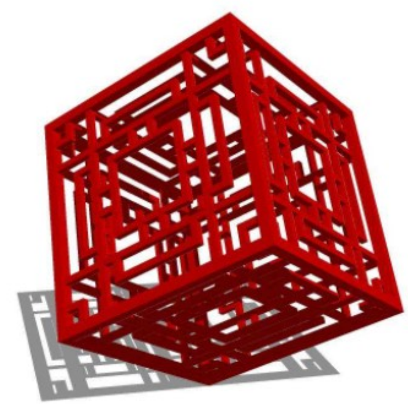
第七章 文本挖掘与主题模型

王 星
中国人民大学统计学院
wangxingwsidom@126.com

大 纲

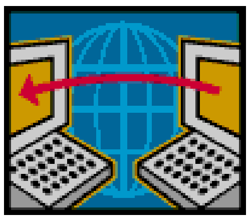
1. 文本预处理与信息提取(IR)
   ① DTM;
   ② Normalization;
   ③ TF-IDF.
2. 潜语义模型 Latent Semantic Analysis(Indexing)LSA,LSI
   ① LM
   ② PLSA
3. 主题模型LDA



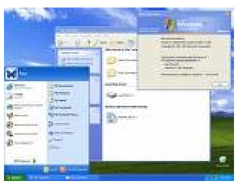
Text data **grow quickly** and cover **all kinds of topics**

How can we turn "big text data" into "big knowledge"?

