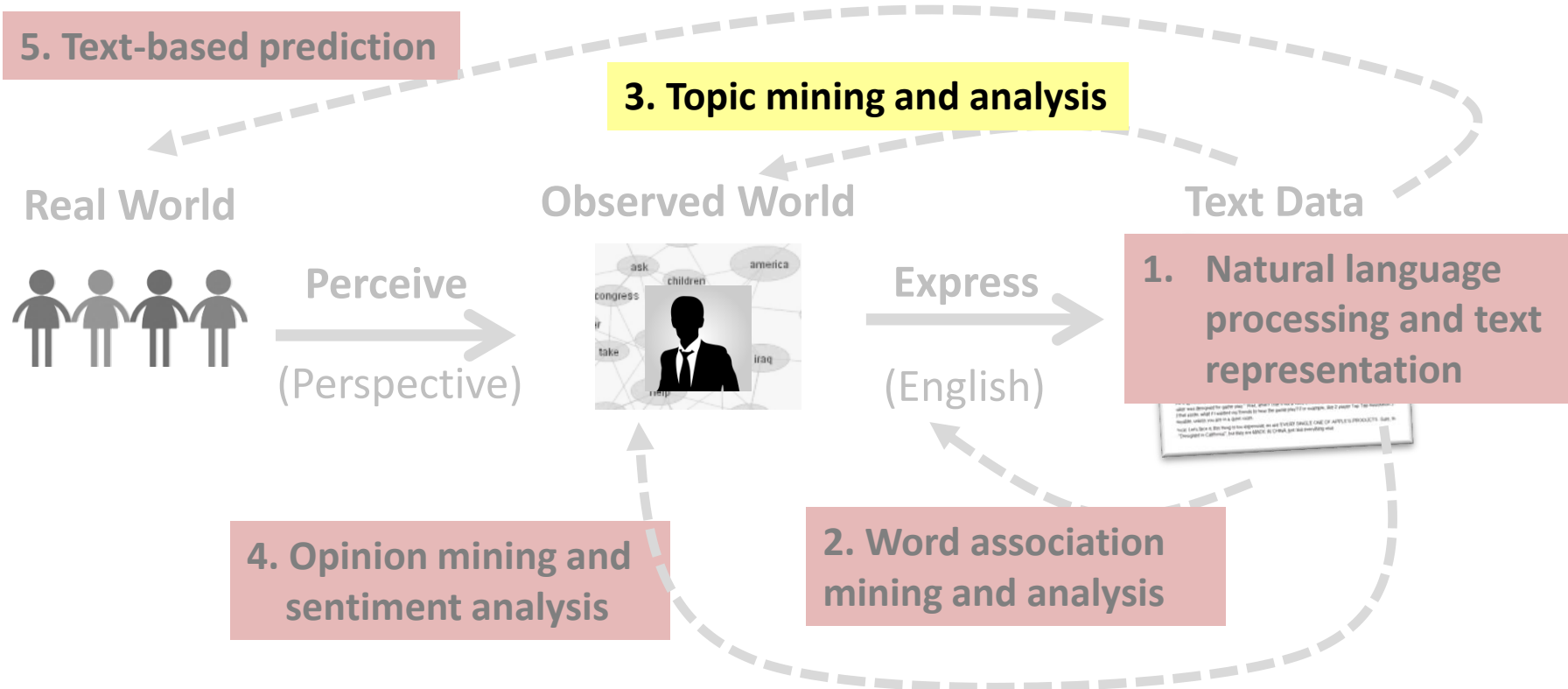
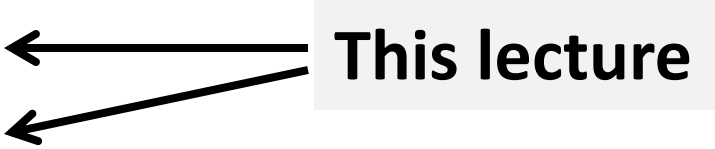


# Text Clustering: Motivation

# Text Clustering: Motivation

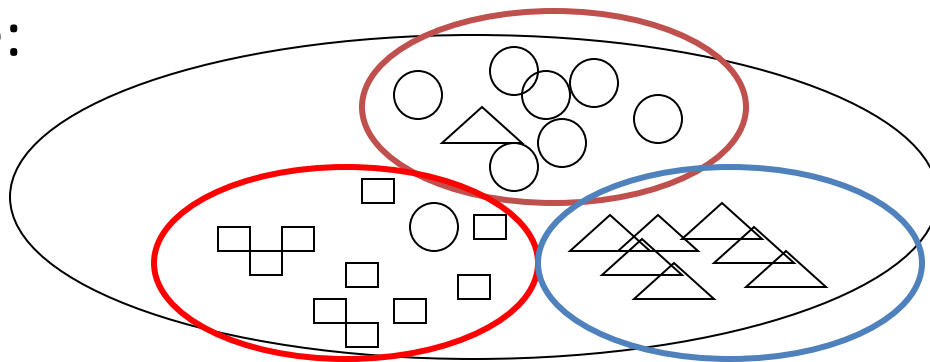


# Overview

- What is text clustering?
  - Why text clustering?
  - How to do text clustering?
    - Generative probabilistic models
    - Other approaches
  - How to evaluate clustering results?
- 
- This lecture**

# What Is Text Clustering?

- Discover “natural structure”
- Group similar objects together
- Objects can be documents, terms, passages, websites,...
- Example:



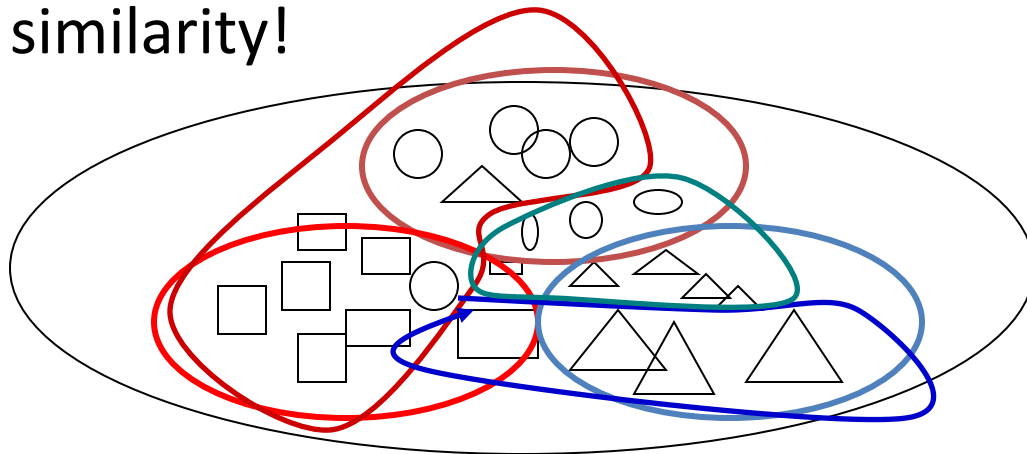
**Not well defined!**

**What does “similar” mean?**

# The “Clustering Bias”

- Any two objects can be similar, depending on how you look at them!
- Are “car” and “horse” similar?
- A user must define the **perspective** (i.e., a “**bias**”) for assessing similarity!

Basis for evaluation



# Examples of Text Clustering

- Clustering of documents in the whole collection
- Term clustering to define “concept”/“theme”/“topic”
- Clustering of passages/sentences or any selected text segments from larger text objects (e.g., all text segments about a topic discovered using a topic model)
- Clustering of websites (text object has multiple documents)
- Text clusters can be further clustered to generate a hierarchy

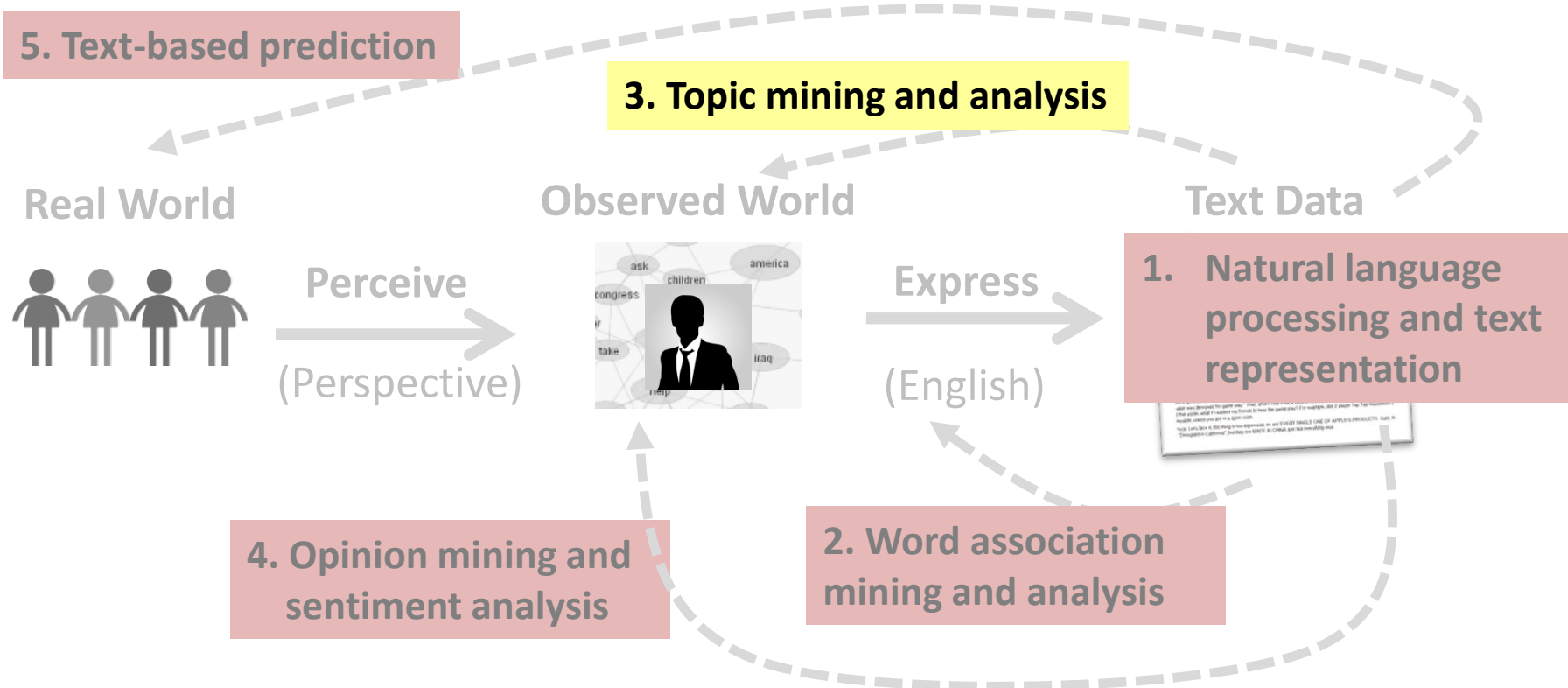
# Why Text Clustering?

