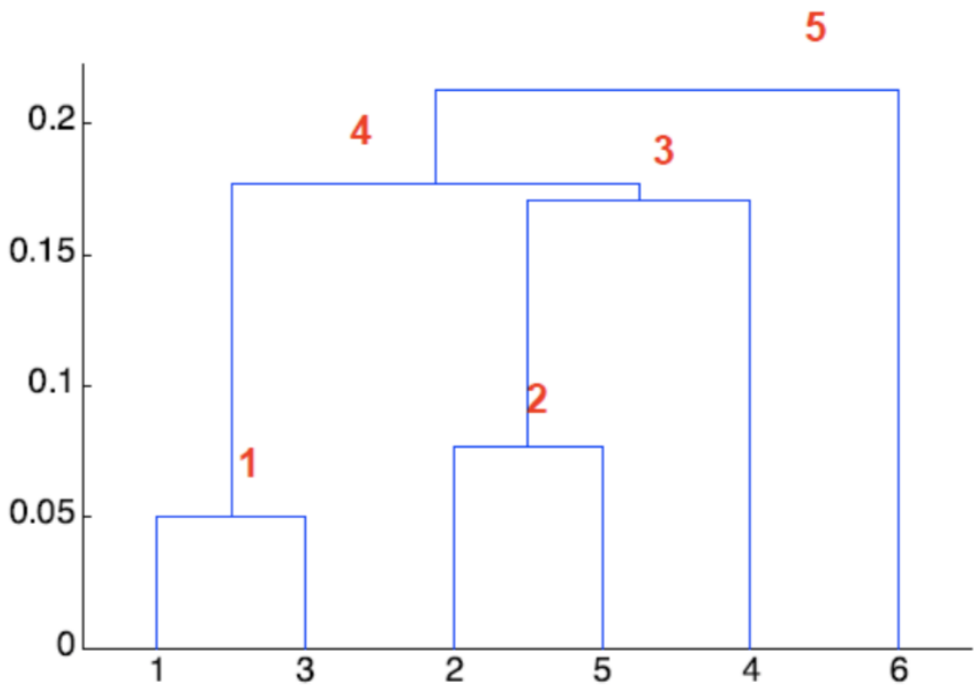
Quiz! draw a dendrogram for this



Illustration

Quiz! draw a dendrogram for this

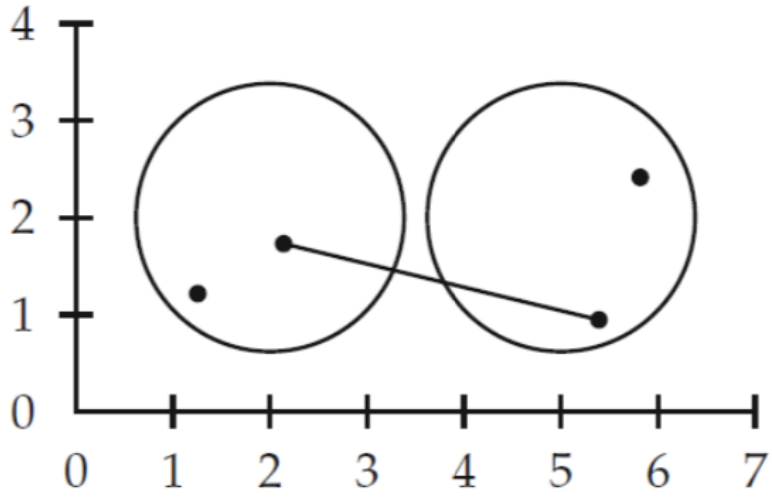




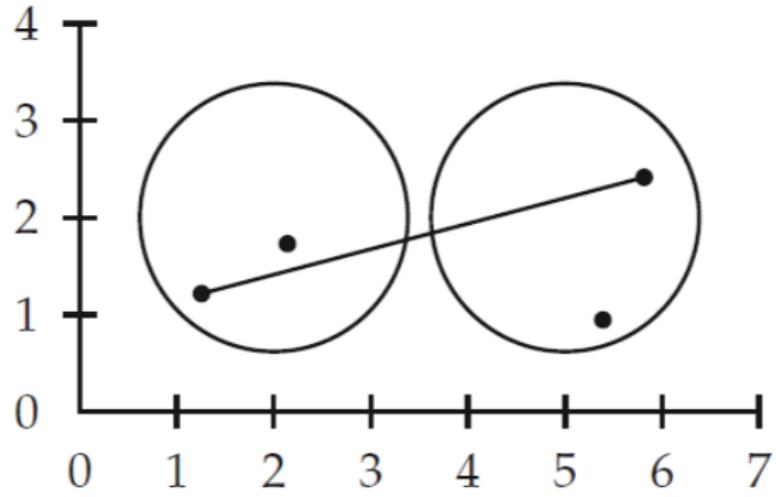
Linkage criteria

- Hierarchical clustering algorithms can be classified on the basis of the *strategy used to identify the most similar clusters* at every step:
  - *Single linkage*: merge the two clusters with the smallest distance between the two most similar data points.
  - *Complete linkage*: merge the two clusters with the smallest distance between the two most dissimilar data points.
  - *Average linkage*: merge the two clusters with the smallest average pairwise distance between all the data points of the two clusters, including pairs from the same cluster.
  - *Centroid linkage*: merge the two clusters with the smallest distance between their centroids; equivalent to average similarity of all pairs of documents from different clusters.