WWW



Personal



Email

Enterprise

Government

Literature

News

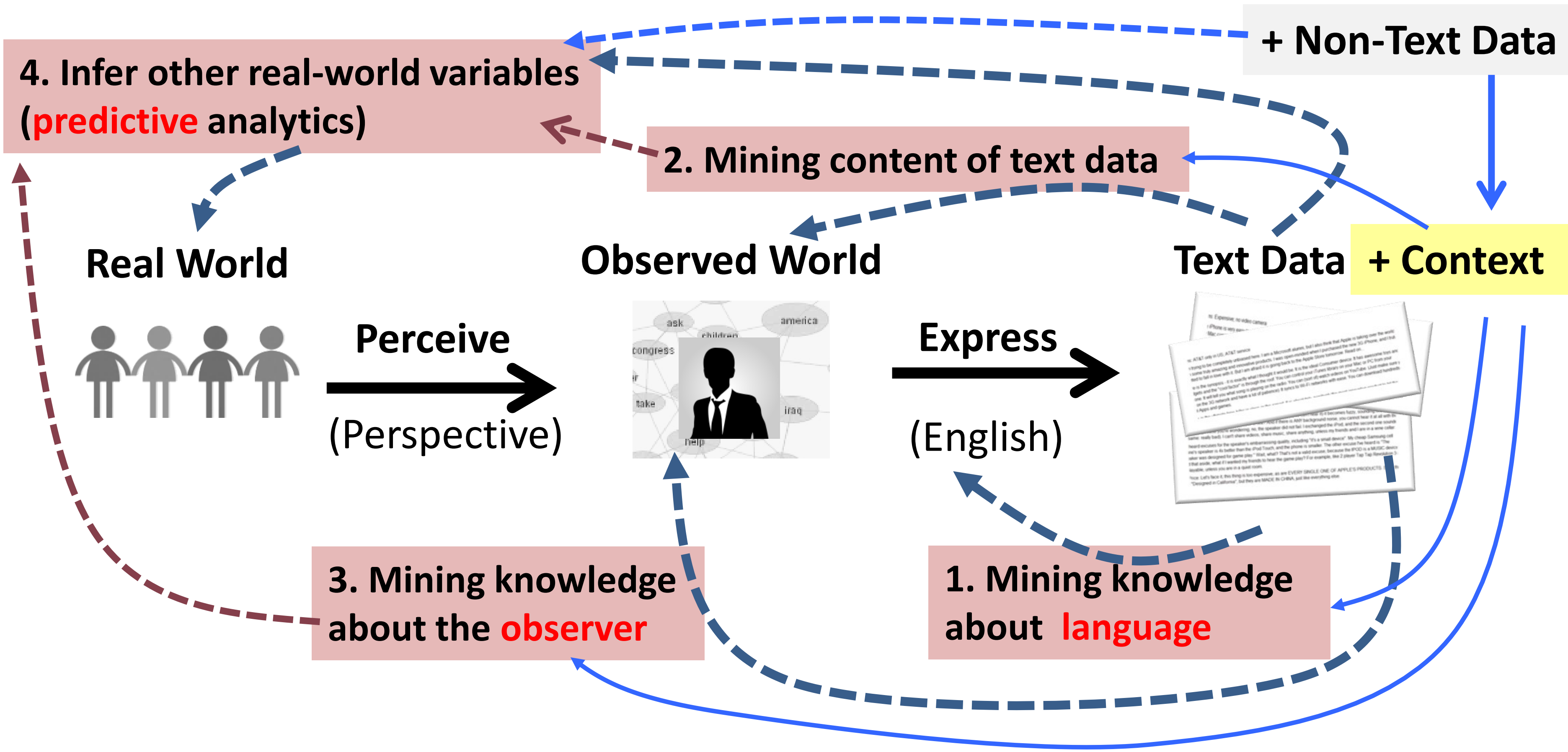
Social Media

• • •

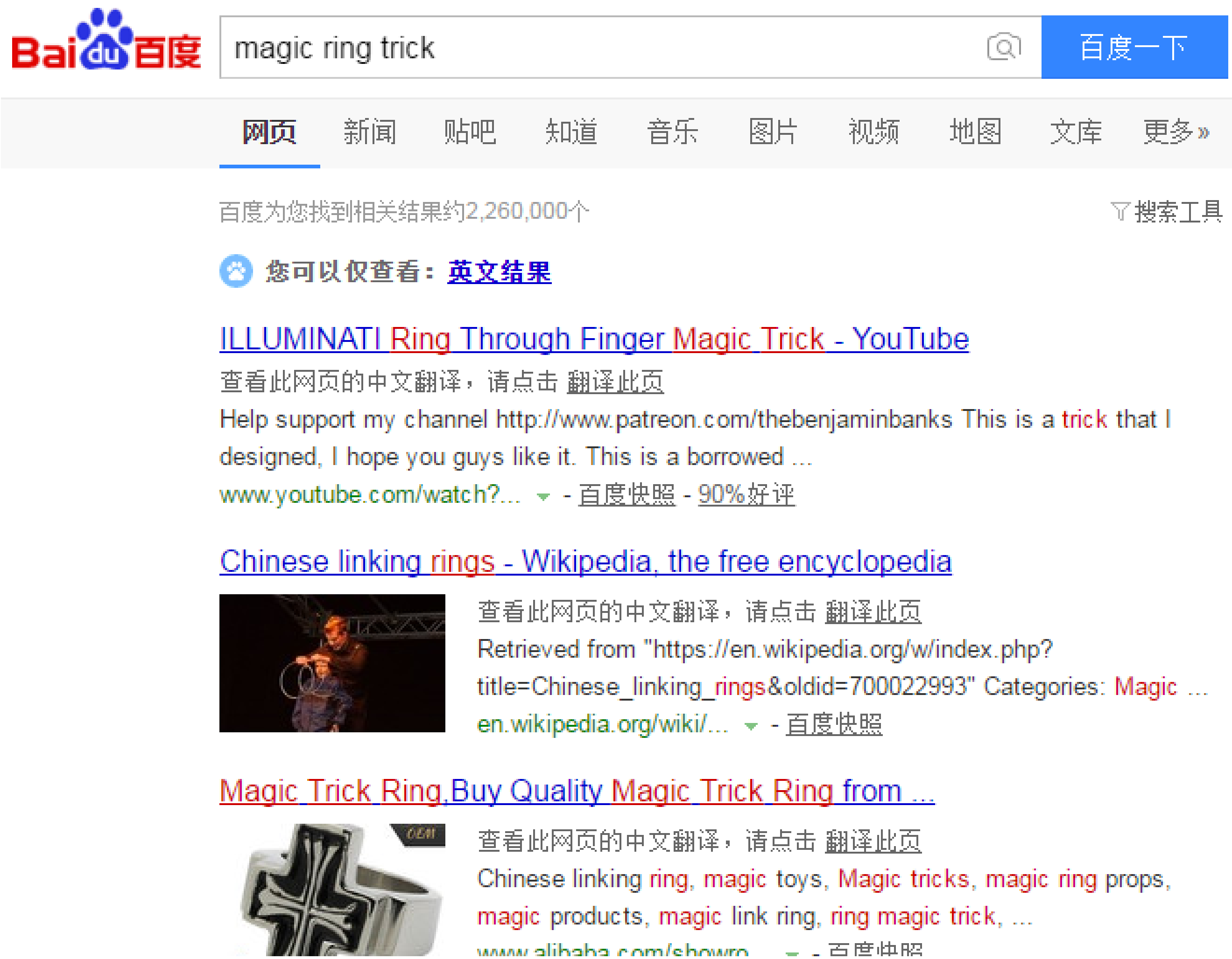
Unique Value of Text Data

- **Useful** in **all big data applications** (since humans are involved in all application domains and they generate text data)
- Especially useful for mining **knowledge** about **people's behavior, attitude, and opinions**
  - The **subjectivity** of human sensors creates both **challenges** in accurately understanding the truth behind text data and also **opportunities** to mine properties about the sensors themselves.
- **Directly express knowledge** about our world ➔ **Small text** data are **also useful**!

Opportunities of Text Mining Applications



1. Information Retrieval



Information retrieval and representations

- **Information retrieval**: given a set of documents (e.g., webpages),our problem is to pull up the k most similar documents to a given query (e.g., "magic ring trick")

- First step is to think of a way of **representing** these documents. We want our representation to:
  - Be easy to generate from the raw documents, and be easy to work with
  - (useful) Highlight important aspects of the documents, and suppress unimportant aspects
- There is kind of a **trade-off** between these two ideas

Try using the meaning of documents

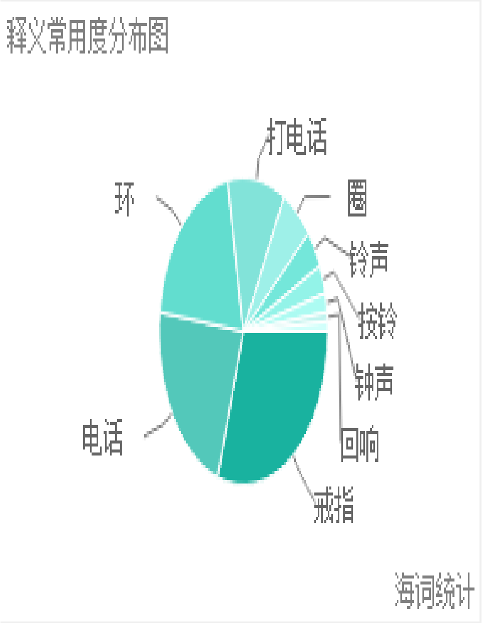
- What if we tried to represent the meaning of documents? E.g.,
  - type.of.trick = sleight of hand;
  - date.of.origin = 1st century;
  - place.of.origin = Turkey, Egypt;
  - name.origin = Chinese jugglers
  - in Britain; ...
- This would be good in terms of our second idea (useful and efficient data reduction), but not our first one (extremely hard to generate, and even hard to use!)