- In general, very useful for text mining and exploratory text analysis:
  - ➔ Get a sense about the overall content of a collection (e.g., what are some of the “typical”/representative documents in a collection?)
  - ➔ Link (similar) text objects (e.g., removing duplicated content)
  - ➔ Create a structure on the text data (e.g., for browsing)
  - ➔ As a way to induce additional features (i.e., clusters) for classification of text objects
- Examples of applications
  - Clustering of search results
  - Understanding major complaints in emails from customers

# Text Clustering: Generative Probabilistic Models (Part 1)



# Text Clustering: Generative Probabilistic Models (Part 1)



# Overview

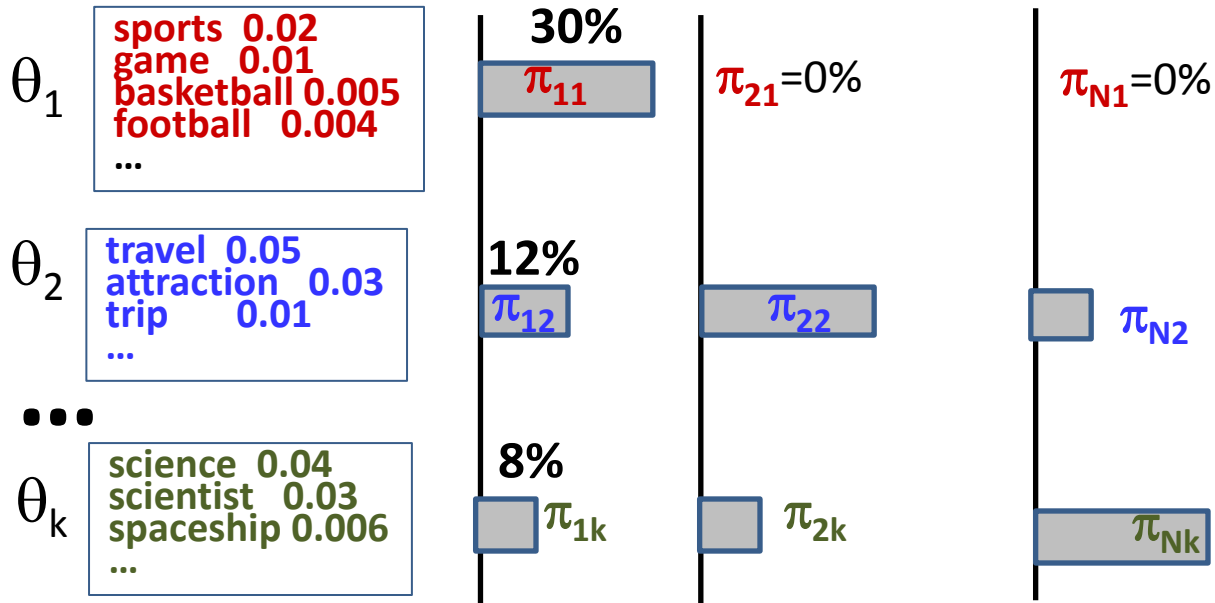
- What is text clustering?
- Why text clustering?
- How to do text clustering?
  - **Generative probabilistic models**
  - Similarity-based approaches
- How to evaluate clustering results?

# Topic Mining Revisited

INPUT:  $C, k, V$

OUTPUT:  $\{ \theta_1, \dots, \theta_k \}, \{ \pi_{i1}, \dots, \pi_{ik} \}$

Text Data



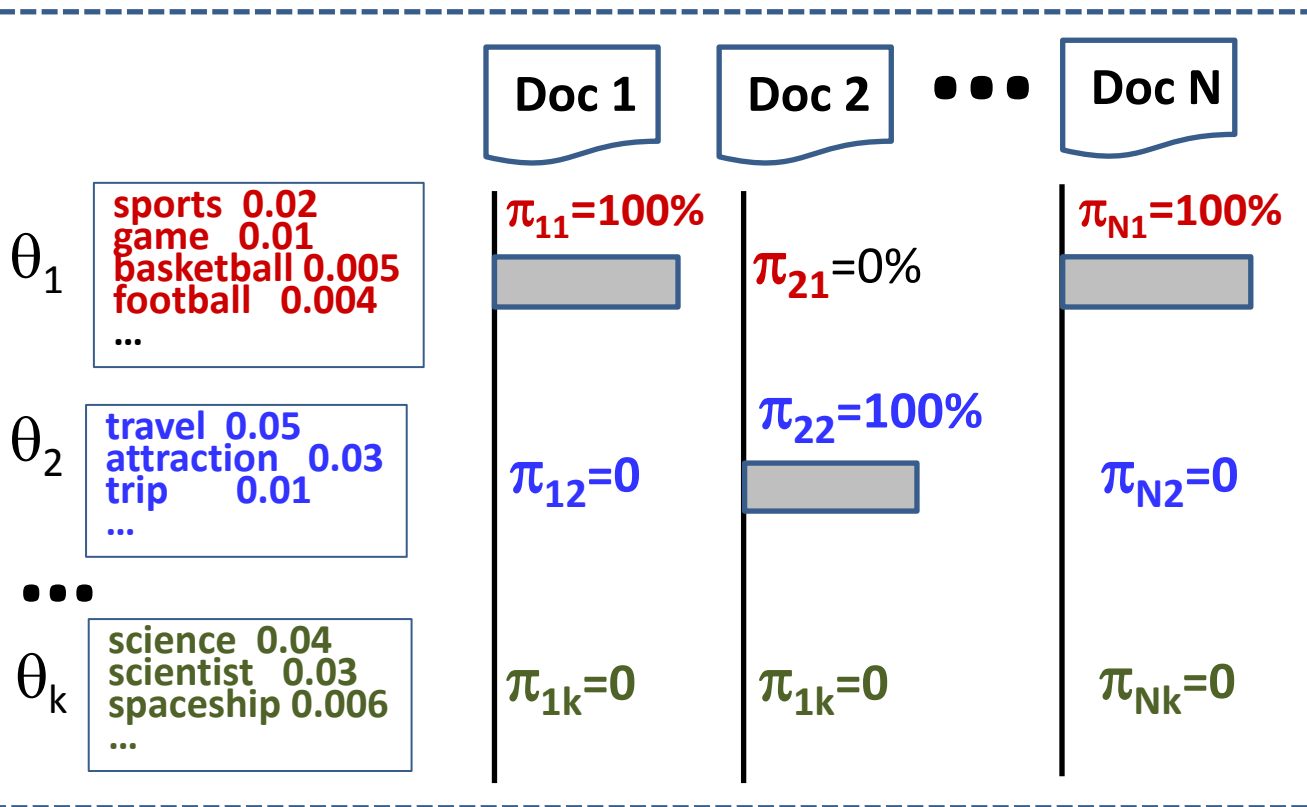
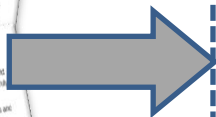
# One Topic(=cluster) Per Document

INPUT:  $C, k, V$

OUTPUT:  $\{ \theta_1, \dots, \theta_k \},$

$\{ c_1, \dots, c_N \} c_i \in [1, k]$

Text Data



# Mining One Topic Revisited

INPUT:  $C=\{d\}$ ,  $V$

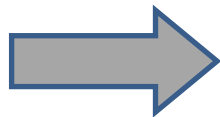
OUTPUT:  $\{\theta\}$

Text Data

any, we can know the good things in it or the bad things in it, and we can know the good things in it or the bad things in it. I am still considering taking it back. Here's why:

Speaker quality is ABSOLUTELY HORRENDOUS. The speaker is simply not functional, unless you are in perhaps recording studio. When you turn it up all the way (because you can't hear it) it becomes fuzzy, sounding like a blown speaker. What is this thing, the size of a fist? And if there is ANY background noise, you cannot hear it at all with the tune up (in case you're wondering, no, the speaker did not fail I exchanged the iPad, and the second one sounds the same, really badly. I can't share videos, share music, share anything, unless my friends and I are in a wine cellar.

I heard someone for the speaker's performance with it, including 50% a small device? It's a shame, because it's not the one's name of the speaker of the speaker.



$P(w|\theta)$

Doc d

100%

text ?  
mining ?  
association ?  
database ?

$\theta$

(1 Doc, 1 Topic)

→ (N Docs, N Topics)

$k < N$

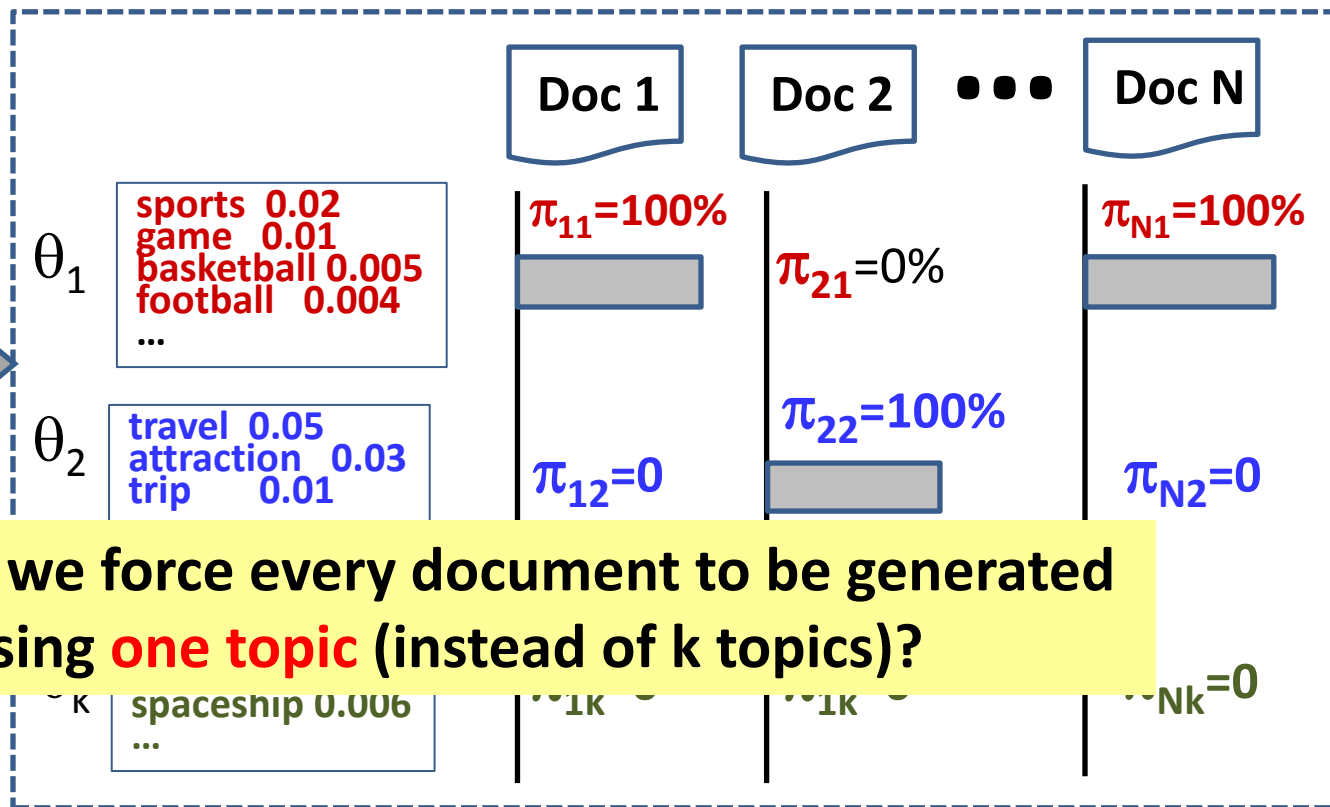
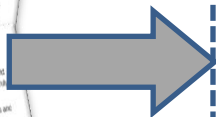
→ (N Docs, k Shared Topics)=Clustering!