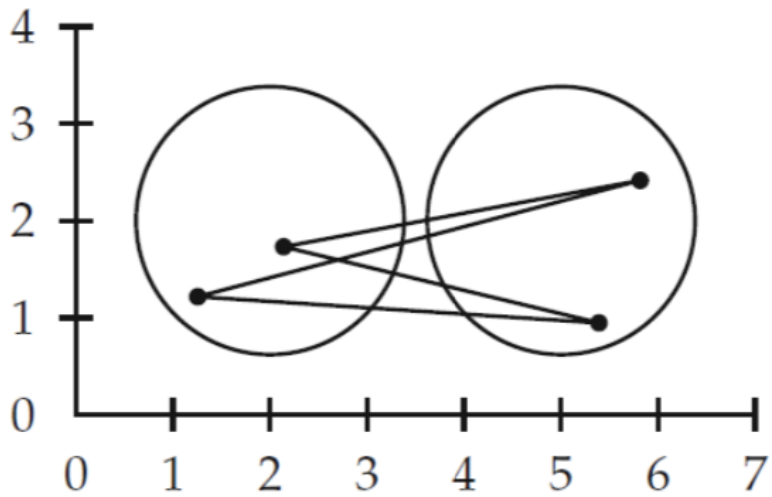
Linkage criteria



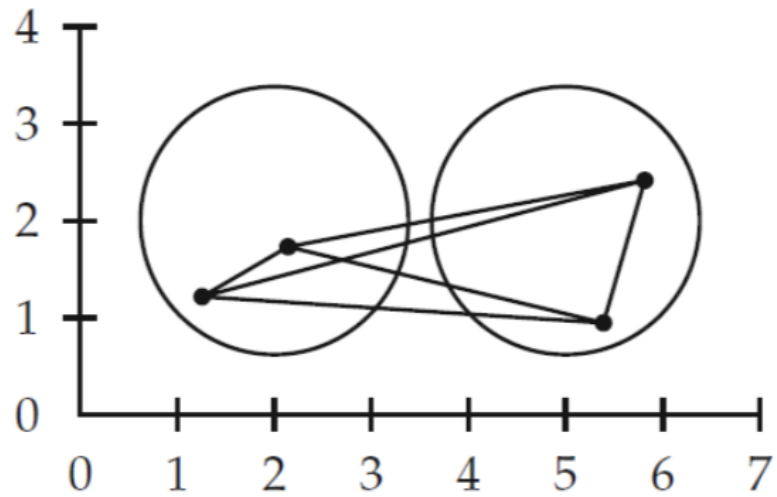
(a) single link: maximum similarity



(b) complete link: minimum similarity



(c) centroid: average inter-similarity



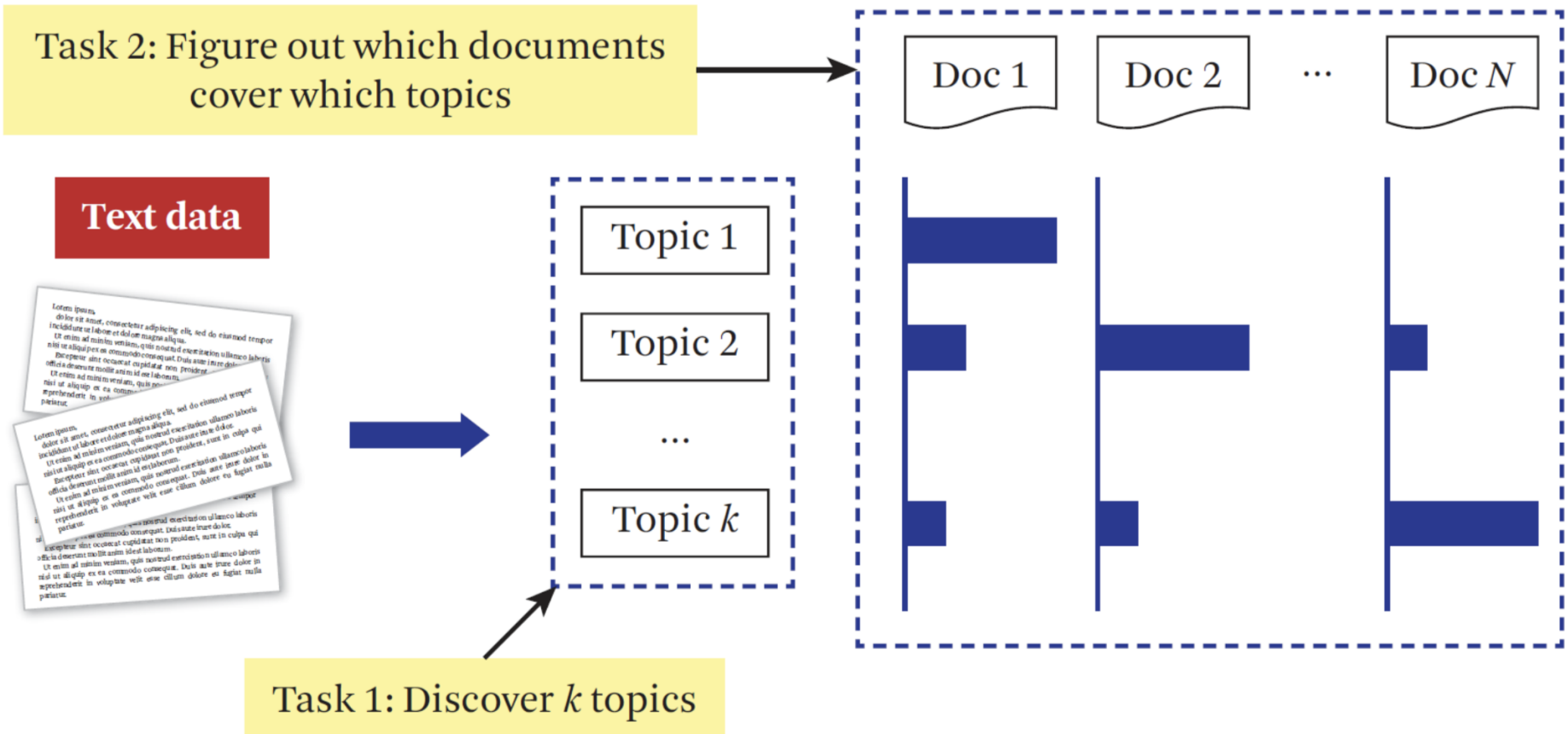
(d) group-average: average of all similarities

Comparison

- K-means is much more computationally efficient than HC: the time complexity of K-means is linear, while that of HC is quadratic.
- In HC, when two clusters are combined, the merged cluster cannot be undone.
- In K-means, different initializations can lead to dramatically different results, while HC is deterministic (results are reproducible).
- K-means works better when the shape of the clusters is hyper spherical, while HC may return meaningful taxonomies.
- K-means requires a pre-determined target number of clusters.

