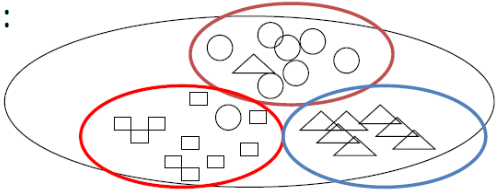
Text Clustering: Motivation

What Is Text Clustering?

- Discover "**natural structure**"
- **Group similar objects** together
- Objects can be documents, terms, passages, websites,…

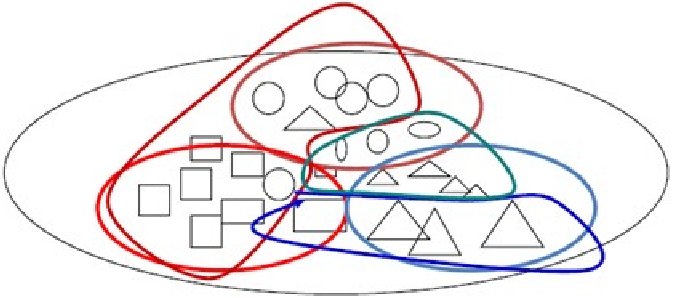
- Example:



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** (basis for evaluation, i.e. "**bias**") for assessing similarity!



Examples of Text Clustering

- Clustering docs in a **collection**
- **Term clustering** to define "concept"/"theme"/"**topic**"
- Clustering of passages/sentences or any **selected text segments** (e.g. about a topic discovered using a topic model) from larger text objects
- **Clustering of websites** (text obj has multiple docs)
- Text clusters can be **further clustered** to generate a hierarchy

Why Text Clustering?

- Very useful for text mining and exploratory text analysis:

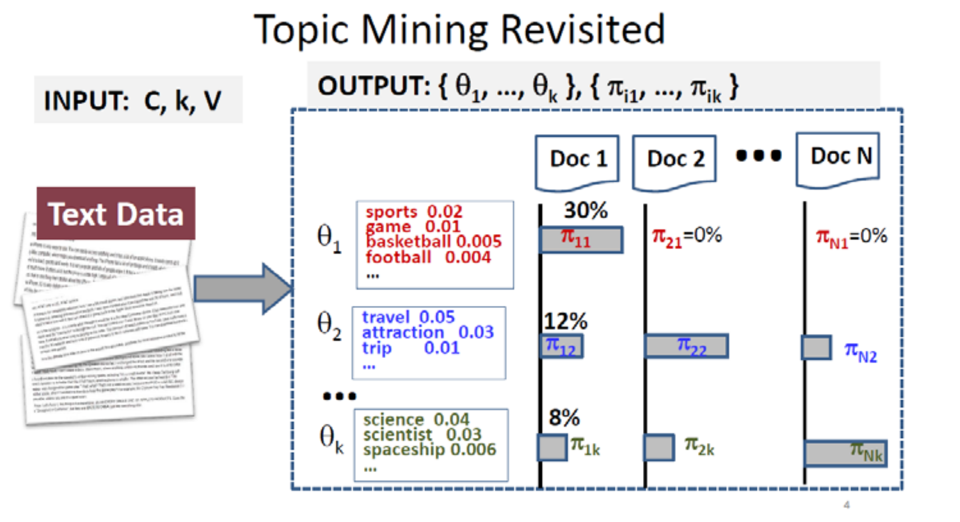
  ➔Get a **sense about the content of C** (e.g., what are some of the "typical"/representative documents in a collection?)