# What Generative Model Can Do Clustering?

INPUT:  $C, k, V$

OUTPUT:  $\{\theta_1, \dots, \theta_k\}, \{c_1, \dots, c_N\} c_i \in [1, k]$

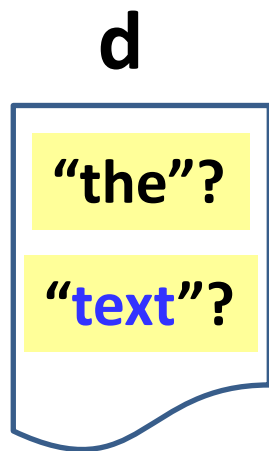
Text Data



How can we force every document to be generated using **one topic** (instead of  $k$  topics)?

# Generative Topic Model Revisited

Why can't this model be used for clustering?



$P(w | \theta_1)$

$P(w | \theta_2)$

$\theta_1$

t 0.04  
mining 0.035  
association 0.03  
clustering 0.005  
...  
the 0.000001

$\theta_2$

the 0.03  
a 0.02  
is 0.015  
we 0.01  
food 0.003  
...  
text 0.000006

$$p(\theta_1) + p(\theta_2) = 1$$

$$P(\theta_1) = 0.5$$

$$P(\theta_2) = 0.5$$

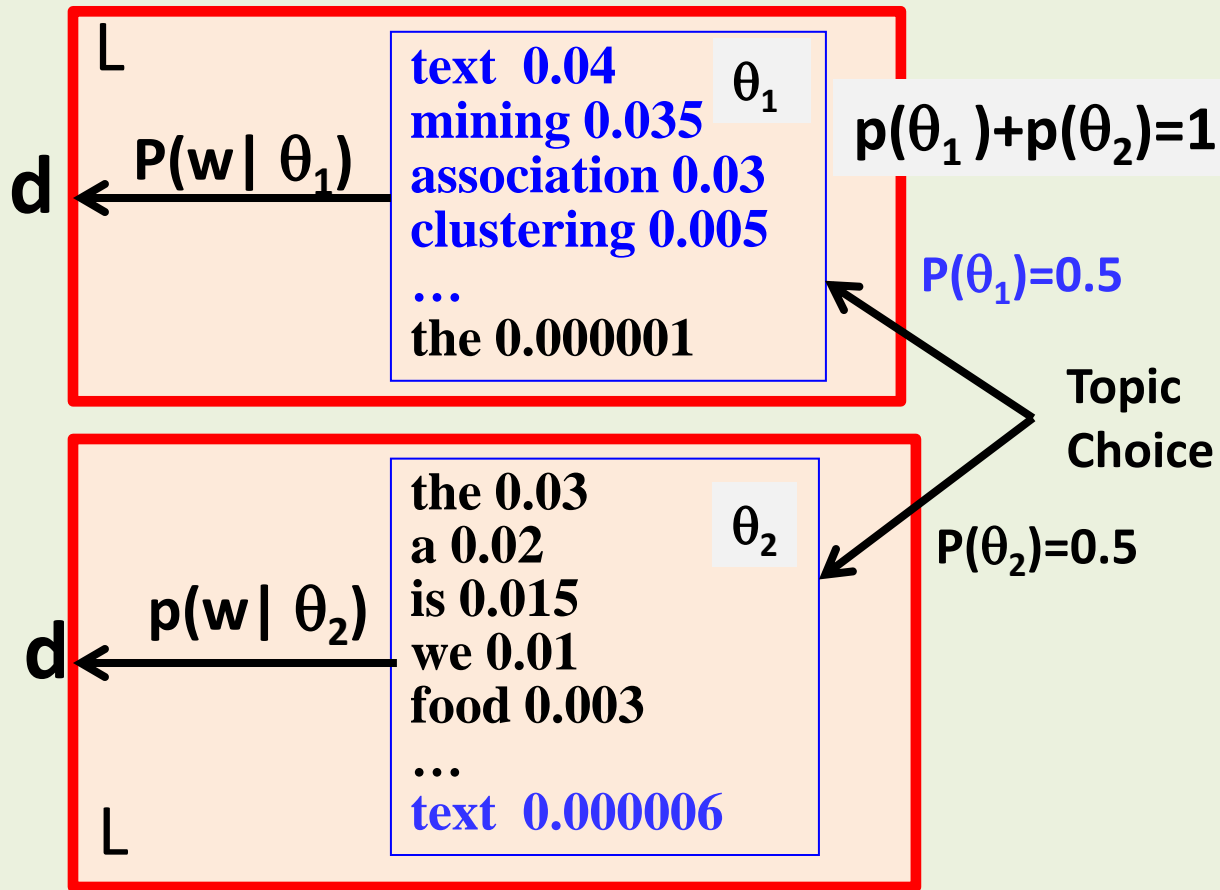
Topic  
Choice

# Mixture Model for Document Clustering

Difference from  
topic model?

$d = x_1 x_2 \dots x_L$

What if  $P(\theta_1)=1$   
or  $P(\theta_2)=1$ ?





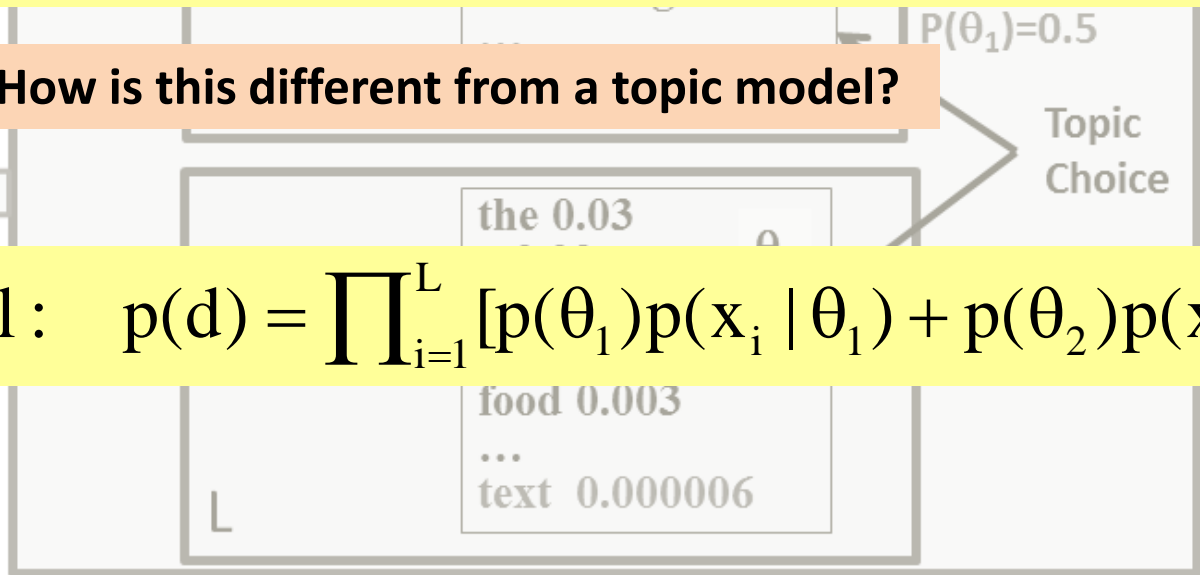
# Likelihood Function: $p(d)=?$

$$\begin{aligned} p(d) &= p(\theta_1)p(d | \theta_1) + p(\theta_2)p(d | \theta_2) \\ &= p(\theta_1)\prod_{i=1}^L p(x_i | \theta_1) + p(\theta_2)\prod_{i=1}^L p(x_i | \theta_2) \end{aligned}$$

$d=x_1 x_2 \dots x_L$

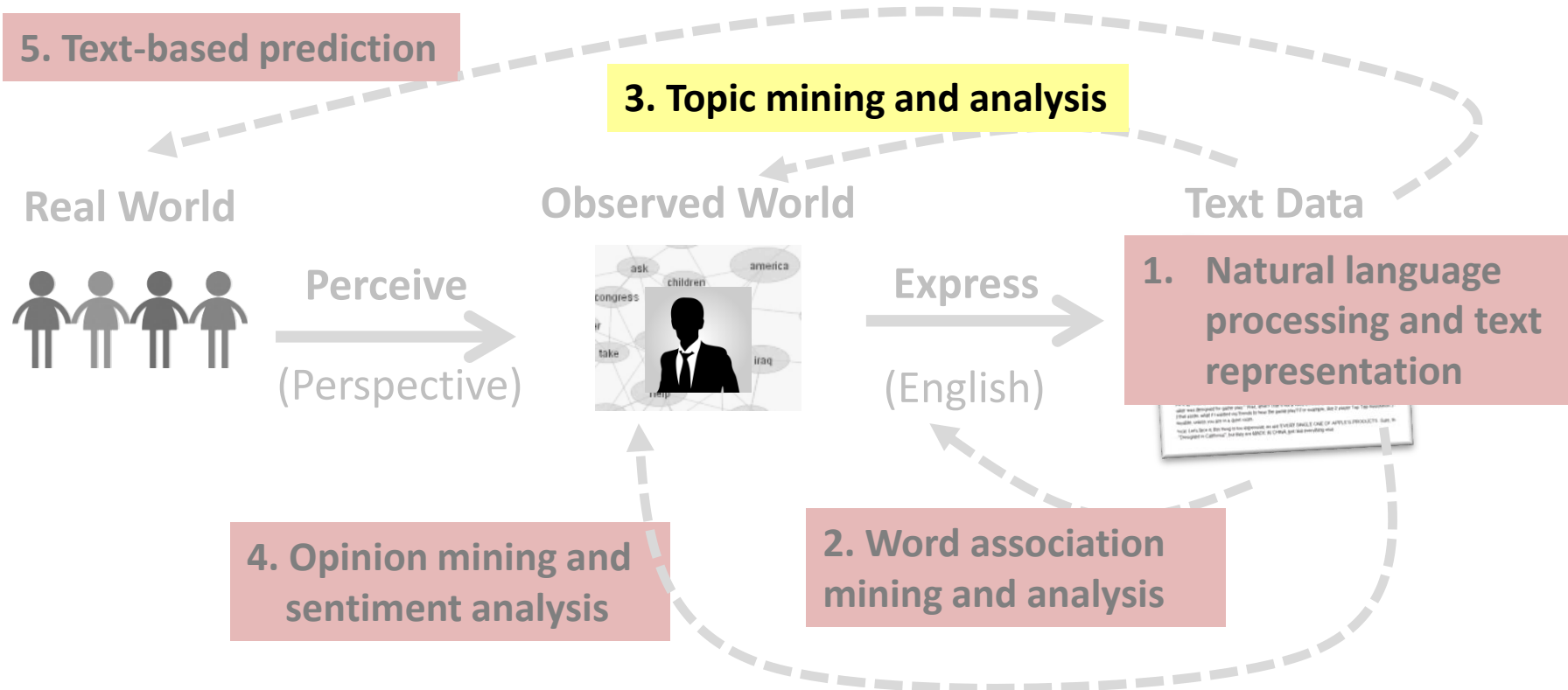
How is this different from a topic model?