Topic modelling

Illustration

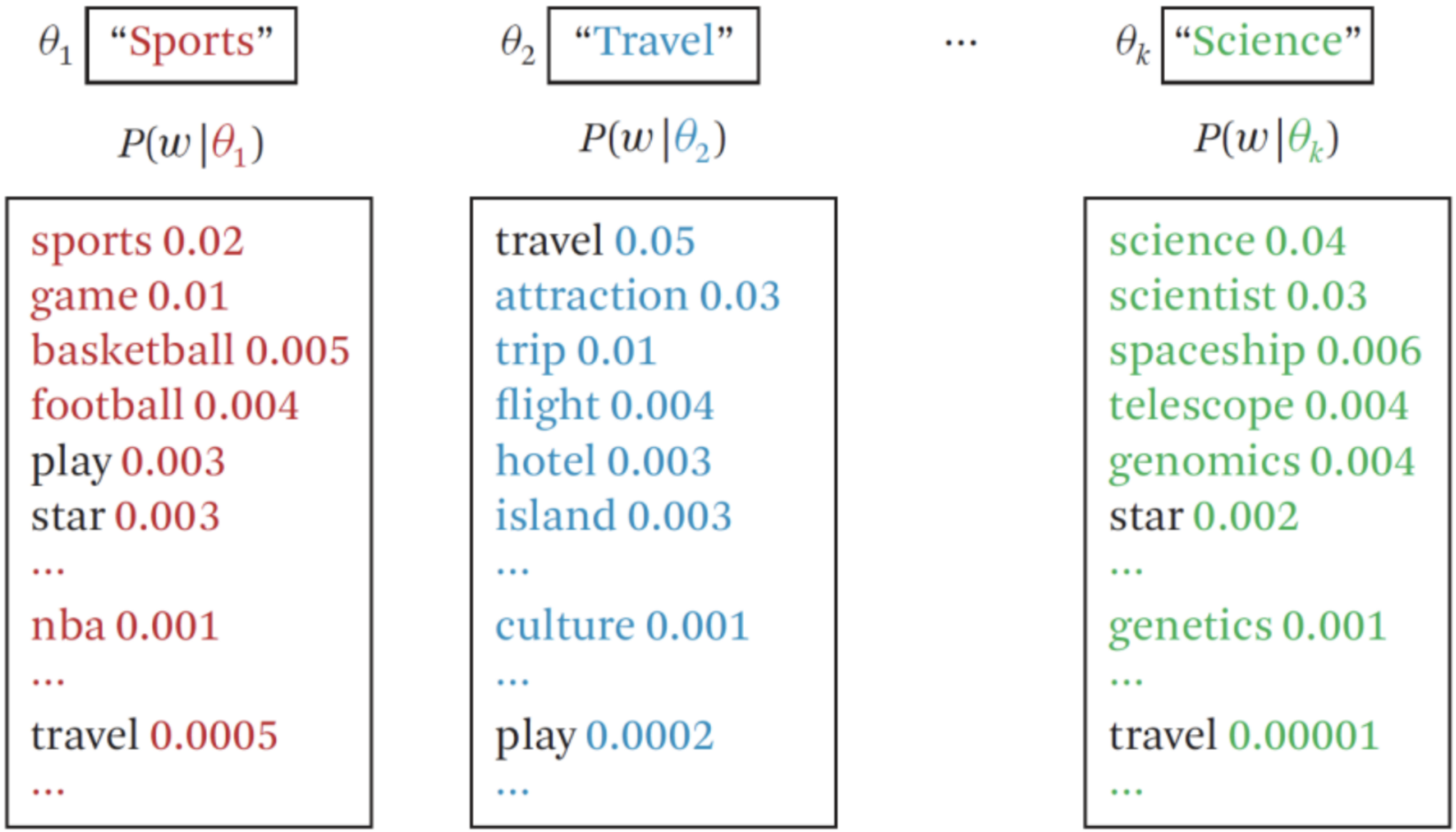


Description

- **Input**:
  1. A collection of N of text documents $D = \{d_1, \ldots, d_N\}$
  2. The number of topics K (a hyperparameter).
- **Output**:
  1. The topics $\{\theta_1, \ldots, \theta_K\}$.
  2. The topic coverage for every document d_i : $\{\pi_{i1}, \ldots, \pi_{iK}\}$, so that $\sum_j^K \pi_{ij} = 1$.
- But then, what is a **topic** θ_i?

Topics as word distributions

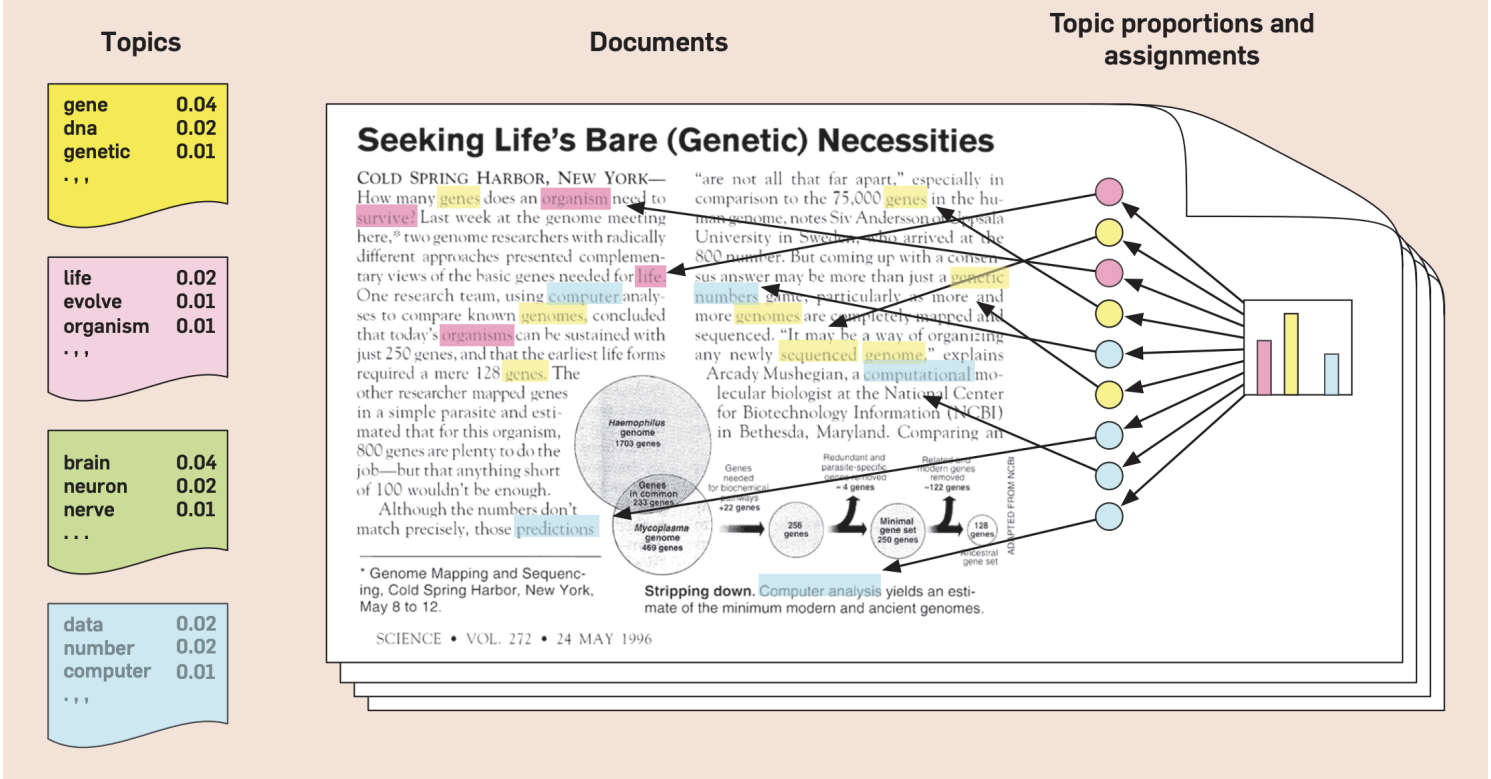
But then, what is a **topic** θ_i? $\sum_{w \in V} p(w|\theta_i) = 1$