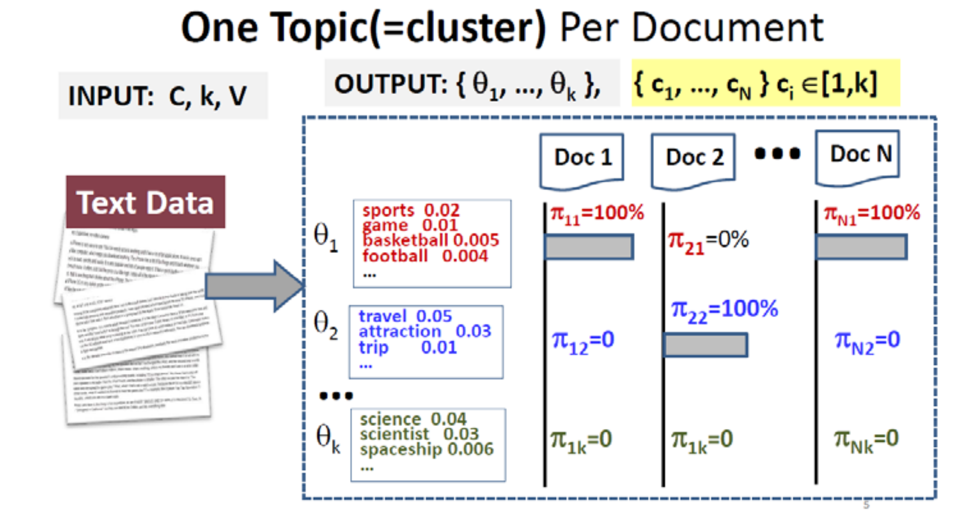
  ➔**Link (similar) text obj** (e.g., remove duplicate content)

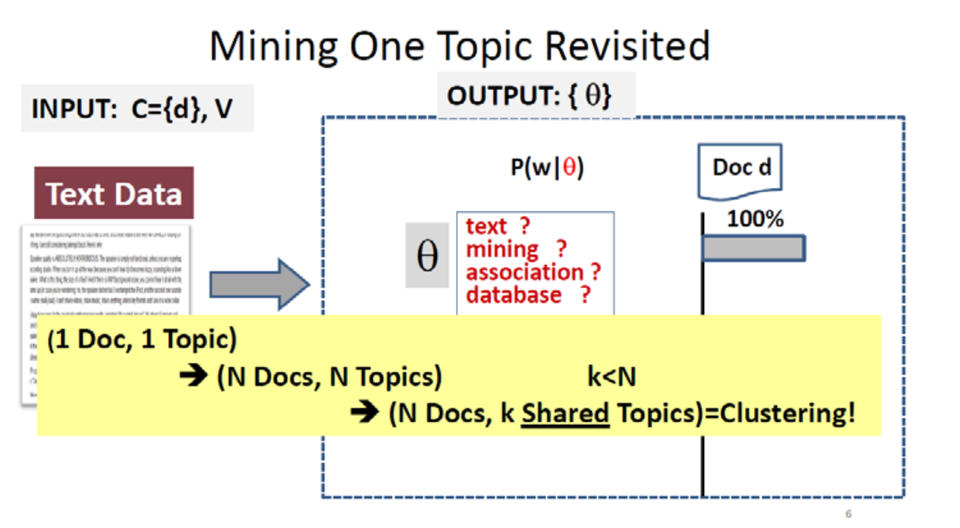
  ➔Create a **structure on the text** data (e.g., for browsing)

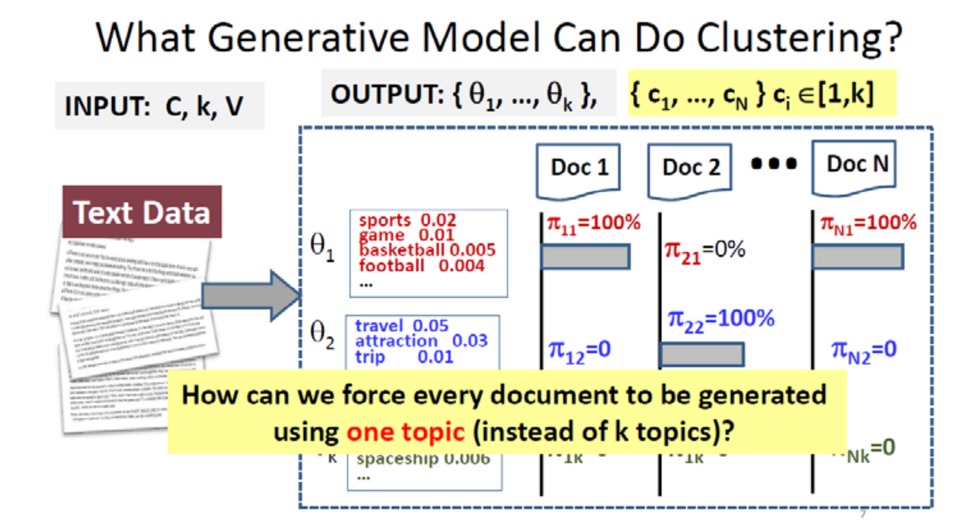
  ➔As a way to **induce additional features** (i.e., clusters) **for classification of text objects!!!**
- Examples of applications
  - Clustering of search results
  - Understanding major complaints in emails from customers