topic model:  $p(d) = \prod_{i=1}^L [p(\theta_1)p(x_i | \theta_1) + p(\theta_2)p(x_i | \theta_2)]$



# Text Clustering: Generative Probabilistic Models (Part 2)

# Text Clustering: Generative Probabilistic Models (Part 2)



# Likelihood Function: $p(d)=?$

$$\begin{aligned} p(d) &= p(\theta_1)p(d | \theta_1) + p(\theta_2)p(d | \theta_2) \\ &= p(\theta_1)\prod_{i=1}^L p(x_i | \theta_1) + p(\theta_2)\prod_{i=1}^L p(x_i | \theta_2) \end{aligned}$$

$d = x_1 x_2 \dots x_L$

the 0.000001

the 0.03

Topic  
Choice

**How can we generalize it to include  $k$  topics/clusters?**

$d \leftarrow$

we 0.01  
food 0.003  
...  
text 0.000006

$L$

# Mixture Model for Document Clustering

- Data: a collection of documents  $C=\{d_1, \dots, d_N\}$
- Model: mixture of  $k$  unigram LMs:  $\Lambda=(\{\theta_i\}; \{p(\theta_i)\})$ ,  $i \in [1, k]$ 
  - To generate a document, first **choose a**  $\theta_i$  according to  $p(\theta_i)$ , and then generate **all** words in the document using  $p(w | \theta_i)$
- Likelihood:

$$\begin{aligned} p(d | \Lambda) &= \sum_{i=1}^k [p(\theta_i) \prod_{j=1}^{|d|} p(x_j | \theta_i)] \\ &= \sum_{i=1}^k [p(\theta_i) \prod_{w \in V} p(w | \theta_i)^{c(w, d)}] \end{aligned}$$

- Maximum Likelihood estimate

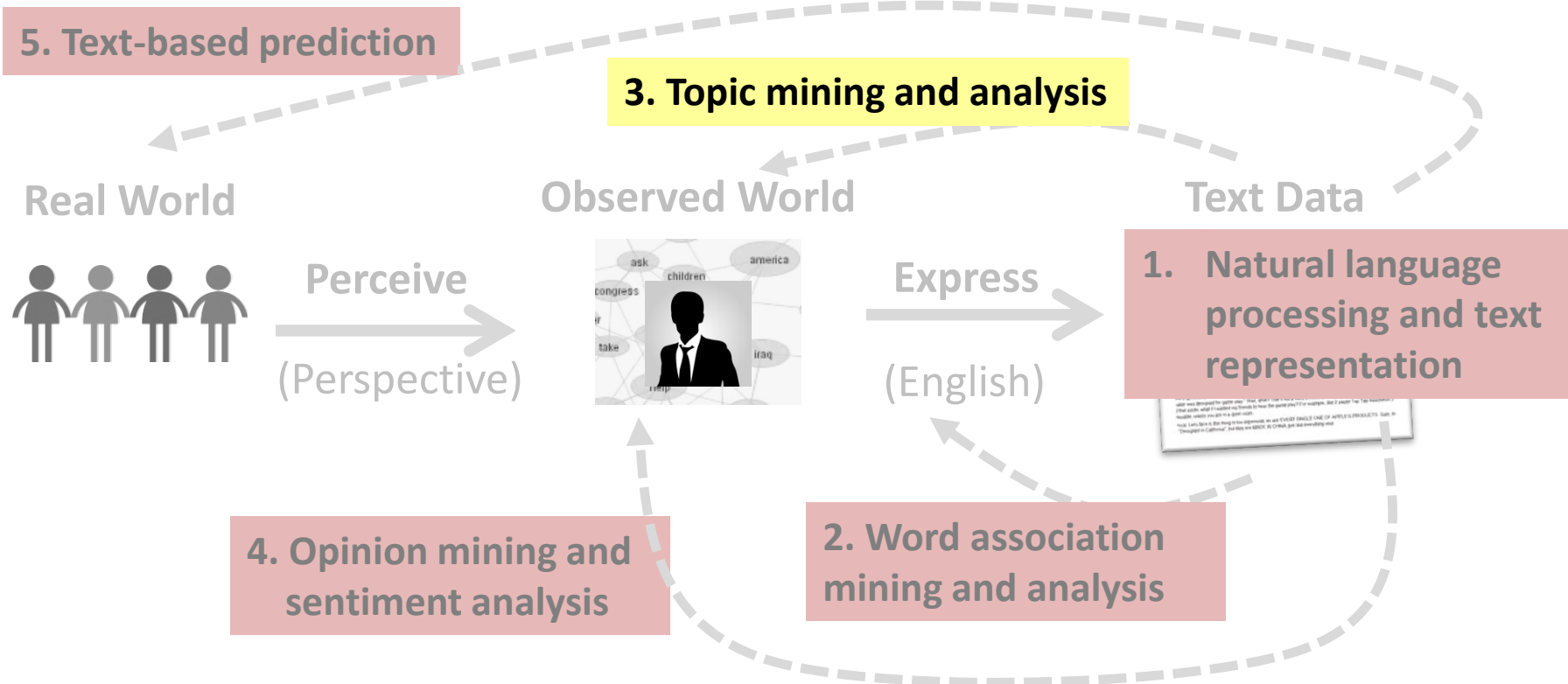
$$\Lambda^* = \arg \max_{\Lambda} p(d | \Lambda)$$

# Cluster Allocation After Parameter Estimation

- **Parameters** of the mixture model:  $\Lambda = (\{\theta_i\}; \{p(\theta_i)\})$ ,  $i \in [1, k]$ 
  - Each  $\theta_i$  represents the **content of cluster i** :  $p(w | \theta_i)$
  - $p(\theta_i)$  indicates the **size of cluster i**
  - Note that unlike in PLSA,  $p(\theta_i)$  doesn't depend on  $d$ !
- Which cluster should document  $d$  belong to?  $c_d = ?$ 
  - **Likelihood only**: Assign  $d$  to the cluster corresponding to the topic  $\theta_i$  that most likely has been used to generate  $d$ 
$$c_d = \arg \max_i p(d | \theta_i)$$
  - **Likelihood + prior  $p(\theta_i)$  (Bayesian)**: favor large clusters
$$c_d = \arg \max_i p(d | \theta_i) p(\theta_i)$$

# Text Clustering: Generative Probabilistic Models (Part 3)

# Text Clustering: Generative Probabilistic Models (Part 3)





# How Can We Compute the ML Estimate?

- Data: a collection of documents  $C=\{d_1, \dots, d_N\}$
- Model: mixture of  $k$  unigram LMs:  $\Lambda=(\{\theta_i\}; \{p(\theta_i)\})$ ,  $i \in [1, k]$ 
  - To generate a document, first **choose a**  $\theta_i$  according to  $p(\theta_i)$  and then generate **all** words in the document using  $p(w | \theta_i)$

- Likelihood:

$$p(d | \Lambda) = \sum_{i=1}^k [p(\theta_i) \prod_{w \in V} p(w | \theta_i)^{c(w, d)}]$$

$$p(C | \Lambda) = \prod_{j=1}^N p(d_j | \Lambda)$$

- Maximum Likelihood estimate

$$\Lambda^* = \arg \max_{\Lambda} p(C | \Lambda)$$

# EM Algorithm for Document Clustering

- Initialization: Randomly set  $\Lambda = (\{\theta_i\}; \{p(\theta_i)\})$ ,  $i \in [1, k]$
- **Repeat until likelihood  $p(C | \Lambda)$  converges**
  - **E-Step: Infer which distribution has been used to generate document  $d$ : hidden variable  $Z_d \in [1, k]$**

$$p^{(n)}(Z_d = i | d) \propto p^{(n)}(\theta_i) \prod_{w \in V} p^{(n)}(w | \theta_i)^{c(w, d)}$$

$$\sum_{i=1}^k p^{(n)}(Z_d = i | d) = 1$$

- **M-Step: Re-estimation of all parameters**

$$p^{(n+1)}(\theta_i) \propto \sum_{j=1}^N p^{(n)}(Z_{d_j} = i | d_j)$$

$$\sum_{i=1}^k p^{(n+1)}(\theta_i) = 1$$

$$p^{(n+1)}(w | \theta_i) \propto \sum_{j=1}^N c(w, d_j) p^{(n)}(Z_{d_j} = i | d_j)$$

$$\sum_{w \in V} p^{(n+1)}(w | \theta_i) = 1, \quad \forall i \in [1, k]$$

# An Example of 2 Clusters

Random Initialization

$$p(\theta_1) = p(\theta_2) = 0.5$$

	$p(w \theta_1)$	$p(w \theta_2)$
text	0.5	0.1
mining	0.2	0.1
medical	0.2	0.75
health	0.1	0.05

E-step

Document d

Hidden variables:

$$Z_d \in \{1, 2\}$$

	$c(w,d)$
text	2
mining	2
medical	0
health	0

$$\begin{aligned}
 p(Z_d = 1 | d) &= \frac{p(\theta_1) p(\text{"text"} | \theta_1)^2 p(\text{"mining"} | \theta_1)^2}{p(\theta_1) p(\text{"text"} | \theta_1)^2 p(\text{"mining"} | \theta_1)^2 + p(\theta_2) p(\text{"text"} | \theta_2)^2 p(\text{"mining"} | \theta_2)^2} \\
 &= \frac{0.5 * 0.5^2 * 0.2^2}{0.5 * 0.5^2 * 0.2^2 + 0.5 * 0.1^2 * 0.1^2} = \frac{100}{101}
 \end{aligned}$$

$$p(Z_d = 2 | d) = ?$$

# Normalization to Avoid Underflow

	$p(w \theta_1)$	$p(w \theta_2)$	$p(w \bar{\theta})$
text	0.5	0.1	$(0.5+0.1)/2$
mining	0.2	0.1	$(0.2+0.1)/2$
medical	0.2	0.75	$(0.2+0.75)/2$
health	0.1	0.05	$(0.1+0.05)/2$