Generative modelling

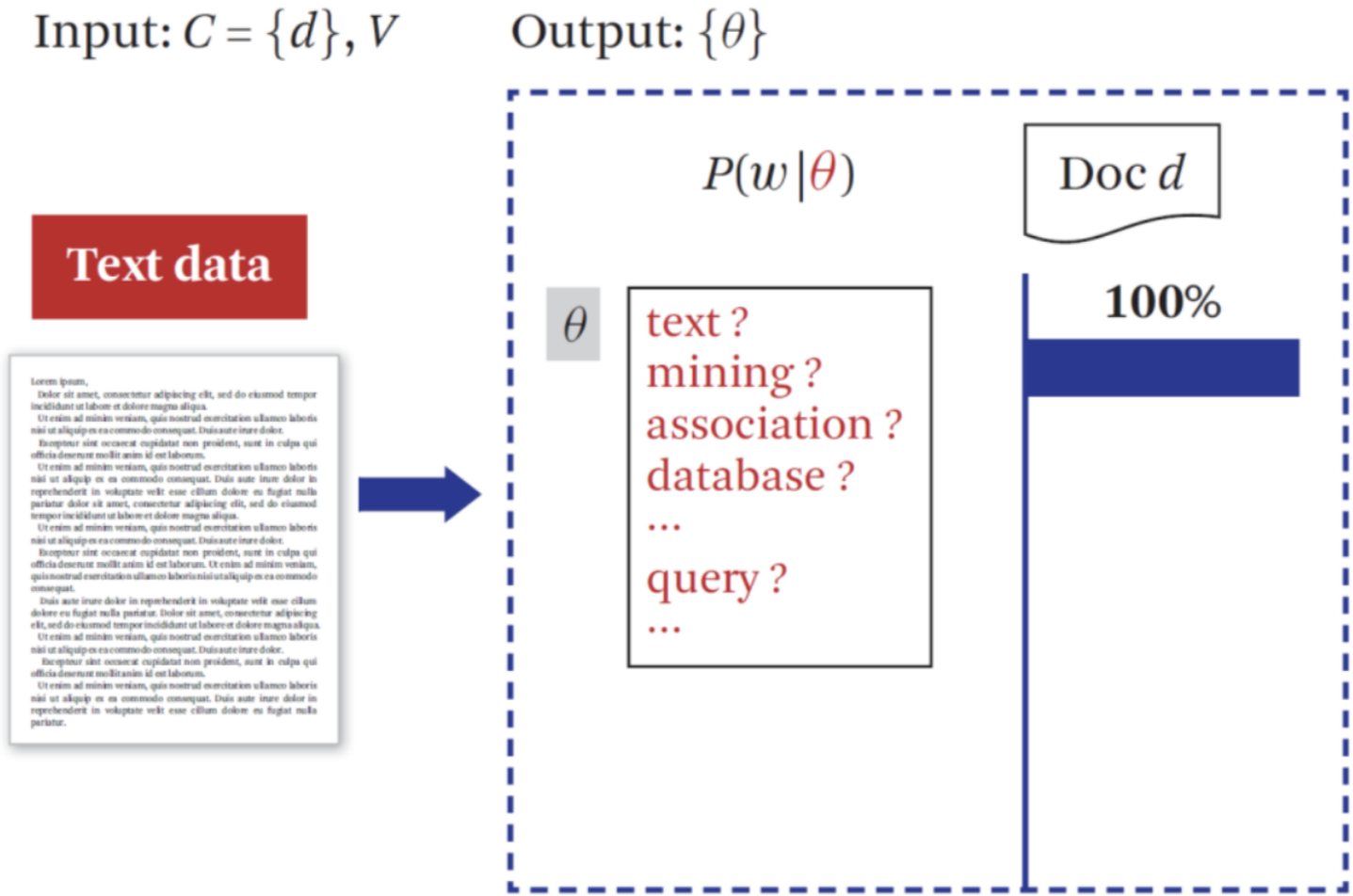
- We can design a **generative model** for our data;
- describing how data are generated;
- using a set Λ of parameters controlling the behavior of the model.

Figure 1. The intuitions behind latent Dirichlet allocation. We assume that some number of "topics," which are distributions over words, exist for the whole collection (far left). Each document is assumed to be generated as follows. First choose a distribution over the topics (the histogram at right); then, for each word, choose a topic assignment (the colored coins) and choose the word from the corresponding topic. The topics and topic assignments in this figure are illustrative—they are not fit from real data. See Figure 2 for topics fit from data.



Topic modelling (optional)

One document, one topic



One document, one topic

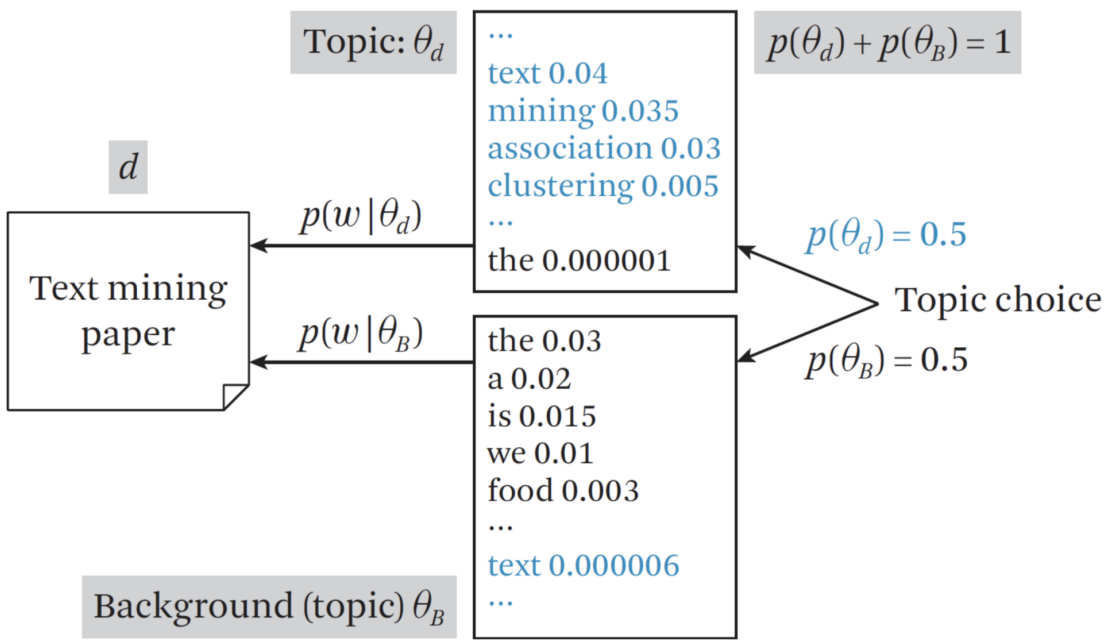
- Let us start simple with one document with one topic.
- **Data**: document $d = x_1, x_2, \ldots, x_{|d|}$, $x_i \in V = \{w_1, \ldots, w_M\}$, where $M = |V|$.
- **Model**: our topic θ is a (usually unigram) LM $\theta : \{\delta_i = p(w_i|\theta)\}, i = 1, \ldots, M; \sum_j \delta_j = 1$.
- **Likelihood function**:

$$\begin{aligned} p(d|\theta) &= p(x_1|\theta) \times \cdots \times p(x_{|d|}|\theta) \\ &= p(w_1|\theta)^{c(w_1,d)} \times \cdots \times p(w_M|\theta)^{c(w_M,d)} \\ &= \prod_{i=1}^{M} p(w_i|\theta)^{c(w_i,d)} = \prod_{i=1}^{M} \delta_i^{c(w_i,d)} \end{aligned}$$

- Where $c(w_i, d)$ is the frequency count of word w_i in document d.
- We can then use MLE to estimate our parameters $(\delta_i, \ldots, \delta_M)$.

Mixture models

- A major issue with this (unigram) model is that higher-rank positions are occupied by very frequent common words (e.g., function words).
- Solution: a **mixture model**, i.e., a model obtained by mixing different component models: common words are generated from a **background word distribution** (i.e., another unigram language model), while the **topic word distribution** takes care of the generation of the content-carrying topical words.

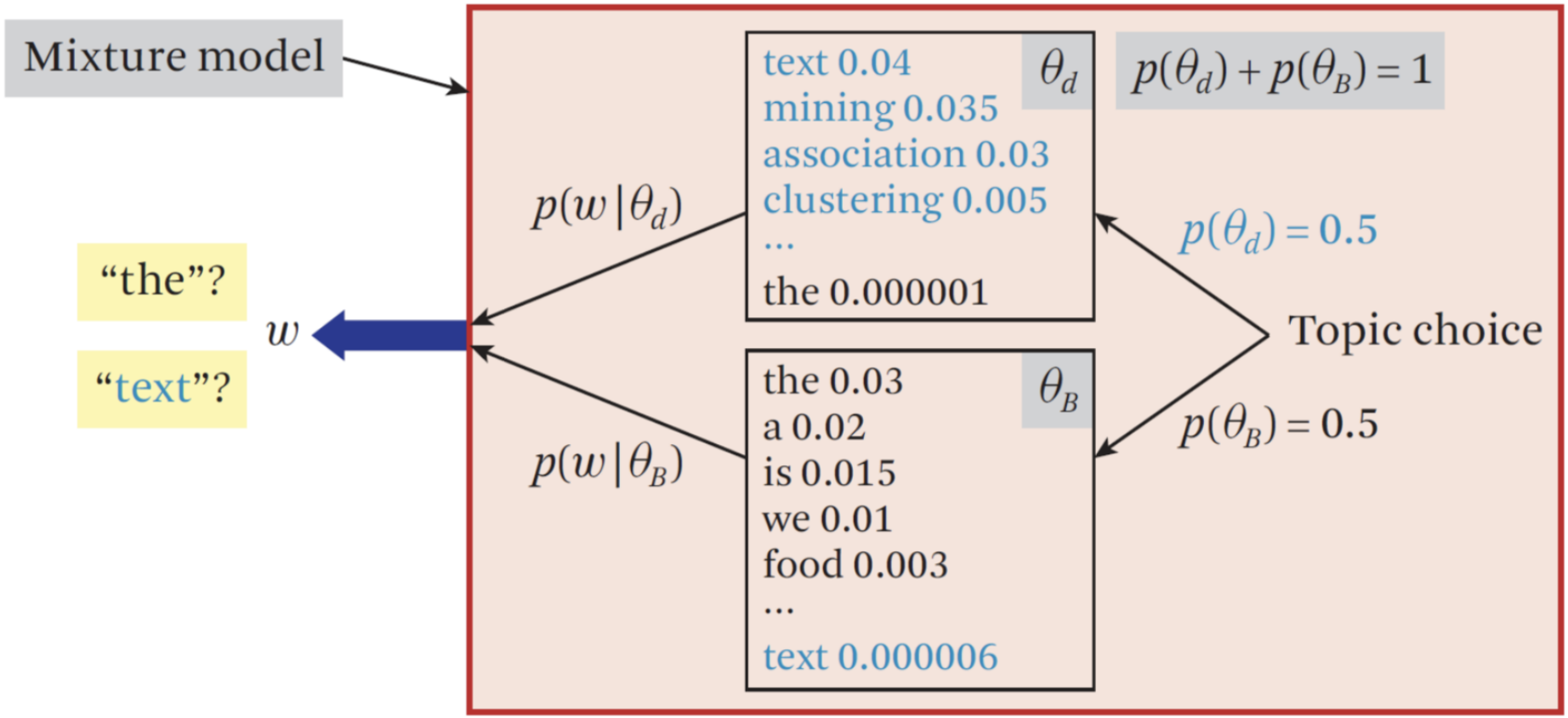


Mixture models

- How does the word generation process change? A new probability distribution over the choices of components controls the choice of the distribution from which words are pulled.
- In our case, this component models the probability of θ_d (of using the topic model) and of θ_B (i.e., of using the background model): $p(\theta_d) + p(\theta_B) = 1$.
- The probability of observing any word w in the document:

$$p(w) = \underbrace{p(\theta_B)\,p(w \mid \theta_B)}_{\text{the probability of observing word } w \text{ from the background language model}} + \underbrace{p(\theta_d)\,p(w \mid \theta_d)}_{\text{the probability of observing word } w \text{ from the topic word distribution}}$$

Mixture models



A simple mixture model

- **Data**: document $d = x_1, x_2, \ldots, x_{|d|}$, $x_i \in V = \{w_1, \ldots, w_M\}$, where $M = |V|$.
- **Model** with parameters Λ:
  - Two LMs defined as before: θ_d (topic model); θ_B (background model).
  - Two mixing weights: $p(\theta_d) + p(\theta_B) = 1$.
- **Likelihood function**:

$$p(d|\Lambda) = \prod_{i=1}^{|d|} p(x_i|\Lambda) = \prod_{i=1}^{|d|} [p(\theta_d)p(x_i|\theta_d) + p(\theta_B)p(x_i|\theta_B)]$$

$$= \prod_{i=1}^{M} [p(\theta_d)p(w_i|\theta_d) + p(\theta_B)p(w_i|\theta_B)]^{c(w_i,d)}$$

- We can then use MLE to estimate our parameters. But how can we do that over the mixture model? **Expectation maximization**.