Text Clustering: Generative Probabilistic Models (Part 1)

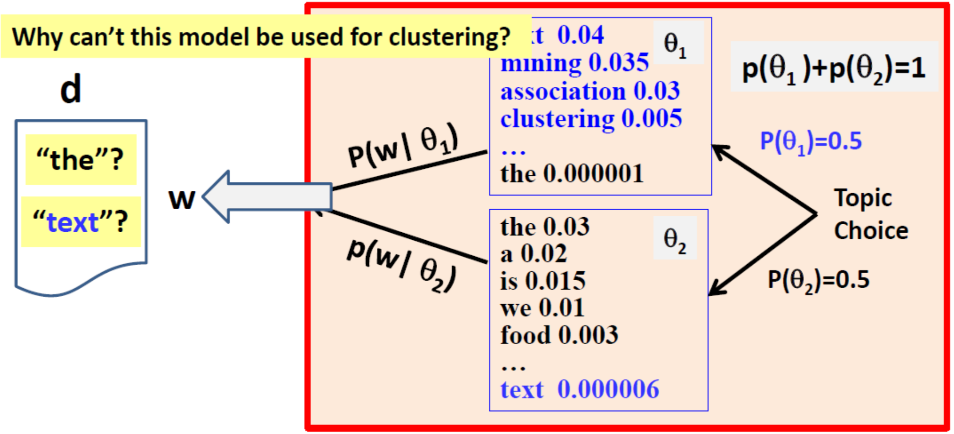

Topic Mining Revisited
INPUT: C, k, V
OUTPUT: { θ1, ..., θk }, { πi1, ..., πik }
Text Data
Doc 1
Doc 2
Doc N
θ1
sports 0.02
game 0.01
basketball 0.005
football 0.004
...
θ2
travel 0.05
attraction 0.03
trip 0.01
...
θk
science 0.04
scientist 0.03
spaceship 0.006
...
30%
π11
π21=0%
πN1=0%
12%
π12
π22
πN2
8%
π1k
π2k
πNk
4


One Topic(=cluster) Per Document
INPUT: C, k, V
OUTPUT: { θ1, ..., θk }, { c1, ..., cN } ci ∈[1,k]
Text Data
Doc 1
Doc 2
Doc N
θ1
sports 0.02
game 0.01
basketball 0.005
football 0.004
...
θ2
travel 0.05
attraction 0.03
trip 0.01
...
θk
science 0.04
scientist 0.03
spaceship 0.006
...
π11=100%
π21=0%
πN1=100%
π12=0
π22=100%
πN2=0
π1k=0
π1k=0
πNk=0
5