**Average of  $p(w|\theta_i)$   
as a possible normalizer**

$$p(Z_d = 1 | d) = \frac{p(\theta_1)p(\text{"text"}|\theta_1)^2p(\text{"mining"}|\theta_1)^2}{p(\text{"text"}|\bar{\theta})^2p(\text{"mining"}|\bar{\theta})^2} \cdot \frac{p(\theta_2)p(\text{"text"}|\theta_2)^2p(\text{"mining"}|\theta_2)^2}{p(\text{"text"}|\bar{\theta})^2p(\text{"mining"}|\bar{\theta})^2} + \frac{p(\theta_1)p(\text{"text"}|\theta_1)^2p(\text{"mining"}|\theta_1)^2}{p(\text{"text"}|\bar{\theta})^2p(\text{"mining"}|\bar{\theta})^2} + \frac{p(\theta_2)p(\text{"text"}|\theta_2)^2p(\text{"mining"}|\theta_2)^2}{p(\text{"text"}|\bar{\theta})^2p(\text{"mining"}|\bar{\theta})^2}$$

# An Example of 2 Clusters (cont.)

## From E-Step

	$P(Z_d=1   d)$		$c(\text{"text"})$	$c(\text{"mining"})$
d1	0.9	d1	2	3
d2	0.1	d2	1	2
d3	0.8	d3	4	3

## M-Step

$$p(\theta_1) = ? \quad p(\theta_2) = ?$$

$$p(\theta_1) = \frac{p(Z_{d_1}=1 | d_1) + p(Z_{d_2}=1 | d_2) + p(Z_{d_3}=1 | d_3)}{3}$$

$$= \frac{0.9 + 0.1 + 0.8}{3} = 0.6$$

	$p(w   \theta_1)$	$p(w   \theta_2)$
text	?	?
mining	?	?
medical	?	?
health	?	?

$$p(\text{"text"} | \theta_1) \propto c(\text{"text"}, d_1) * p(Z_{d_1}=1 | d_1) + \dots + c(\text{"text"}, d_3) * p(Z_{d_3}=1 | d_3)$$

$$= 2 * 0.9 + 1 * 0.1 + 4 * 0.8$$

$$p(\text{"mining"} | \theta_1) \propto 3 * 0.9 + 2 * 0.1 + 3 * 0.8$$

$$p(\text{"text"} | \theta_1) + p(\text{"mining"} | \theta_1) + p(\text{"medical"} | \theta_1) + p(\text{"health"} | \theta_1) = 1$$

# Summary of Generative Model for Clustering

- A slight variation of topic model can be used for clustering documents
  - Each **cluster** is represented by a **unigram LM**  $p(w|\theta_i)$  → **Term cluster**
  - A document is generated by first choosing a unigram LM and then generating **ALL words** in the document using this **single LM**
  - Estimated model parameters give both a topic characterization of each cluster and a probabilistic assignment of a document into each cluster
  - “Hard” clusters can be obtained by forcing a document into the cluster corresponding to the unigram LM most likely used to generate the document
- EM algorithm can be used to compute the ML estimate
  - Normalization is often needed to avoid underflow

# Text Clustering: Similarity-based Approaches

# Overview

- What is text clustering?
- Why text clustering?
- How to do text clustering?
  - Generative probabilistic models
  - **Similarity-based approaches**
- How to evaluate clustering results?



# Similarity-based Clustering: General Idea

- Explicitly define a similarity function to measure similarity between two text objects (i.e., providing “clustering bias”)
- Find an optimal partitioning of data to
  - maximize intra-group similarity and
  - minimize inter-group similarity
- Two strategies for obtaining optimal clustering
  - Progressively construct a hierarchy of clusters (hierarchical clustering)
    - Bottom-up (agglomerative): gradually group similar objects into larger clusters
    - Top-down (divisive): gradually partition the data into smaller clusters
  - Start with an initial tentative clustering and iteratively improve it (“flat” clustering, e.g., k-Means)

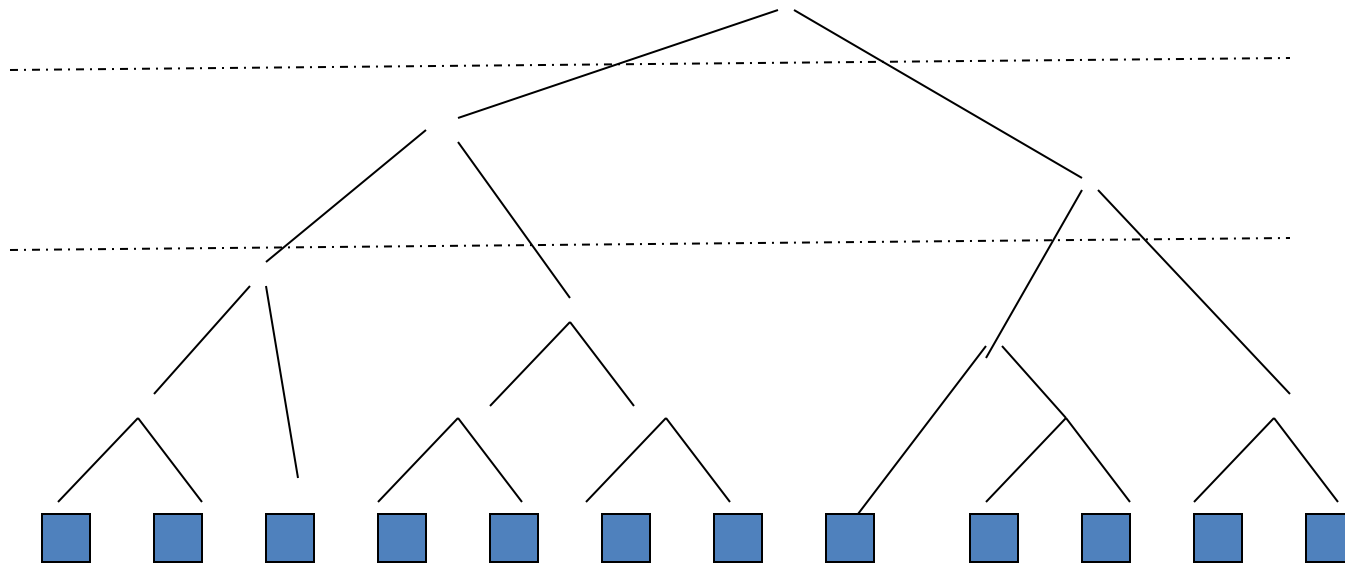
# Similarity-based Clustering Methods

- Many general clustering methods are available!
- Two representative methods
  - Hierarchical Agglomerative Clustering (HAC)
  - k-means

# Agglomerative Hierarchical Clustering

- Given a similarity function to measure similarity between two objects
- Gradually group similar objects together in a bottom-up fashion to form a hierarchy
- Stop when some stopping criterion is met
- Variations: different ways to compute group similarity based on individual object similarity

# Similarity-induced Structure



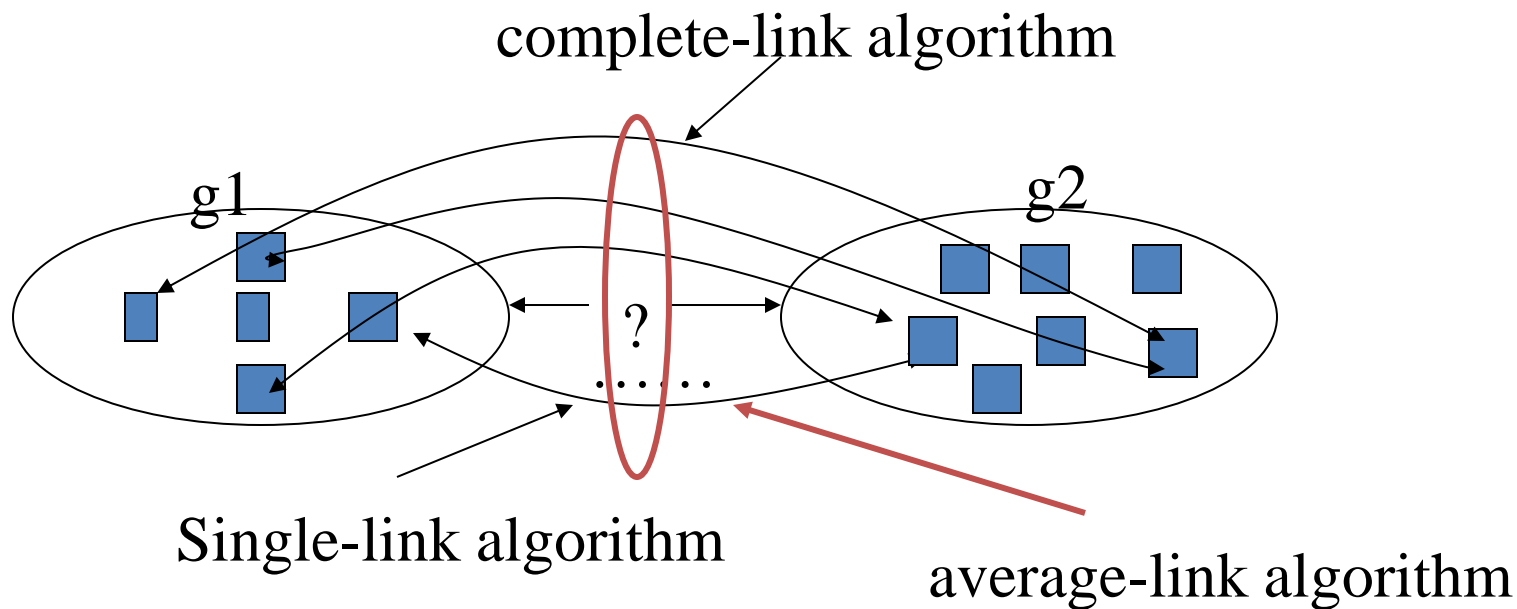
# How to Compute Group Similarity

Three popular methods:

Given two groups  $g1$  and  $g2$ ,

- Single-link algorithm:  $s(g1, g2)$  = similarity of the **closest** pair
- Complete-link algorithm:  $s(g1, g2)$  = similarity of the **farthest** pair
- Average-link algorithm:  $s(g1, g2)$  = **average** of similarity of all pairs

# Group Similarity Illustrated



# Comparison of Single-Link, Complete-Link, and Average-Link

- Single-link
  - “Loose” clusters
  - Individual decision, sensitive to outliers
- Complete-link
  - “Tight” clusters
  - Individual decision, sensitive to outliers
- Average-link
  - “In between”
  - Group decision, insensitive to outliers
- Which one is the best? It depends on what you need!