Expectation maximization

- Using vanilla MLE, we cannot guarantee that θ_B "specializes" on common words and θ_d on topic words instead. Without some way to connect them, the two topics would be just independent (unigram) models.
- **Expectation-Maximization (EM)** is a family of algorithms for computing the MLE of mixture models. The general idea is to iterate over inferring a $p(\theta|w)$ using a tentative estimate of parameters (expectation), and then use the inferred partitioning of words into topics to improve the estimate of parameters (maximization). We stop when we reached a local minimum.

Expectation maximization

- Introduce a hidden variable $z \in \{0, 1\}$, which indicates if we consider topic θ_d $(z = 0)$ or θ_B $(z = 1)$.

Hidden variable:
$z \in [0, 1]$

	z
the	1
paper	1
presents	1
a	1
text	0
mining	0
algorithm	0
for	1
clustering	0
...	...

Initialize $p(w|\theta_d)$ with random values.
Then iteratively improve it using E-step and M-step.
Stop when likelihood doesn't change.

$$p^{(n)}(z = 0|w) = \frac{p(\theta_d)p^{(n)}(w|\theta_d)}{p(\theta_d)p^{(n)}(w|\theta_d) + p(\theta_B)p(w|\theta_B)}$$

E-step

How likely w is from θ_d

$$p^{(n+1)}(w|\theta_d) = \frac{c(w, d)p^{(n)}(z = 0|w)}{\sum_{w' \in V} c(w', d)p^{(n)}(z = 0|w')}$$

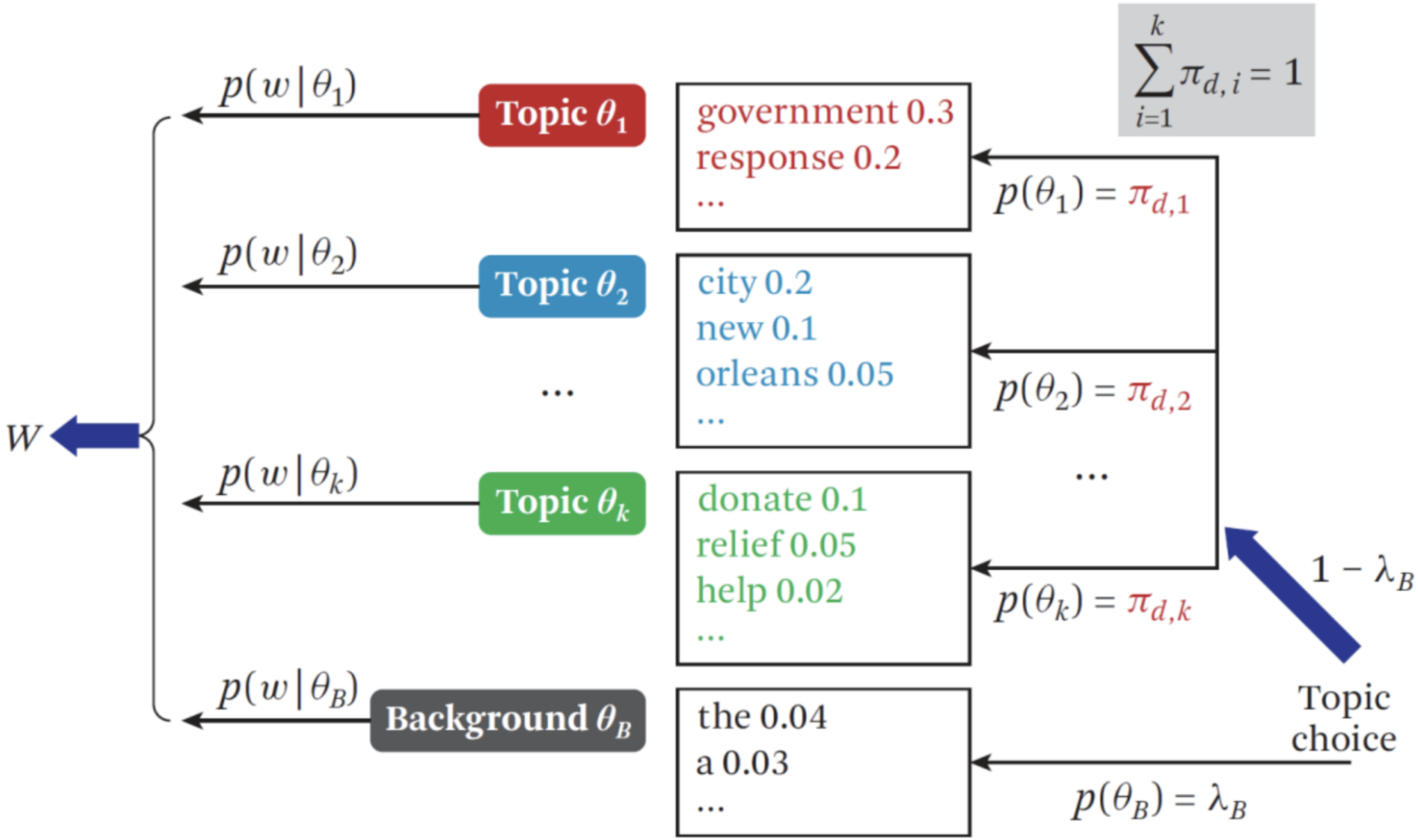
M-step

- We have seen a simple version of this before: *K-means*!

Probabilistic Latent Semantic Analysis (PLSA)

- PLSA is a generalization of the simple two-component mixture model to more than two components.
- The model includes $k + 1$ component (unigram) language models:
  - k component models, each representing a distinct topic;
  - a background model θ_B.