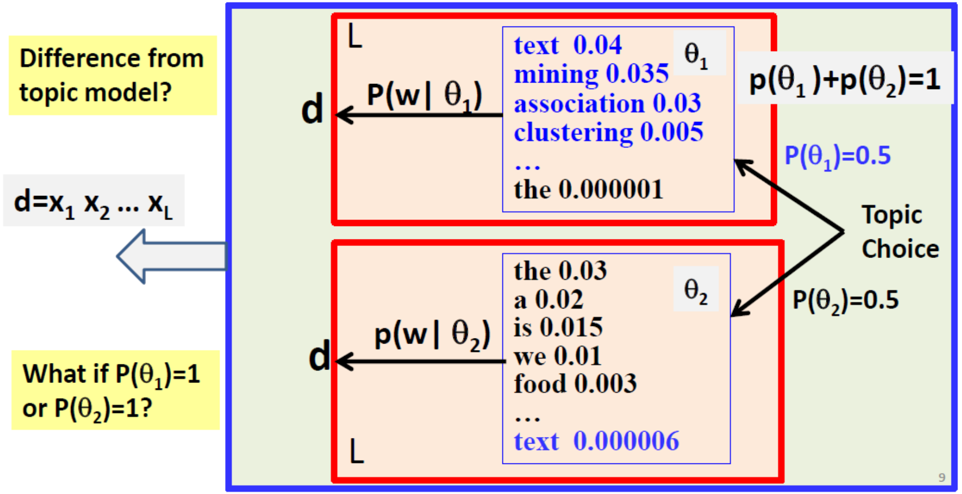

Mining One Topic Revisited
INPUT: C={d}, V
OUTPUT: { θ}
Text Data
P(w|θ)
Doc d
θ
text ?
mining ?
association ?
database ?
100%
(1 Doc, 1 Topic)
→ (N Docs, N Topics) k<N
→ (N Docs, k Shared Topics)=Clustering!
6


What Generative Model Can Do Clustering?
INPUT: C, k, V
OUTPUT: { θ1, ..., θk }, { c1, ..., cN } ci ∈[1,k]
Text Data
Doc 1
Doc 2
Doc N
θ1
sports 0.02
game 0.01
basketball 0.005
football 0.004
...
θ2
travel 0.05
attraction 0.03
trip 0.01
π11=100%
π21=0%
πN1=100%
π12=0
π22=100%
πN2=0
How can we force every document to be generated using one topic (instead of k topics)?
spaceship 0.006
...
7

Generative Topic Model Revisited



Mixture Model for Document Clustering



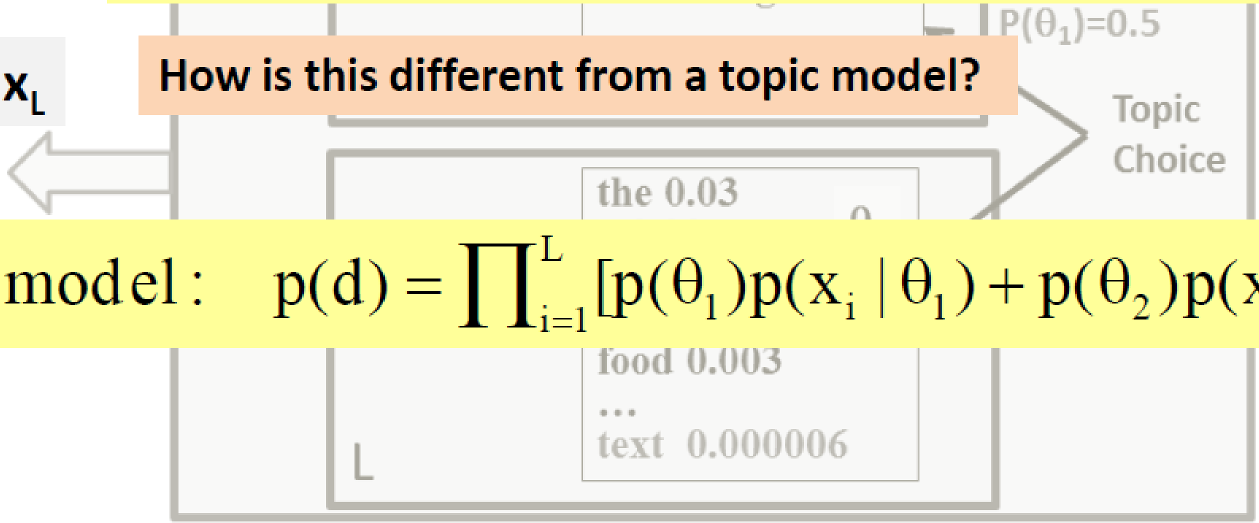
MOST IMPORTANT SLIDE: COMPARISON OF 2 ML F(X)

Likelihood Function: p(d)=?

$$p(d) = p(\theta_1)p(d \mid \theta_1) + p(\theta_2)p(d \mid \theta_2)$$
$$= p(\theta_1)\prod_{i=1}^{L} p(x_i \mid \theta_1) + p(\theta_2)\prod_{i=1}^{L} p(x_i \mid \theta_2)$$

$d = x_1\ x_2 \ldots x_L$

How is this different from a topic model?