Bag-of-words representation

- Bag-of-words representation of a document is very simple-minded: just list all the words and how many times they appeared. E.g.,

magic = 29; ring = 34; trick = 6; illusion = 7; link = 9; ...

- Very easy to generate and easy to use (first idea), but is it too much of a reduction, or can it still be useful (second idea)?
- Idea: by itself “ring" can take on a lot of meanings, but we can learn from the other words in the document besides “ring". E.g.,
  - Words “perform", “illusion", “gimmick", “Chinese", “unlink", “audience", “stage" suggest the right type of rings;
  - Words “diamond", “carat", “gold", “band", “wedding", ”engagement", “anniversary" suggest the wrong type



Counting words

- Recall problem: given a query and a set of documents, find the k documents most similar to the query

Counting words:

- First make a list of all of the words present in the documents and the query
- Index the words $w = 1, \ldots W$ (e.g., in alphabetical order), and the documents $d = 1, \ldots D$ (just pick some order)
- For each document d, count how many times each word w appears (could be zero), and call this X_{dw}. The vector $X_d = (X_{d1}, \ldots X_{dW})$ gives us the word counts for the dth document
- Do the same thing for the query: let Y_w be the number of times the wth word appears, so the vector $Y = (Y_1, \ldots Y_W)$ contains the word counts for the query

Simple example

Documents:

1: “Shiyuan loves statistics." and 2: “Zhufei hates, hates statistics!“

Query:

D = 2 documents and W = 5 words total. For each document and query, we count the number of occurences of each word:

	hates	'Zhufei	loves	Shiyuan	statistics
X_1	0	0	1	1	1
X_2	2	1	0	0	1
Y	1	0	0	0	1

This is called the document-term matrix

Distances and similarity measures

- We represented each document Xd and query Y in a convenient vector format. Now how to measure similarity between vectors, or equivalently, dissimilarity or distance?
- Measures of distance between n-dimensional vectors X; Y

  - The ℓ_2 or Euclidean distance is

$$\|X - Y\|_2 = \sqrt{\sum_{i=1}^{n} (X_i - Y_i)^2}$$

  - The ℓ_1 or Manhattan distance is

$$\|X - Y\|_1 = \sum_{i=1}^{n} |X_i - Y_i|$$

Basic idea: find k vectors X_d with the smallest $\|X_d - Y\|_2$
(Note: ℓ_1 distance doesn't work as well here)

Bigger Example(TMNT.R)

Documents: 8 Wikipedia articles, 4 about the TMNT Leonardo, Raphael, Michelangelo, and Donatello, and 4 about the painters of the same name

1 2 3 4 5 6 7 8

Query: "Raphael is cool but rude, Michelangelo is a party dude!"

	but	cool	dude	party	michelangelo	raphael	rude	...	dist
doc 1	19	0	0	0	4	24	0		309.453
doc 2	8	1	0	0	7	45	1		185.183
doc 3	7	0	4	3	77	23	0		330.970
doc 4	2	0	0	0	4	11	0		220.200
doc 5	17	0	0	0	9	6	0		928.467
doc 6	36	0	0	0	17	101	0		646.474
doc 7	10	0	0	0	159	2	0		527.256
doc 8	2	0	0	0	0	0	0		196.140
query	1	1	1	1	1	1	1		0.000

Varying document lengths and normalization

- Different documents have different lengths. Total word counts:

doc 1	doc 2	doc 3	doc 4	doc 5	doc 6	doc 7	doc 8	query
3114	1976	3330	2143	8962	6524	4618	1766	7

- Wikipedia entry on Michelangelo the painter is almost twice as long as that on Michelangelo the TMNT (6524 vs 3330 words). And query is only 7 words long! We should normalize the count vectors X and Y in some way
  - Document length normalization: divide X by its sum,

$$X \leftarrow X / \sum_{w=1}^{W} X_w$$

  - ℓ_2 length normalization: divide X by its ℓ_2 length,

$$X \leftarrow X / \|X\|_2$$

Back to our Wikipedia example

```
> d
                  dist/doclen dist/l2len
1 (tmnt leo)      0.3852639   1.373039
2 (tmnt rap)      0.3777607   1.321860
3 (tmnt mic)      0.3781185   1.319045
4 (tmnt don)      0.3887625   1.393433
5 (real leo)      0.3904765   1.404966
6 (real rap)      0.3820547   1.349480
7 (real mic)      0.3811387   1.325174
8 (real don)      0.3932484   1.411498
9 (tmnt leo)      0.0000000   0.000000
```

- So far we've dealt with varying document lenghts. What about some words being more helpful than others? **Common words**,especially, are not going to help us nd relevant documents

Common words and IDF weighting

To deal with common words, we could just keep a list of words like “the", “this", “that", etc. to exclude from our representation. But this would be both too crude and time consuming.