PLSA


$\sum_{i=1}^{k} \pi_{d,i} = 1$
$p(w|\theta_1)$
Topic θ_1
government 0.3
response 0.2
...
$p(\theta_1) = \pi_{d,1}$
$p(w|\theta_2)$
Topic θ_2
city 0.2
new 0.1
orleans 0.05
...
$p(\theta_2) = \pi_{d,2}$
...
W
$p(w|\theta_k)$
Topic θ_k
donate 0.1
relief 0.05
help 0.02
...
$p(\theta_k) = \pi_{d,k}$
$1 - \lambda_B$
$p(w|\theta_B)$
Background θ_B
the 0.04
a 0.03
...
Topic choice
$p(\theta_B) = \lambda_B$

PLSA – Likelihood

Percentage of background words (known)

Background LM (known)

Coverage of topic θ_j in doc d

Probability of word w in topic θ_j

$$p_d(w) = \lambda_B p(w|\theta_B) + (1-\lambda_B)\sum_{j=1}^{k} \pi_{d,j} p(w|\theta_j)$$

$$\log p(d) = \sum_{w \in V} c(w,d) \log [\lambda_B p(w|\theta_B) + (1-\lambda_B)\sum_{j=1}^{k} \pi_{d,j} p(w|\theta_j)]$$

} likelihood function for the document

$$\log p(C|\Lambda) = \sum_{d \in C} \sum_{w \in V} c(w,d) \log [\lambda_B p(w|\theta_B) + (1-\lambda_B)\sum_{j=1}^{k} \pi_{d,j} p(w|\theta_j)]$$

} Likelihood function for a collection of texts

Unknown parameters: $\Lambda = (\{\pi_{d,j}\}, \{\theta_j\}), j = 1, \ldots, k$

PLSA – Expectation

Hidden variable (= topic indicator): $z_{d,w} \in \{B, 1, 2, ..., k\}$

Probability that w in doc d is generated from topic θ_j

Use of Bayes Rule

$$p(z_{d,w} = j) = \frac{\pi_{d,j}^{(n)} p^{(n)}(w|\theta_j)}{\sum_{j'=1}^{k} \pi_{d,j'}^{(n)} p^{(n)}(w|\theta_{j'})}$$

$$p(z_{d,w} = B) = \frac{\lambda_B p(w|\theta_B)}{\lambda_B p(w|\theta_B) + (1-\lambda_B)\sum_{j=1}^{k} \pi_{d,j}^{(n)} p^{(n)}(w|\theta_j)}$$

Probability that w in doc d is generated from background θ_B

PLSA – Maximization

Hidden variable (= topic indicator): $z_{d,w} \in \{B, 1, 2, ..., k\}$

Re-estimated probability of doc d covering topic θ_j

ML estimate based on "allocated" word counts to topic θ_j

$$\pi_{d,j}^{(n+1)} = \frac{\sum_{w \in V} c(w,d)(1 - p(z_{d,w} = B))\, p(z_{d,w} = j)}{\sum_{j'} \sum_{w \in V} c(w,d)(1 - p(z_{d,w} = B))\, p(z_{d,w} = j')}$$

$$p^{(n+1)}(w|\theta_j) = \frac{\sum_{d \in C} c(w,d)(1 - p(z_{d,w} = B))\, p(z_{d,w} = j)}{\sum_{w' \in V} \sum_{d \in C} c(w',d)(1 - p(z_{d,w'} = B))\, p(z_{d,w'} = j)}$$

Re-estimated probability of word w for topic θ_j

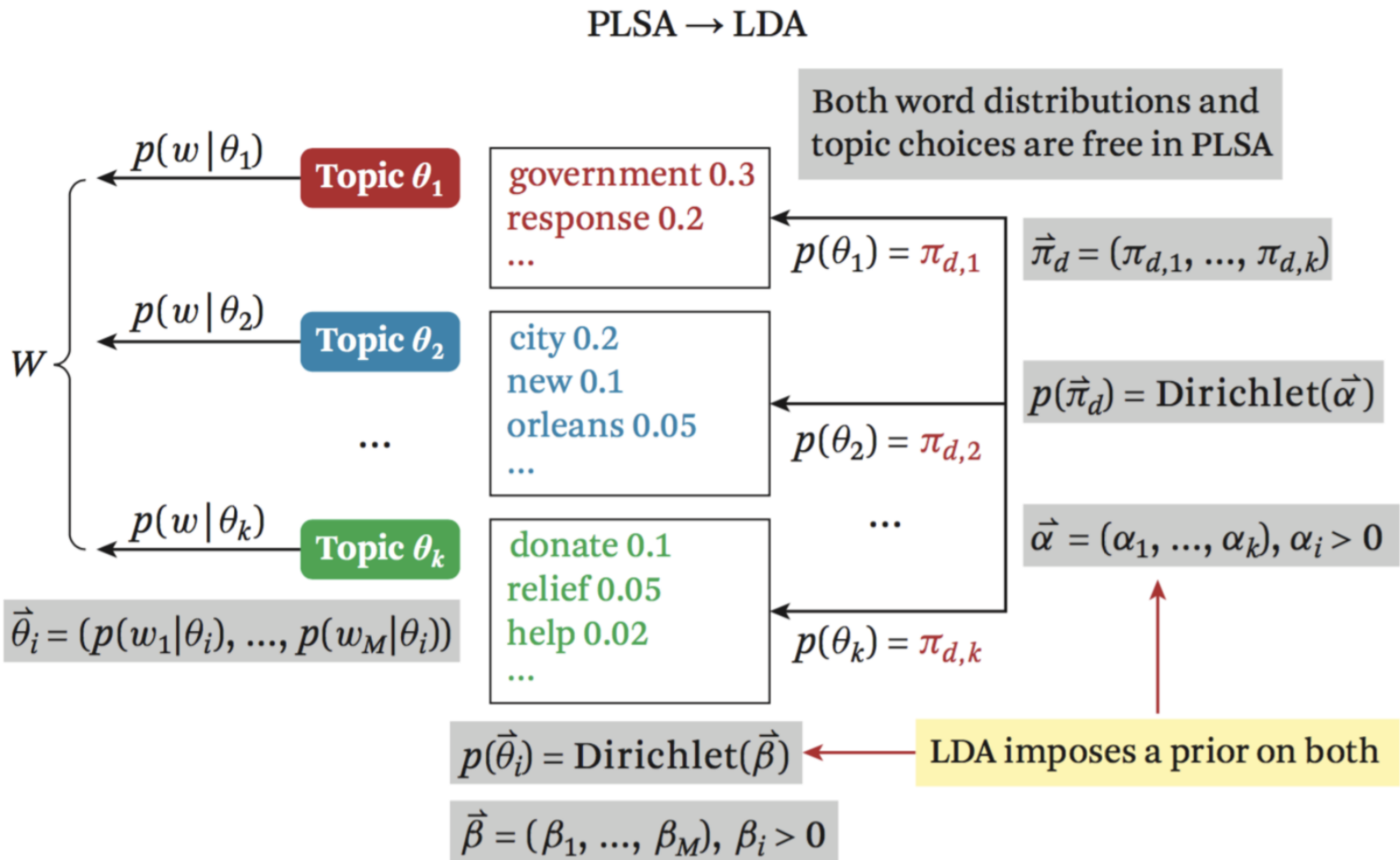
Latent Dirichlet Allocation (LDA)

- PLSA is a generative model for modeling words in a given document, but t is not a generative model for documents, because it cannot give a probability of a new document:
  - topic coverage in PLSA is tied to an observed document;
  - the model does not specify the coverage of topics in a new document.
- PLSA also contains a large number of parameters that grows linearly with the number of documents (this may lead to overfitting).
- Solution: **add priors on the parameters of PLSA and make a Bayesian version of the model**: this is **LDA** (and many variants which came afterwards).