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

Text Clustering:
Generative Probabilistic Models (Part 2)

How can we generalize it to include k topics/clusters? Mixture Model for Document Clustering

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

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

- Maximum Likelihood estimate

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

Cluster Allocation After Parameter Estimation

- **Parameters** of the mixture model: **Λ=({θ_i}; {$p(\theta_i)$}), i$\in$[1,k]**
  - Each θ_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 θ_i** that **most likely** has been used to generate d

$$c_d = \arg\max_i p(d \mid \theta_i)$$

  - **Likelihood + prior $p(\theta_i)$ (Bayesian):** favor large clusters

$$c_d = \arg\max_i p(d \mid \theta_i) p(\theta_i)$$

Text Clustering:
Generative Probabilistic Models (Part 3)

How Can We Compute the ML Estimate? 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 $Z_d \in[1, k]$**

$$p^{(n)}(Z_d = i \mid d) \propto p^{(n)}(\theta_i)\prod_{w\in V} p^{(n)}(w \mid \theta_i)^{c(w,d)} \qquad \sum_{i=1}^{k} p^{(n)}(Z_d = i \mid d) = 1$$

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

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

$$p^{(n+1)}(w \mid \theta_i) \propto \sum_{j=1}^{N} c(w,d_j) p^{(n)}(Z_{d_j} = 1 \mid d_j) \qquad \sum_{w\in V} p^{(n+1)}(w \mid \theta_i) = 1, \quad \forall i \in [1,k]$$

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**

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

Hidden variables:

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

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

$$=\frac{0.5*0.5^2*0.2^2}{0.5*0.5^2*0.2^2+0.5*0.1^2*0.1^2}=\frac{100}{101}$$

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

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

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

An Example of 2 Clusters (cont.)

From E-Step

	$P(Z_d=1\|d)$
d1	0.9
d2	0.1
d3	0.8

	c("text")	c("mining")
d1	2	3
d2	1	2
d3	4	3

M-Step $p(\theta_1)=?\ p(\theta_2)=?$

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

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

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

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

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

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

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

Summary of Generative Model for Clustering

- A slight **variation of topic model** used for clustering docs
  - Each **cluster - unigram LM $p(w|\theta_i)$ ➔ Term cluster**
  - A doc - first **choose a unigram LM**, then **generate ALL words** (with this single LM)
  - Estim. model parameters – 1) topic characterization of each cluster and 2) probabilistic assignment of doc into each cluster
  - **"Hard" clusters** can be obtained by **forcing a doc** into the cluster corresponding to the **unigram LM most likely** used to generate the doc
- **EM algorithm** can be used to compute the **ML estimate**
  - **Normalization** is often needed to avoid underflow

Text Clustering:
Similarity-based Approaches

Similarity-based Clustering: General Idea

- **Similarity function** for two text objects (i.e., “clustering bias”)
- Find an **optimal partitioning** of data to maximize intra-group and inter-group similarity
- Two optimal clustering strategies:
  - Progressively construct a hierarchy of clusters (hierarchical clustering)
    - **Bottom-u**p (agglomerative): gradually group similar objects into larger clusters
    - **Top-down** (divisive): gradually partition the data into smaller clusters
  - Start with **initial tentative clustering** + **iterations** to 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