Inverse document frequency (IDF) weighting is smarter and more efficient

- For each word w, let n_w be the number of documents that contain this word
- Then for each vector X_d and Y, multiply wth component by $\log(D/n_w)$

If a word appears in every document, then it gets a weight of zero, so effectively tossed out of the representation

(Future reference: IDF performs something like variable selection)

Putting it all together

Think of the document-term matrix:

	word 1	word 2	...	word W
doc 1				
doc 2				
$\vdots$				
doc D				

- Normalization scales each row by something (divides a row vector X by its sum $\sum_{i=1}^{W} X_i$ or its ℓ_2 norm $\|X\|_2$)
- IDF weighting scales each column by something (multiplies the wth column by $\log(D/n_w)$)
- We can use both, just normalize first and then perform IDF weighting

Back to our Wikipedia example, again

	dist/doclen/idf	dist/l2len/idf
doc 1 (tmnt leo)	0.623	1.704
doc 2 (tmnt rap)	0.622	1.708
doc 3 (tmnt mic)	0.620	1.679
doc 4 (tmnt don)	0.623	1.713
doc 5 (real leo)	0.622	1.693
doc 6 (real rap)	0.622	1.703
doc 7 (real mic)	0.622	1.690
doc 8 (real don)	0.624	1.747
query	0.000	0.000

- This didn't work as well as we might have hoped. Why?
- (Hint: our collection only contains 8 documents and 1 query ...)

Text mining in R

Helpful methods implemented in the package tm, available on the CRAN repository

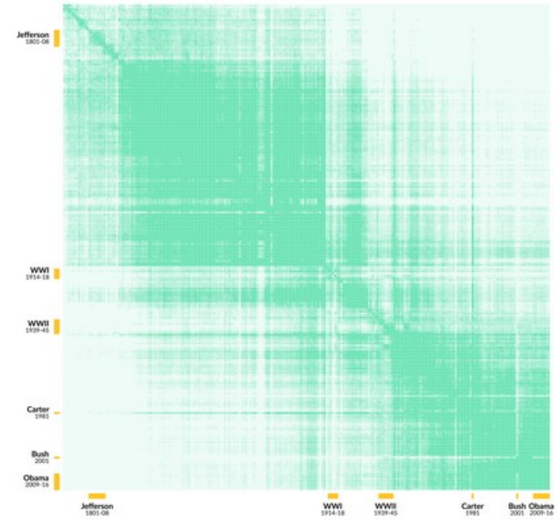
E.g.,

```
dtm = DocumentTermMatrix(corp,
control=list(tolower=TRUE, # 转换成小写字母
removePunctuation=TRUE, # 去掉标点
removeNumbers=TRUE, # 去掉数字
stemming=TRUE,   #词干化
weighting=weightTfIdf))   #TfIDF化
```

- 1

Computing similarities between addresses

A lot of this post is about comparing different State of the Union addresses to see how similar they are. Here's the (relatively simple) text analysis methodology for doing that.I took the data set of State of the Union address texts and performed the following steps:

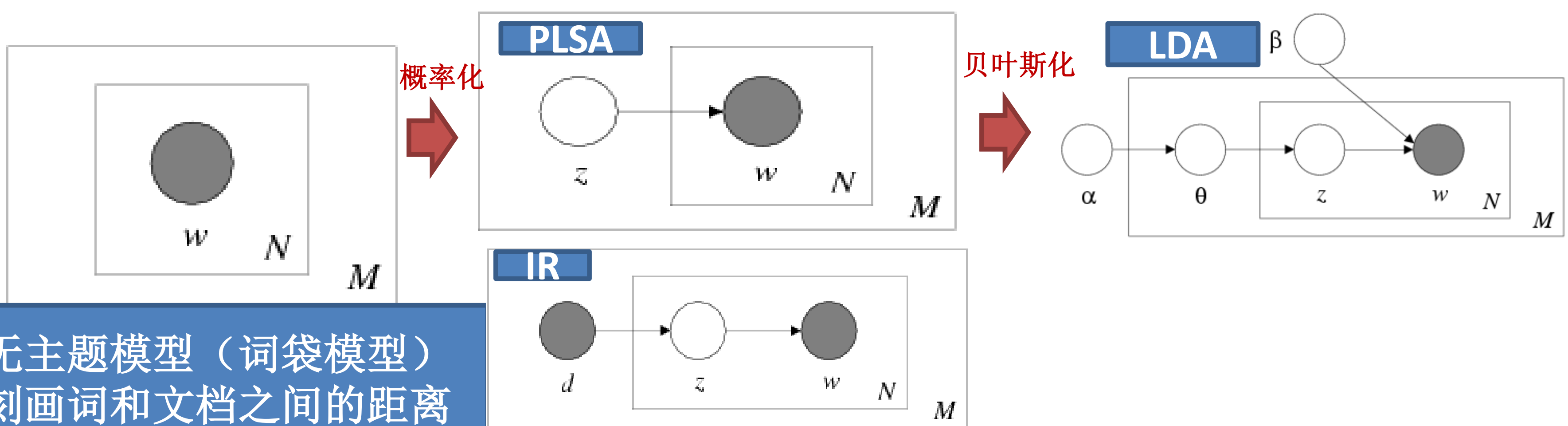


- split each address into sentences, and each sentence into words;
- combined the list of sentences for each address into a bag of words;
- removed stopwords (e.g., “the”,“a”,“an”), very rare words, numbers, and punctuation;
- created a sparse matrix with the word counts for each speech (number of addresses by number of words);
- weighted the words using log entropy weighting;and finally computed the cosine similarity between each pair of address row vectors.
- This results in a matrix of similarity scores between addresses, which can be visualized as follows.