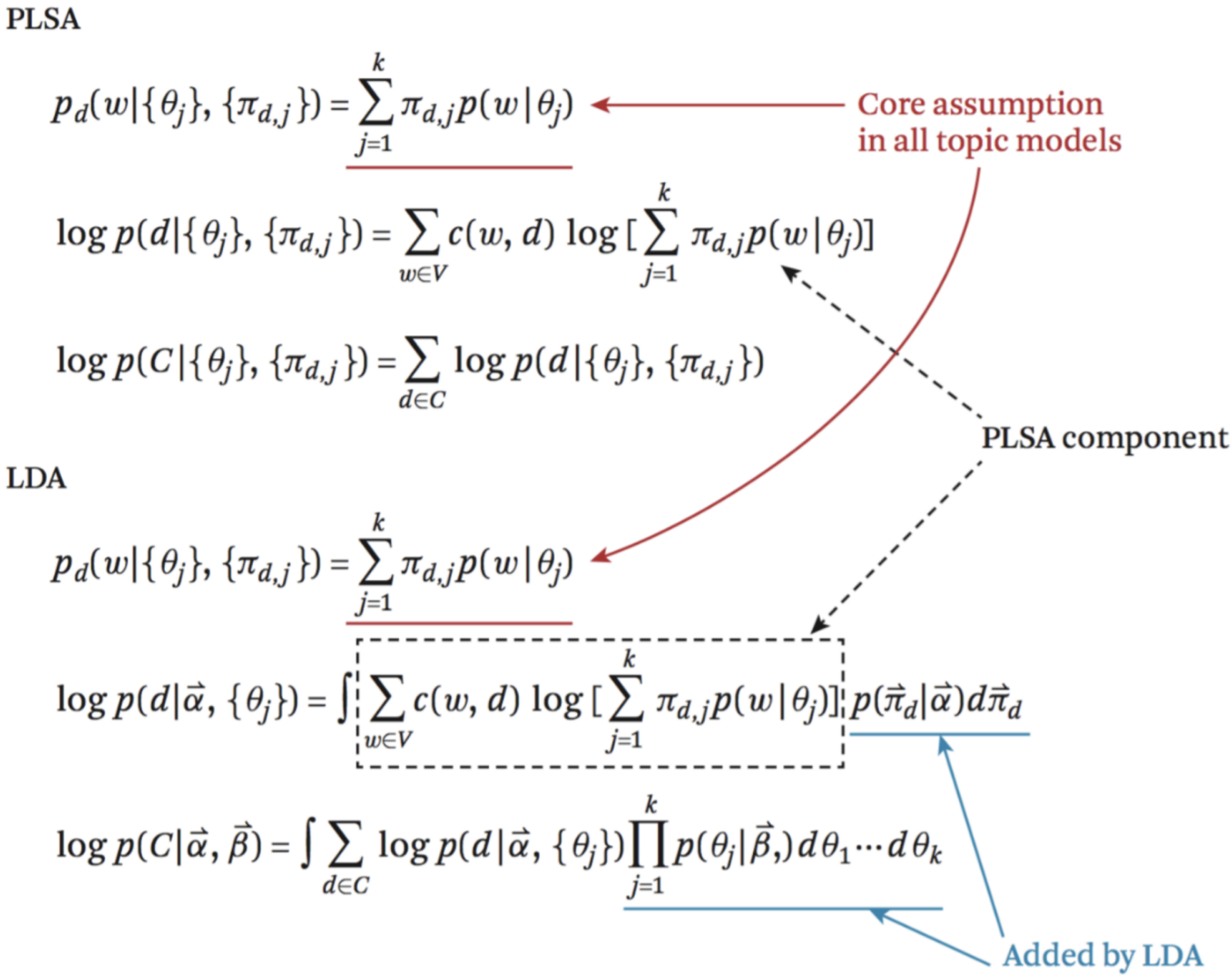
Latent Dirichlet Allocation (LDA)

- The *topic coverage* distribution for each document is assumed to be drawn from a **Dirichlet prior** (α in the next slide). Dirichlet distributions are used in Bayesian statistics because they are the conjugate priors to categorical and multinomial distributions.
- All *word distributions* representing latent topics in a collection of texts are assumed to be drawn from another Dirichlet distribution (β in the next slide).
- Once the parameters needed to characterise these two distributions are fixed, the behavior of the whole generative model is fixed as well. We simply need to sample the word distributions in the corpus and the topic coverage in a document. In PLSA, topic and word distributions are unknown model parameters.

From PLSA to LDA



From PLSA to LDA



Latent Dirichlet Allocation (LDA)

- LDA has $k + M$ parameters:
  - the Dirichlet distribution governing the topic word distributions has M parameters, $\beta_1, \ldots, \beta_M$;
  - the Dirichlet distribution governing the topic coverage has k parameters, $\alpha_1, \ldots, \alpha_k$.
- MLE can be used to estimate these parameters, however the output of interest (the topic coverage of each document and word distributions in the topics) are not immediately available.
- To obtain values for these latent variables we use posterior inference:

$$p(\theta_i, \pi_{d,j}|C, \alpha, \beta) = \frac{p(C|\theta_i, \pi_{d,j})p(\theta_i, \pi_{d,j}|\alpha, \beta)}{p(C|\alpha, \beta)}$$

- This gives us a posterior distribution over all the possible values of these variables of interest (that can be used to obtain point estimates). Inference algorithms (e.g., collapsed Gibbs Sampling) can be used to compute the conditional distribution of topic structure given the observed document.

References

1. You can find fast implementations of LDA and more in Gensim.
2. On clustering and the EM algorithm, see E, ch. 5.
3. C. X. Zhai and S. Massung. 2016. *Text Data Management and Analysis. A Practical Introduction to Information Retrieval and Text Mining*. Association for Computing Machinery and Morgan & Claypool Publishers, ch. 17 (pages 329–387 are the source for some of the topic modelling slides and figures here).
4. Original LDA paper: `http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf`
5. More friendly summary of probabilistic topic models `https://dl.acm.org/citation.cfm?id=2133826`.