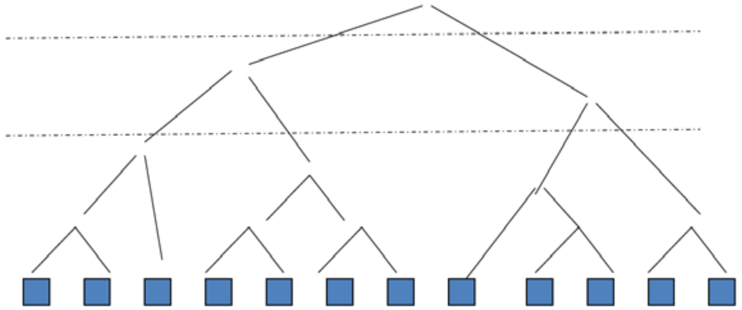
- Given a **similarity function** to measure similarity between two objects
- Gradually **group** similar objects together in a **bottom-up** fashion to **form a hierarchy**
- Stop when some **stopping criterion** is met
- Variations: **different ways** to compute **group similarity** based on individual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

Three popular methods to compute **group similarity**:

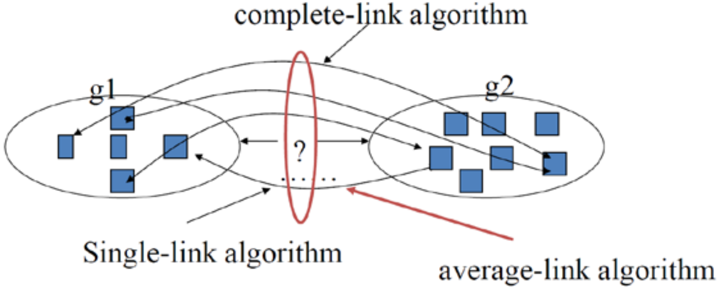
Given two groups g1 and g2:

- **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**

Similarity-induced Structure



Group Similarity Illustrated



Comparison

- Single-link
  - "**Loose**" clusters
  - <u>Individual</u> decision, <u>sensitive to outliers</u>
- Complete-link
  - "**Tight**" clusters
  - <u>Individual</u> decision, <u>sensitive to outliers</u>
- Average-link
  - "**In between**"
  - <u>Group</u> decision, <u>insensitive to outliers</u>
- Which is the best – **application dependent**!

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**
- **Re-compute the centroid** for each cluster based on the newly assigned vectors in the cluster
- **Repeat** this process until the **similarity-based objective function** (i.e., within cluster sum of squares) **converges** (to a local minimum)

Summary of Clustering Methods

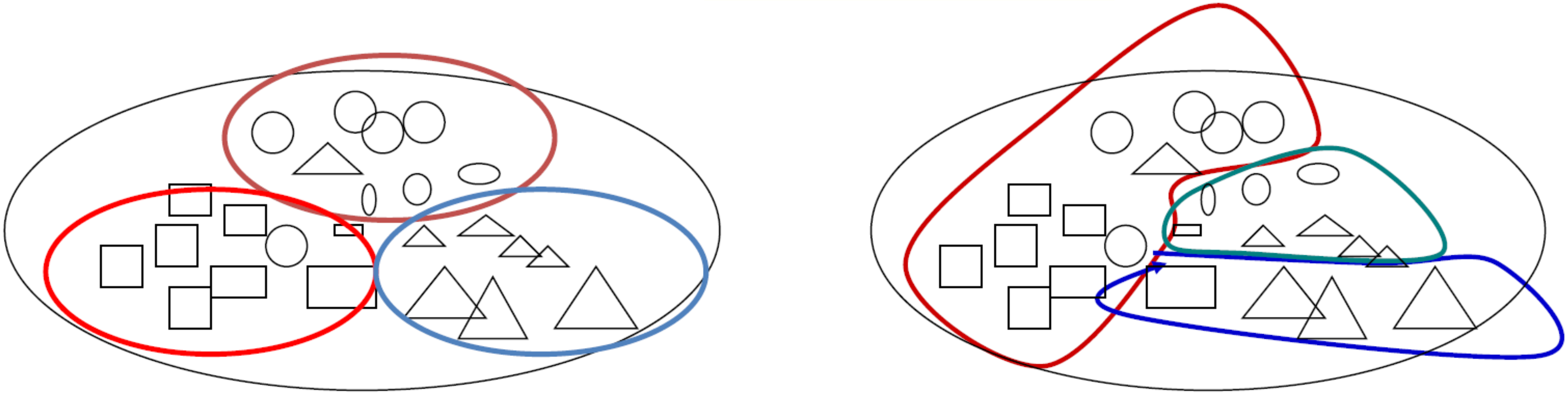
- **Model based** approaches (mixture model)
  - Uses an implicit **similarity function** (model ➔ clustering bias)
  - Cluster structure is "built" into a generative model
  - Complex generative models can discover complex structures
  - Prior can be leveraged to further customize the clustering algorithm
  - 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