同义词、多义词，文档与词的关系是怎样建立起来的（生成机制），在文档和词之间重要的连接参数是什么？如何估计

从潜语义模型到主题模型演变简史

- Papadimitriou、Raghavan、Tamaki和Vempala在1998年发表的论文中提出了潜在语义索引。
- 1999年，Thomas Hofmann又在此基础上，提出了概率性潜在语义索引（Probabilistic Latent Semantic Indexing，简称PLSI）。
- Blei, David M.、吴恩达和Jordan Michael I于2003年提出隐狄利克雷分配LDA是目前最常见的主题模型，是PLSI的推广版本。LDA允许文档拥有多种主题。
- 其它主题模型一般是在LDA基础上改进的。例如Pachinko分布在LDA度量词语关联之上，还加入了主题的关联度。



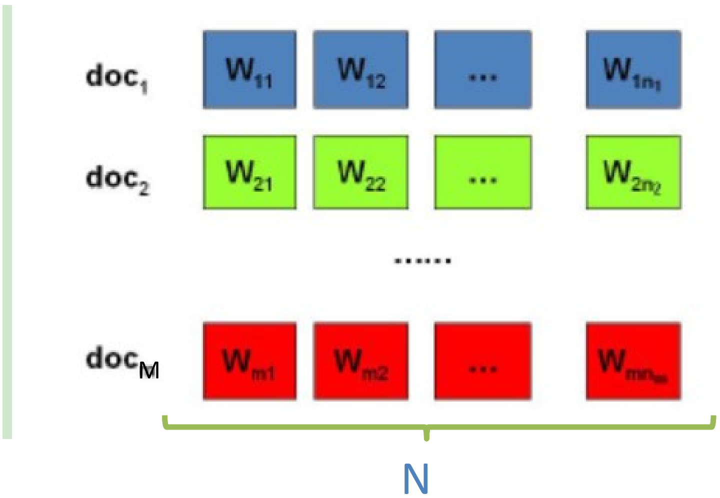
无主题模型（词袋模型）
刻画词和文档之间的距离

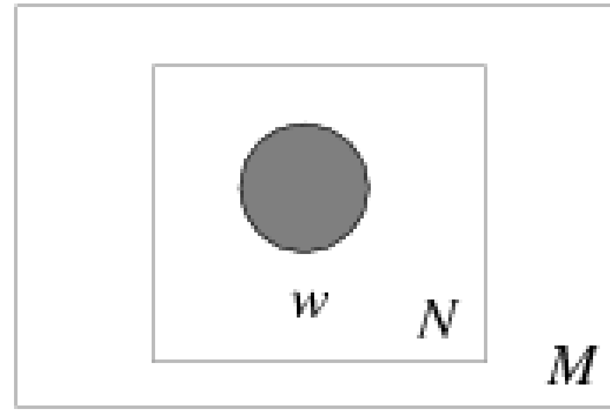
2.1The Simplest Language Model
(一元语言模型Unigram Model)

- 假设单词之间互相独立的
- 一篇文章的分布称为一元语言模型：

  $p(w_1\ w_2\ \ldots\ w_n)=p(w_1)p(w_2)\ldots p(w_n)$
- 参数: $\{p(w_i)\}$ $p(w_1)+\ldots+p(w_N)=1$ (N is voc. size)
- 常用多项分布表示一篇文章
- 一段文字可以认为是从词分布中产生的样本

Unigram Model—无主题模型

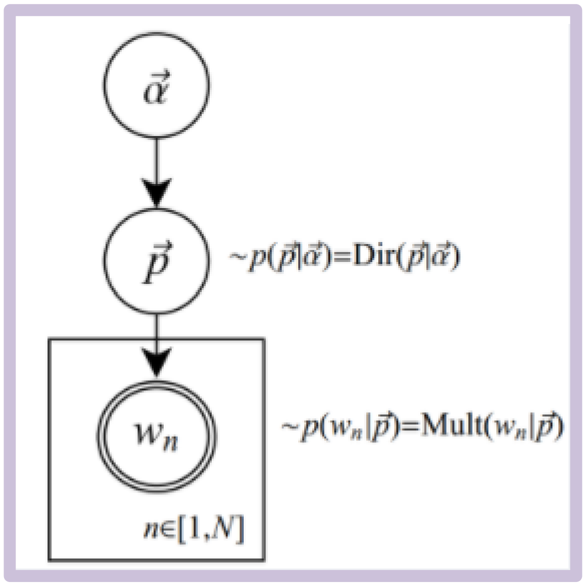




unigram model假设文本中的词频服从Multinomial分布，Multinomial分布的先验分布是Dirichlet分布。 上图中的w.n表示在文本中观察到的第n个词，n∈[1,N]表示该文本中一共有N个单词。加上方框表示重复观测，即一共有N个这样的随机变量。