# K-Means Clustering

- Represent each text object as a term vector and assume a similarity function defined on two objects
- Start with  $k$  randomly selected vectors and assume they are the centroids of  $k$  clusters (initial tentative clustering) → **Initialization**
- Assign every vector to a cluster whose centroid is the closest to the vector      **≈ E-step    difference?**
- Re-compute the centroid for each cluster based on the newly assigned vectors in the cluster      **≈ M-step    difference?**
- Repeat this process until the similarity-based objective function (i.e., within cluster sum of squares) converges (to a local minimum)

**Very similar to clustering with EM for mixture model!**



# Summary of Clustering Methods

- Model based approaches (mixture model)
  - Uses an implicit similarity function (model → clustering bias)
  - Cluster structure is “built” into a generative model
  - Complex generative models can discover complex structures
  - Prior can be leveraged to further customize the clustering algorithm
  - However, no easy way to directly control the similarity measure
- Similarity-based approaches
  - Allows for direct and flexible specification of similarity
  - Objective function to be optimized is not always clear
- Both approaches can generate both term clusters and doc clusters

# Text Clustering: Evaluation

# Overview

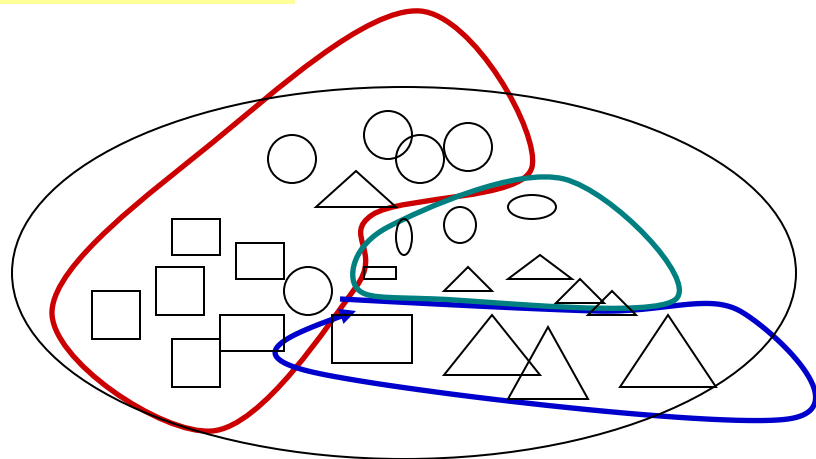
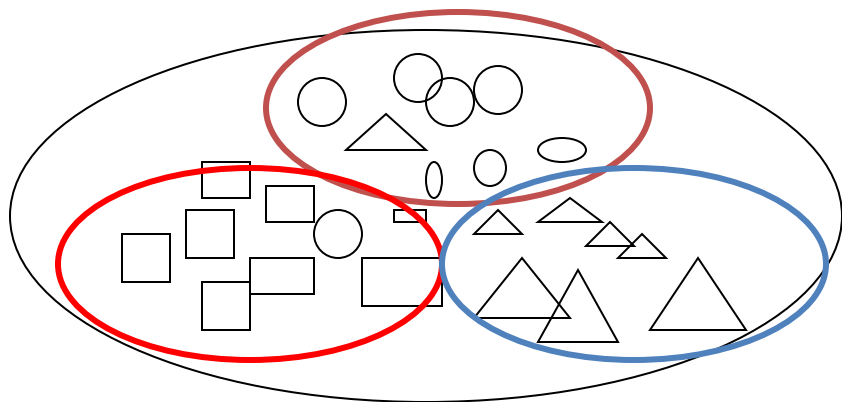
- What is text clustering?
- Why text clustering?
- How to do text clustering?
  - Generative probabilistic models
  - Similarity-based approaches
- **How to evaluate clustering results?**

# The “Clustering Bias”

- Any two objects can be similar, depending on how you look at them!
- A user must define the **perspective** (i.e., a “**bias**”) for assessing similarity!



Basis for evaluation



# Direct Evaluation of Text Clusters

- Question to answer: How close are the system-generated clusters to the ideal clusters (generated by humans)?
  - “Closeness” can be assessed from multiple perspectives
  - “Closeness” can be quantified
  - “Clustering bias” is imposed by the human assessors
- Evaluation procedure:
  - Given a test set, have humans to create an ideal clustering result (i.e., an ideal partitioning of text objects or “gold standard”)
  - Use a system to produce clusters from the same test set
  - Quantify the similarity between the system-generated clusters and the gold standard clusters
  - Similarity can be measured from multiple perspectives (e.g., purity, normalized mutual information, F measure)

# Indirect Evaluation of Text Clusters

- Question to answer: how useful are the clustering results for the intended applications?
  - “Usefulness” is inevitably application specific
  - “Clustering bias” is imposed by the intended application
- Evaluation procedure:
  - Create a test set for the intended application to quantify the performance of any system for this application
  - Choose a baseline system to compare with
  - Add a clustering algorithm to the baseline system → “clustering system”
  - Compare the performance of the clustering system and the baseline in terms of any performance measure for the application

# Summary of Text Clustering

- Text clustering is an unsupervised general text mining technique to
  - obtain an overall picture of the text content (exploring text data)
  - discover interesting clustering structures in text data
- Many approaches are possible
  - Strong clusters tend to show up no matter what method used
  - Effectiveness of a method highly depends on whether the desired clustering bias is captured appropriately (either through using the right generative model or the right similarity function)
  - Deciding the optimal number of clusters is generally a difficult problem for any method due to the unsupervised nature
- Evaluation of clustering results can be done both directly and indirectly

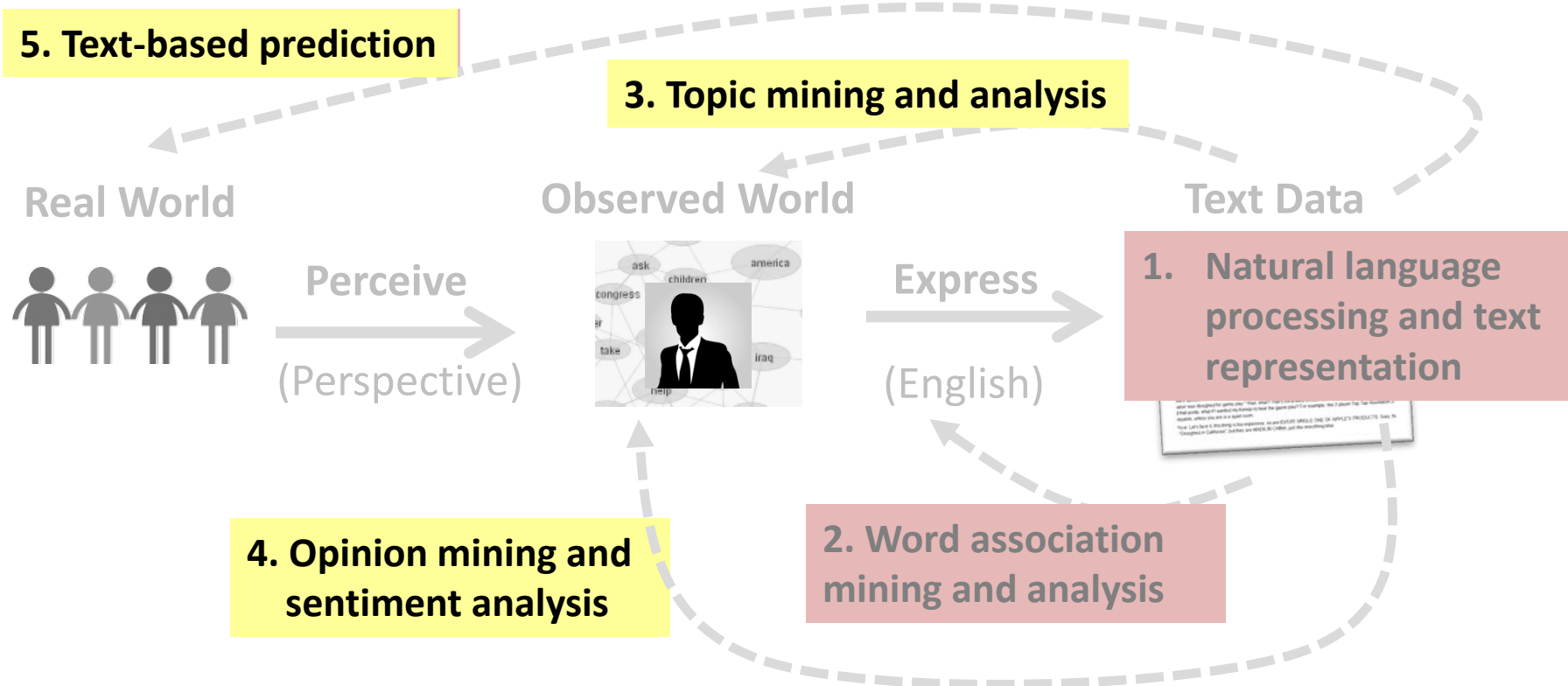
# Suggested Reading

- Manning, Chris D., Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge: Cambridge University Press, 2007. (Chapter 16)

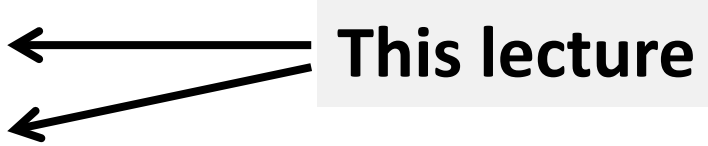


# Text Categorization: Motivation

# Text Categorization

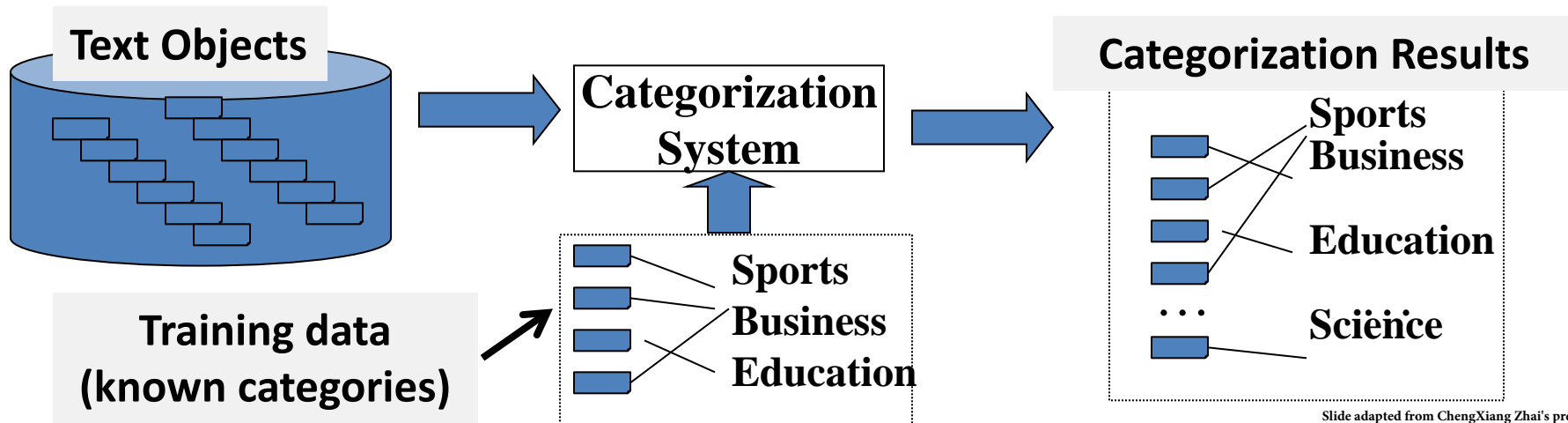


# Overview

- What is text categorization? ←
  - Why text categorization? ←
  - How to do text categorization?
    - Generative probabilistic models
    - Discriminative approaches
  - How to evaluate categorization results?
- This lecture**
- 