The "Clustering Bias"

- **Any two objects** can be **similar**, depending on <u>how you look</u> 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

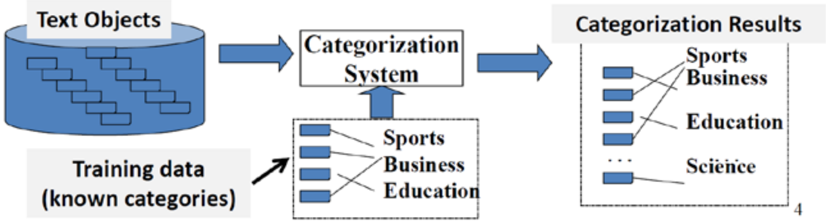
Summary of Text Clustering

- Text clustering is an **unsupervised general text mining** technique to
  - obtain an **overall** picture of the **text con**tent (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 <u>right generative model</u> or the <u>right similarity function</u>)
  - 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**

Text Categorization: Motivation

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**



Variants of Problem Formulation

- **Binary** categorization: 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

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., topics, sentiments)
  - **"External"** categories that characterize an entity associated with the text object (e.g., author attribution, other meaningful categories)
- 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

Why Text Categorization?

- To **enrich text representation** (understanding)
  - Multiple levels of text representation (keywords + categories)
  - Semantic categories - 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

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**

"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)

ML for Text Categorization

- **General setup**: Learn a classifier f: X➔Y
  - Input: X = all text obj-s; Output: Y = all categories
  - Learn a classifier function, f: X➔Y, such that **f(x)=y ∈Y** gives the **correct category** for x∈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)

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

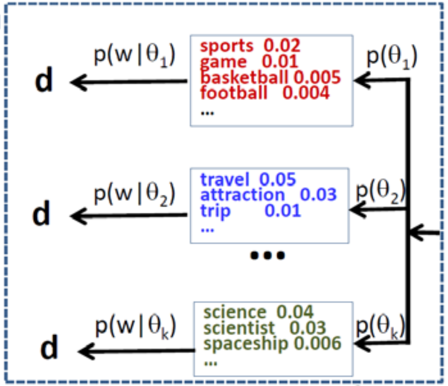
Document Clustering Revisited

Which cluster does d belong to? ➔ Which θ_i was used to generate d?

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

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

$$p(\theta_i \mid d) = \frac{p(d \mid \theta_i) p(\theta_i)}{p(d)} = \frac{p(d \mid \theta_i) p(\theta_i)}{\sum_{j=1}^{k} p(d \mid \theta_j) p(\theta_j)}$$



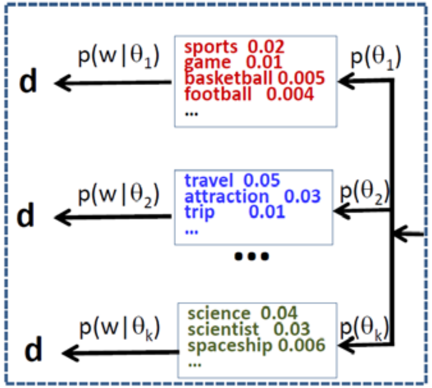
Text Categorization with Naïve Bayes Classifier