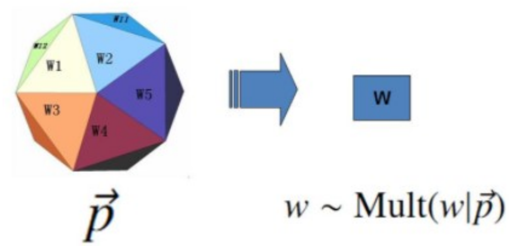
$$p(\mathbf{w}) = \prod_{n=1}^{N} p(w_n)$$

Unigram Model：文档是怎样生成的？



最简单的 Unigram Model：

假设词典中一共有N个词，最简单的 Unigram Model 假设上帝是按照如下的游戏规则产生文本的。



其中，p和α是隐含未知变量：

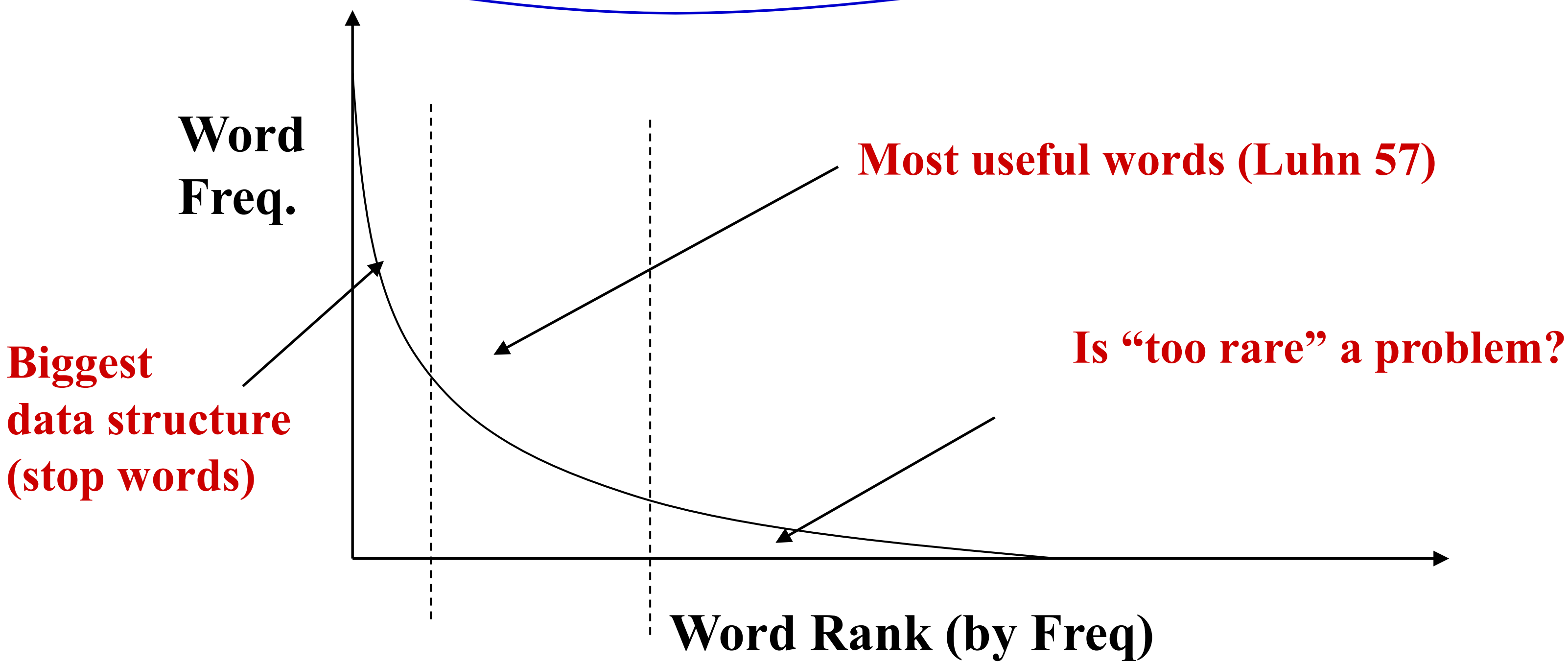
- p是词服从的Multinomial分布的参数；
- α是Dirichlet分布（即Multinomial分布的先验分布）的参数。一般α由经验事先给定，p由观察到的文本中出现的词学习得到，表示文本中出现每个词的概率。

值得注意的是：词存在经验分布

- 人们长期使用语言决定了一种固定的语言使用分布
- 一些词用的比较多，而另外一些词使用的比较少，特别是在新闻领域：
  - Top 4 words: 10~15% word occurrences
  - Top 50 words: 35~40% word occurrences
- 这些高频词在一篇文章中可能出现频繁，但在另一篇文章中却非常罕见。

Zipf's Law

- rank * frequency ≈ constant $F(w)=\frac{C}{r(w)^{\alpha}} \quad \alpha\approx 1, C\approx 0.1$