# Text Categorization

- Given the following:
  - A set of **predefined categories**, possibly forming a hierarchy *and often*
  - A **training set** of labeled text objects
- Task: **Classify** a text object into **one or more** of the **categories**



# Examples of Text Categorization

- **Text objects can vary** (e.g., documents, passages, or collections of text)
- **Categories can also vary**
  - “**Internal**” categories that characterize a text object (e.g., topical categories, sentiment categories)
  - “**External**” categories that characterize an entity associated with the text object (e.g., author attribution or any other meaningful categories associated with text data)
- **Some examples of applications**
  - News categorization, literature article categorization (e.g., MeSH annotations)
  - Spam email detection/filtering
  - Sentiment categorization of product reviews or tweets
  - Automatic email sorting/routing
  - Author attribution

# Variants of Problem Formulation

- **Binary** categorization: Only two categories
  - Retrieval: {relevant-doc, non-relevant-doc}
  - Spam filtering: {spam, non-spam}
  - Opinion: {positive, negative}
- **K-category** categorization: More than two categories
  - Topic categorization: {sports, science, travel, business,...}
  - Email routing: {folder1, folder2, folder3,...}
- **Hierarchical** categorization: Categories form a hierarchy
- **Joint** categorization: Multiple **related** categorization tasks done in a joint manner

Binary categorization can potentially support all other categorizations

# Why Text Categorization?

- To **enrich text representation** (more understanding of text)
  - Text can now be represented in multiple levels (keywords + categories)
  - Semantic categories assigned can be directly or indirectly useful for an application
  - Semantic categories facilitate aggregation of text content (e.g., aggregating all positive/negative opinions about a product)
- To **infer properties of entities** associated with text data (discovery of **knowledge about the world**)
  - As long as an entity can be associated with text data, we can always use the text data to help categorize the associated entities
  - E.g., discovery of non-native speakers of a language; prediction of party affiliation based on a political speech

# Text Categorization: Methods



# Overview

- What is text categorization?
- Why text categorization?
- **How to do text categorization?**
  - Generative probabilistic models
  - Discriminative approaches
- How to evaluate categorization results?

# Categorization Methods: Manual

- Determine the category based on rules that are carefully designed to reflect the domain knowledge about the categorization problem
- Works well when
  - The categories are very well defined
  - Categories are easily distinguished based on surface features in text (e.g., special vocabulary is known to only occur in a particular category)
  - Sufficient domain knowledge is available to suggest many effective rules
- Problems
  - Labor intensive → doesn't scale up well
  - Can't handle uncertainty in rules; rules may be inconsistent → not robust
- Both problems can be solved/alleviated by using machine learning

# Categorization Methods: “Automatic”

- Use **human experts** to
  - Annotate data sets with **category labels** → Training data
  - Provide a set of **features** to represent each text object that can potentially provide a “clue” about the category
- Use **machine learning** to learn “soft rules” for categorization from the training data
  - Figure out **which features are most useful** for separating different categories
  - **Optimally combine the features to minimize the errors** of categorization on the training data
  - The trained classifier can then be applied to a new text object to predict the most likely category (that a human expert would assign to it)

# Machine Learning for Text Categorization

- **General setup:** Learn a classifier  $f: X \rightarrow Y$ 
  - Input:  $X$  = all text objects; Output:  $Y$  = all categories
  - Learn a classifier function,  $f: X \rightarrow Y$ , such that  $f(x)=y \in Y$  gives the correct category for  $x \in X$  (“correct” is based on the training data)
- **All methods**
  - Rely on discriminative features of text objects to distinguish categories
  - Combine multiple features in a weighted manner
  - Adjust weights on features to minimize errors on the training data
- **Different methods** tend to vary in
  - Their way of measuring the errors on the training data (may optimize a different objective/loss/cost function)
  - Their way of combining features (e.g., linear vs. non-linear)

# Generative vs. Discriminative Classifiers

- **Generative** classifiers (learn **what the data “looks”** like in each category)
  - Attempt to model  $p(X,Y) = p(Y)p(X|Y)$  and compute  $p(Y|X)$  based on  $p(X|Y)$  and  $p(Y)$  by using Bayes Rule
  - Objective function is likelihood, thus indirectly measuring training errors
  - E.g., Naïve Bayes
- **Discriminative** classifiers (learn **what features separate categories**)
  - Attempt to model  $p(Y|X)$  directly
  - Objective function directly measures errors of categorization on training data
  - E.g., Logistic Regression, Support Vector Machine (SVM), k-Nearest Neighbors (kNN)

# Text Categorization: Generative Probabilistic Models

# Overview

- What is text categorization?
- Why text categorization?
- How to do text categorization?
  - **Generative probabilistic models**
  - Discriminative approaches
- How to evaluate categorization results?

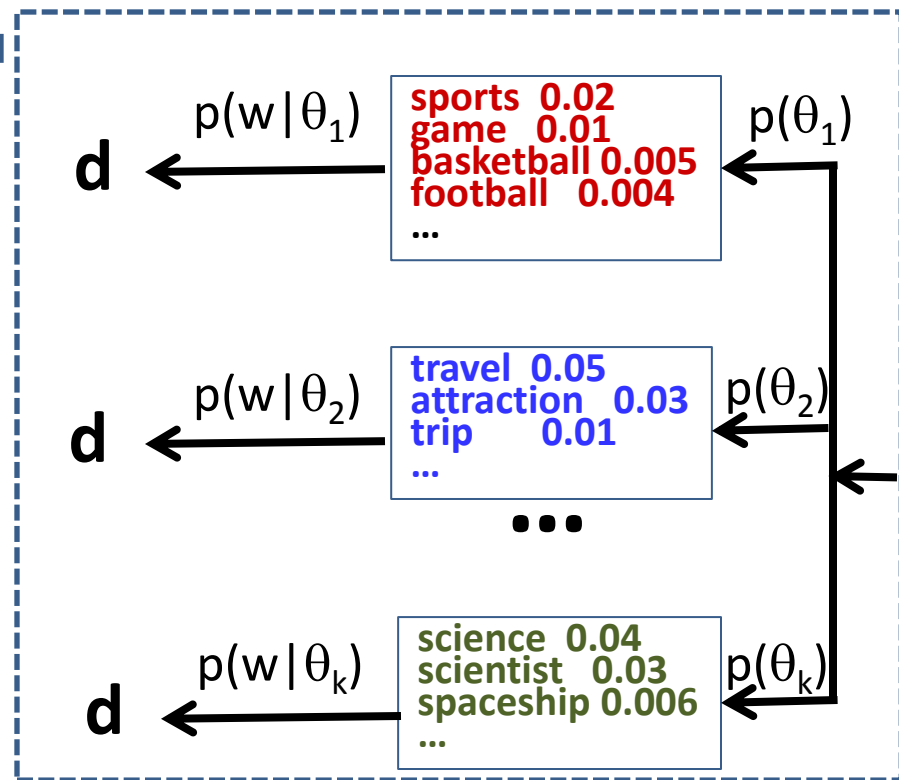
# Document Clustering Revisited

Which cluster does  $d$  belong to?  $\rightarrow$  Which  $\theta_i$  was used to generate  $d$ ?

$d = x_1 x_2 \dots x_L$  where  $x_i \in V$  

$$\begin{aligned}\text{cluster}(d) &= \arg \max_i p(\theta_i | d) \\ &= \arg \max_i p(d | \theta_i) p(\theta_i) \\ &= \arg \max_i \prod_{j=1}^L p(x_j | \theta_i) p(\theta_i) \\ &= \arg \max_i \prod_{w \in V} p(w | \theta_i)^{c(w, d)} p(\theta_i)\end{aligned}$$

$$\begin{aligned}p(\theta_i | d) &= \frac{p(d | \theta_i) p(\theta_i)}{p(d)} \\ &= \frac{p(d | \theta_i) p(\theta_i)}{\sum_{j=1}^k p(d | \theta_j) p(\theta_j)}\end{aligned}$$





# Text Categorization with Naïve Bayes Classifier

$d = x_1 x_2 \dots x_L$  where  $x_i \in V$

IF  $\theta_i$  represents category  $i$  accurately,  
then...

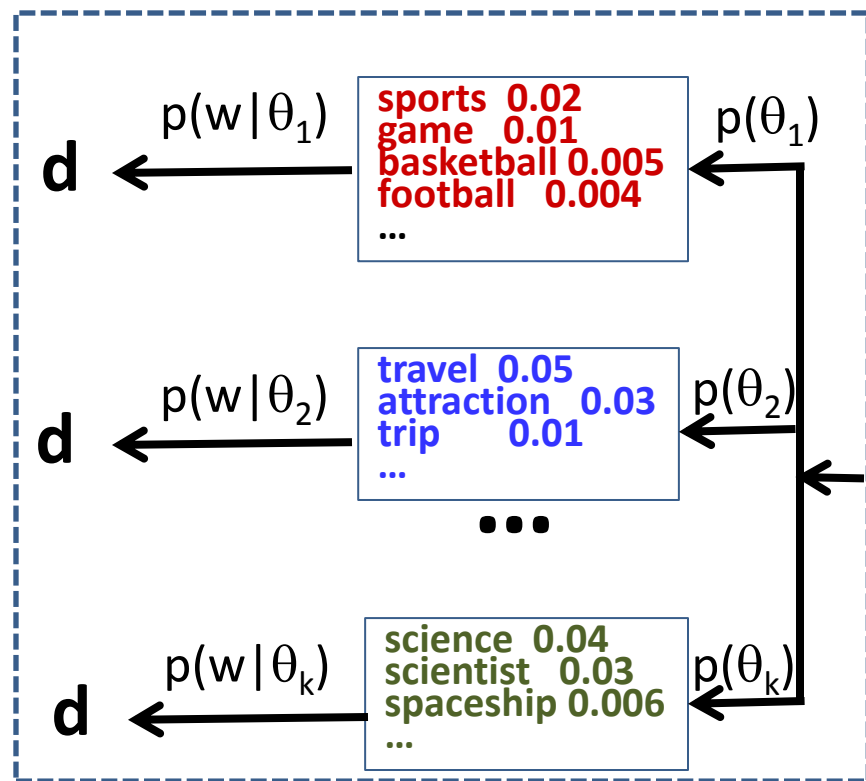
How can we make this happen?