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

IF θ_i represents category i accurately, then…

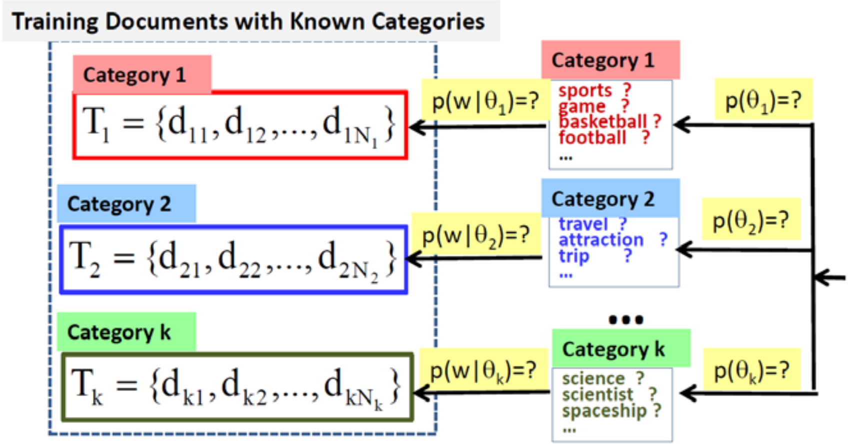
How can we make this happen?

$$\begin{aligned} category(d) &= \arg\max_i p(\theta_i \mid d) \\ &= \arg\max_i p(d \mid \theta_i) p(\theta_i) \\ &= \arg\max_i \prod_{w \in V} p(w \mid \theta_i)^{c(w,d)} p(\theta_i) \end{aligned}$$

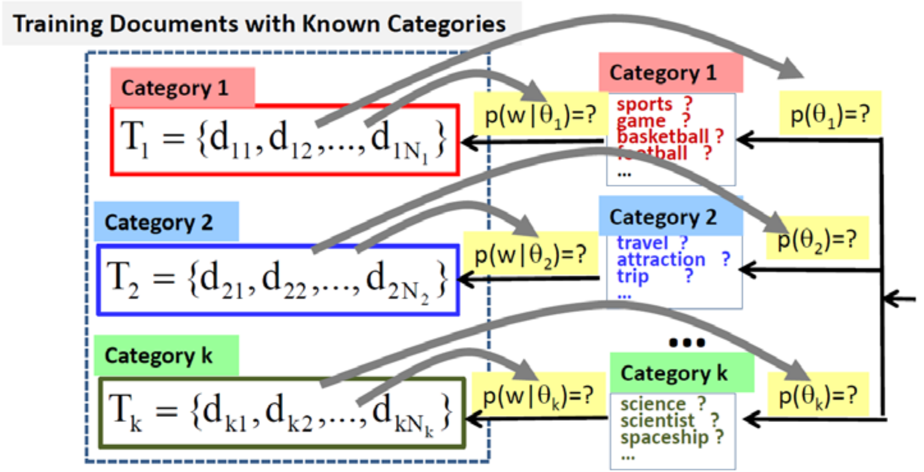


$$\boxed{category(d) = \arg\max_i \log p(\theta_i) + \sum_{w \in V} c(w,d) \log p(w \mid \theta_i)}$$

Learn from the Training Data



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



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

Category 1

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

Category 2

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

Category k

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

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 $p(w|\theta_i)$?

Anatomy of Naïve Bayes Classifier

Two categories: θ_1 and θ_2

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

$$= \boxed{\log \frac{p(\theta_1)}{p(\theta_2)}} + \sum_{w \in V} c(w,d) \boxed{\log \frac{p(w|\theta_1)}{p(w|\theta_2)}}$$

Category bias (β_0) doesn't depend on d!

Sum over all words (features $\{f_i\}$)

Feature value: f_i=c(w,d)

Weight on each word (feature) β_i

Generalize

$$d = (f_1, f_2, \ldots, f_M), \quad f_i \in \Re$$

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

= Logistic Regression!

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 $\delta \to \infty$?

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

$$p(w|\theta_i) = \frac{\sum_{j=1}^{N_i} c(w, d_{ij}) + \mu p(w|\theta_B)}{\sum_{w' \in V} \sum_{j=1}^{N_i} c(w', d_{ij}) + \mu} \quad \mu \geq 0$$

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

What if $\mu \to \infty$?