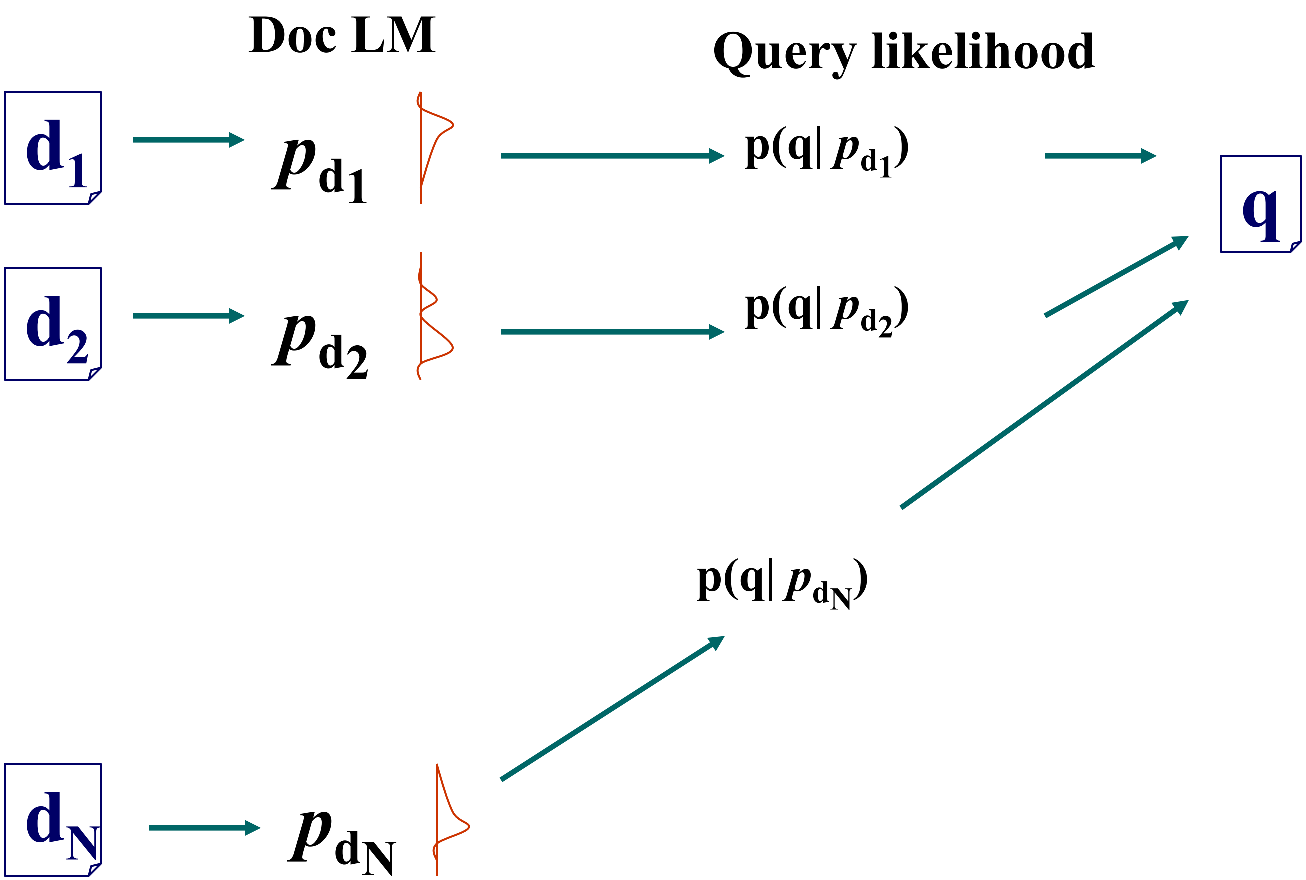
Generalized Zipf's law: $F(w)=\frac{C}{[r(w)+B]^{\alpha}}$ **Applicable in many domains**

哈佛大学的语言学家乔治·金斯利·齐夫（George Kingsley Zipf）于1949年发表的实验定律。

更一般的LM模型

- N元语言模型
  - 一般而言，$p(w_1 w_2 \ldots w_n)=p(w_1)p(w_2|w_1)\ldots p(w_n|w_1 \ldots w_{n-1})$
  - N元组: 只和其后的n-1个词有关系
  - E.g., 二元模型: $p(w_1 \ldots w_n)=p(w_1)p(w_2|w_1)\ p(w_3|w_2)\ \ldots p(w_n|w_{n-1})$
- 远相依语言模型Remote-dependence language models (e.g., Maximum Entropy model)
- 结构语言模型Structured language models (e.g., probabilistic context-free grammar) [Jelinek 98, Manning & Schutze 99, Rosenfeld 00]

一元语言模型的应用：通过查询对文档排序



信息提取的核心--LM Estimation

- Document ranking based on *query likelihood*

$$\log p(q \mid d) = \sum_{i=1}^{m} \log p(q_i \mid d) = \sum_{i=1}^{|V|} c(w_i, q) \log \boxed{p(w_i \mid d)}$$

$$where,\ \ q = q_1 q_2 \ldots q_m$$

Document language model

- Retrieval problem $\approx$ Estimation of $p(w_i|d)$
- Smoothing is an important issue, and distinguishes different approaches
- Many smoothing methods are available

对查询建模: 几种假设

- Multi-Bernoulli: 对查询中的词是否出现建模
  - $q = (x_1, \ldots, x_{|V|})$, $x_i = 1$ for presence of word w_i; $x_i = 0$ for absence

$$p(q=(x_1,\ldots,x_{|V|})\mid d) = \prod_{i=1}^{|V|} p(w_i = x_i \mid d) = \prod_{i=1, x_i=1}^{|V|} p(w_i = 1 \mid d) \prod_{i=1, x_i=0}^{|V|} p(w_i = 0 \mid d)$$