$$\text{category}(d) = \arg \max_i p(\theta_i | d)$$

$$= \arg \max_i p(d | \theta_i) p(\theta_i)$$

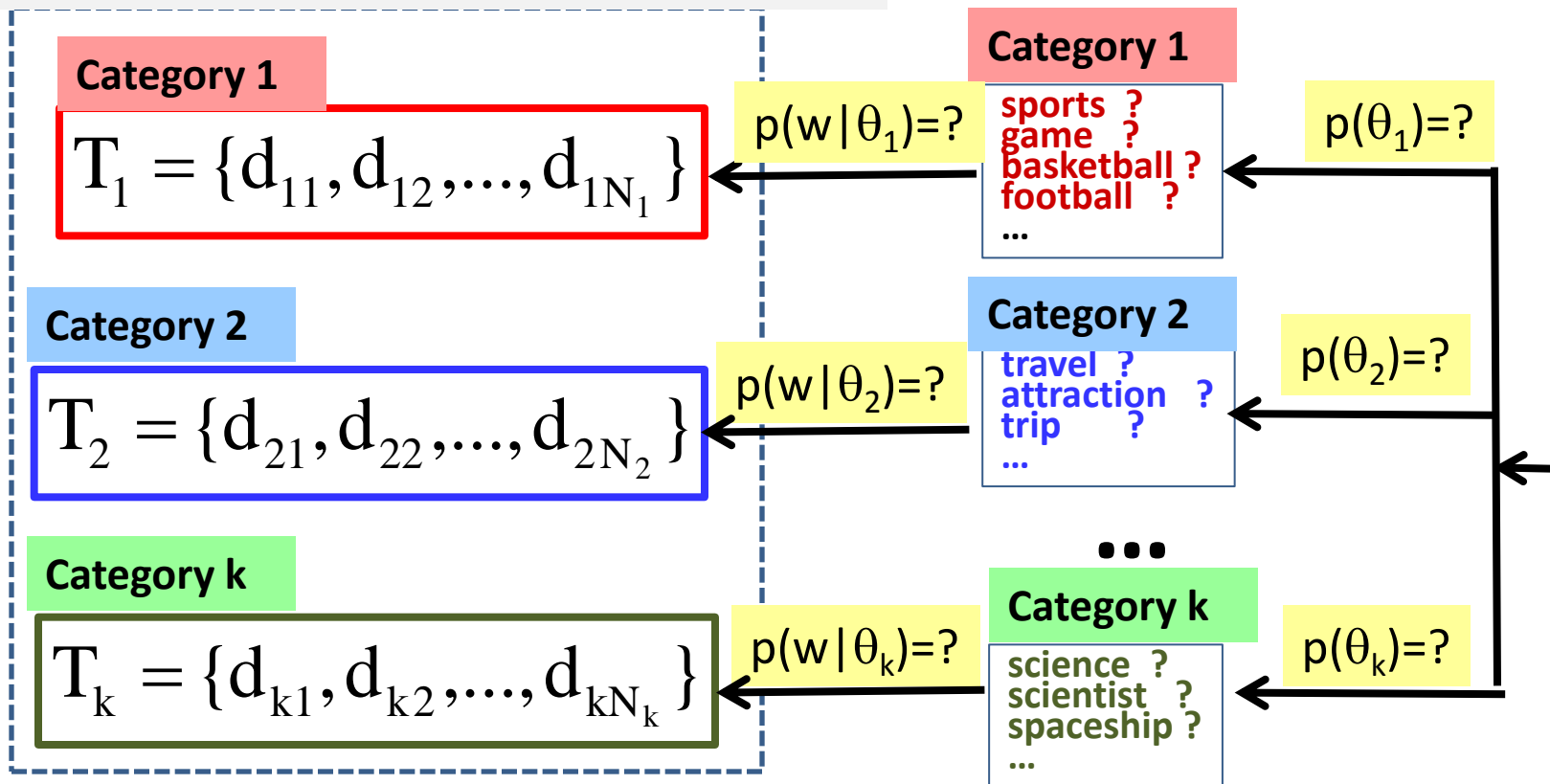
$$= \arg \max_i \prod_{w \in V} p(w | \theta_i)^{c(w, d)} p(\theta_i)$$

$$\text{category}(d) = \arg \max_i \log p(\theta_i) + \sum_{w \in V} c(w, d) \log p(w | \theta_i)$$



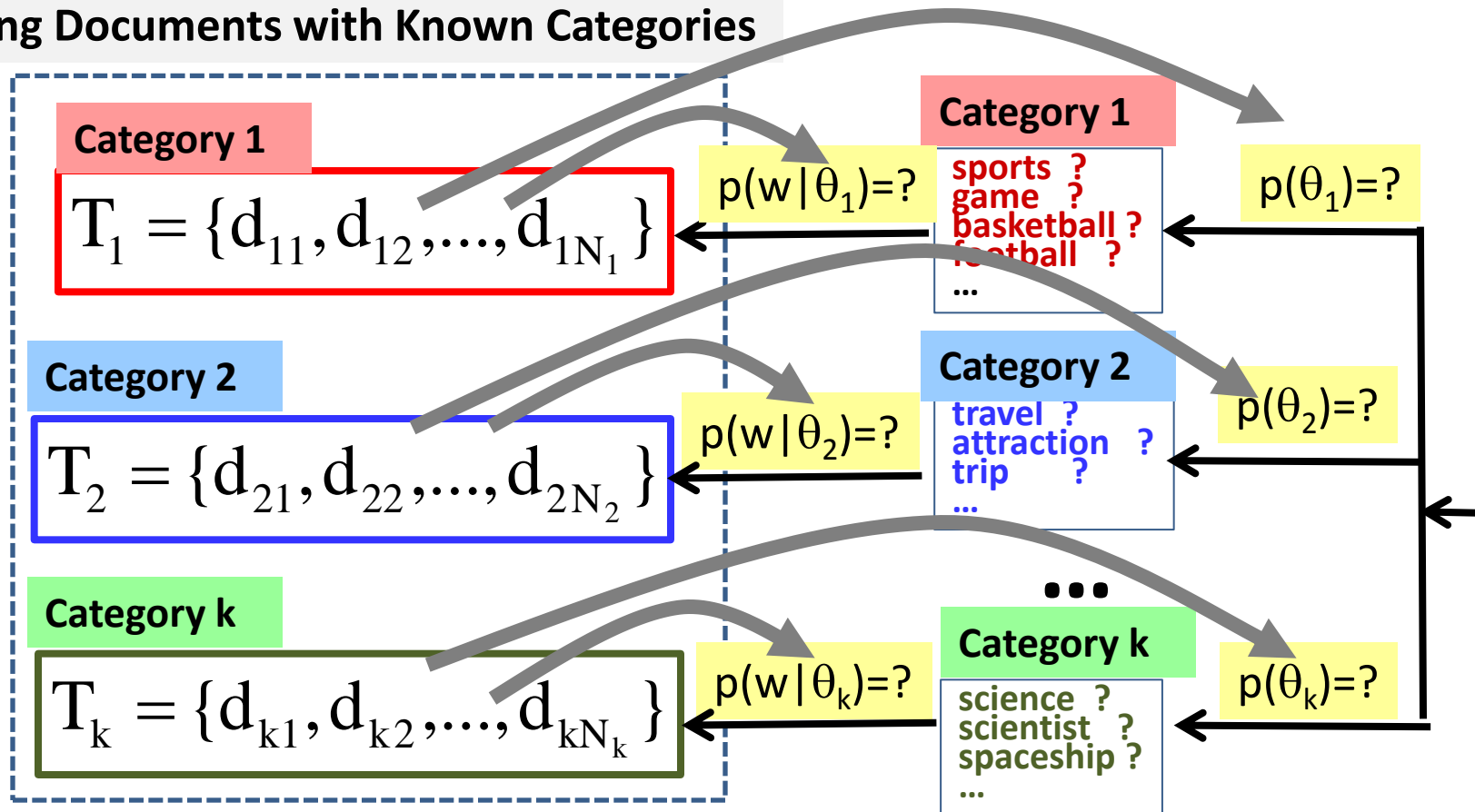
# Learn from the Training Data

## Training Documents with Known Categories



# How to Estimate $p(w | \theta_i)$ and $p(\theta_i)$

## Training Documents with Known Categories



# Naïve Bayes Classifier: $p(\theta_i)=?$ and $p(w | \theta_i)=?$

Category 1

$$T_1 = \{d_{11}, d_{12}, \dots, d_{1N_1}\}$$

Category 2

$$T_2 = \{d_{21}, d_{22}, \dots, d_{2N_2}\}$$

Category k

$$T_k = \{d_{k1}, d_{k2}, \dots, d_{kN_k}\}$$

Which category is most popular?

↓

$$p(\theta_i) = \frac{N_i}{\sum_{j=1}^k N_j} \propto |T_i|$$

$$p(w | \theta_i) = \frac{\sum_{j=1}^{N_i} c(w, d_{ij})}{\sum_{w' \in V} \sum_{j=1}^{N_i} c(w', d_{ij})} \propto c(w, T_i)$$

Which word is most frequent in category i?

What are the constraints on  $p(\theta_i)$  and  $p(w | \theta_i)$ ?

# Smoothing in Naïve Bayes

- Why smoothing?
  - Address data sparseness (training data is small → zero prob.)
  - Incorporate prior knowledge
  - Achieve discriminative weighting (i.e., IDF weighting)

- How?

$$p(\theta_i) = \frac{N_i + \delta}{\sum_{j=1}^k N_j + k\delta} \quad \delta \geq 0$$

What if  $\delta \rightarrow \infty$ ?

$p(w | \theta_B)$ : background LM

$$p(w | \theta_i) = \frac{\sum_{j=1}^{N_i} c(w, d_{ij}) + \mu p(w | \theta_B)}{\sum_{w' \in V} \sum_{j=1}^{N_i} c(w', d_{ij}) + \mu}$$

$\mu \geq 0$

$p(w | \theta_B) = 1/|V|$ ?

What if  $\mu \rightarrow \infty$ ?

# Anatomy of Naïve Bayes Classifier

Two categories:  $\theta_1$  and  $\theta_2$

$$\text{score}(d) = \log \frac{p(\theta_1 | d)}{p(\theta_2 | d)} = \log \frac{p(\theta_1) \prod_{w \in V} p(w | \theta_1)^{c(w,d)}}{p(\theta_2) \prod_{w \in V} p(w | \theta_2)^{c(w,d)}}$$

$$= \underbrace{\log \frac{p(\theta_1)}{p(\theta_2)}}_{\text{Category bias } (\beta_0) \text{ doesn't depend on } d!} + \sum_{w \in V} \underbrace{c(w,d)}_{\text{Sum over all words (features } \{f_i\})} \underbrace{\log \frac{p(w | \theta_1)}{p(w | \theta_2)}}_{\text{Weight on each word (feature) } \beta_i}$$

Feature value:  $f_i = c(w,d)$



Generalize

$$d = (f_1, f_2, \dots, f_M), \quad f_i \in \mathcal{R}$$

$$\text{score}(d) = \beta_0 + \sum_{i=1}^M f_i \beta_i \quad \beta_i \in \mathcal{R}$$

= Logistic Regression!