  - Parameters: $\{p(w_i=1|d), p(w_i=0|d)\}$ $p(w_i=1|d) + p(w_i=0|d) = 1$
- Multinomial (Unigram LM): 对词频建模
  - $q = q_1, \ldots q_m$, where q_j is a query word

$$p(q = q_1 \ldots q_m \mid d) = \prod_{j=1}^{m} p(q_j \mid d) = \prod_{i=1}^{|V|} p(w_i \mid d)^{c(w_i, q)}$$

  - $c(w_i,q)$ is the count of word w_i in query q
  - Parameters: $\{p(w_i|d)\}$ $p(w_1|d) + \ldots p(w_{|v|}|d) = 1$

[Ponte & Croft 98] **uses Multi-Bernoulli; most other work uses multinomial**

Multinomial seems to work better [Song & Croft 99, McCallum & Nigam 98,Lavrenko 04]

Difficulty in Feedback with Query Likelihood

- Traditional query expansion [Ponte 98, Miller et al. 99, Ng 99]
  - Improvement is reported, but there is a conceptual inconsistency
  - What's an expanded query, a piece of text or a set of terms?
- Avoid expansion
  - Query term reweighting [Hiemstra 01, Hiemstra 02]
  - Translation models [Berger & Lafferty 99, Jin et al. 02]
  - Only achieving limited feedback
- Doing relevant query expansion instead [Nallapati et al 03]
- The difficulty is due to the lack of a query/relevance model
- The difficulty can be overcome with alternative ways of using LMs for retrieval (e.g., relevance model [Lavrenko & Croft 01] , Query model estimation [Lafferty & Zhai 01b; Zhai & Lafferty 01b])

3.pLSA: Motivation

What did people say in their blog articles about "Hurricane Katrina"?

Query = "Hurricane Katrina"   飓风 卡特里娜

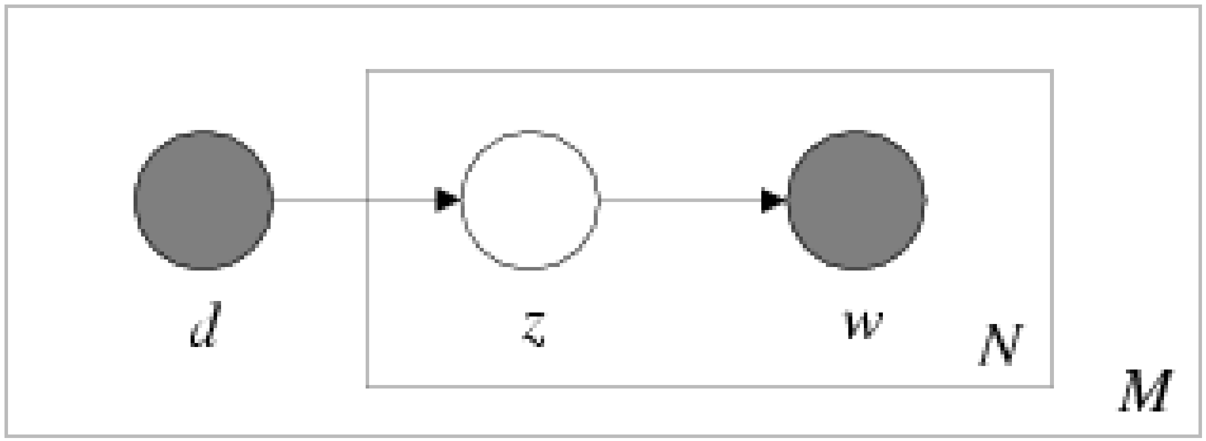
Results:

Government Response	New Orleans	Oil Price	Praying and Blessing	Aid and Donation	Personal
bush 0.071	city 0.063	price 0.077	god 0.141	donate 0.120	i 0.405
president 0.061	orleans 0.054	oil 0.064	pray 0.047	relief 0.076	my 0.116
federal 0.051	new 0.034	gas 0.045	prayer 0.041	red 0.070	me 0.060
government 0.047	louisiana 0.023	increase 0.020	love 0.030	cross 0.065	am 0.029
fema 0.047	flood 0.022	product 0.020	life 0.025	help 0.050	think 0.015
administrate 0.023	evacuate 0.021	fuel 0.018	bless 0.025	victim 0.036	feel 0.012
response 0.020	storm 0.017	company 0.018	lord 0.017	organize 0.022	know 0.011
brown 0.019	resident 0.016	energy 0.017	jesus 0.016	effort 0.020	something 0.007
blame 0.017	center 0.016	market 0.016	will 0.013	fund 0.019	guess 0.007
governor 0.014	rescue 0.012	gasoline 0.012	faith 0.012	volunteer 0.019	myself 0.006

Probabilistic Latent Semantic Analysis/Indexing (pLSA/pLSI) [Hofmann 99]

- Mix k multinomial distributions to generate a document
- Each document has a potentially different set of mixing weights which captures the topic coverage
- When generating words in a document, each word may be generated using a DIFFERENT multinomial distribution (this is in contrast with the document clustering model where, once a multinomial distribution is chosen, all the words in a document would be generated using the same model)
- We may add a background distribution to "attract" background words

主题模型Topic Model / Probabilistic LSI



$$p(d, w_n) = p(d) \sum_z p(w_n \mid z) p(z \mid d)$$

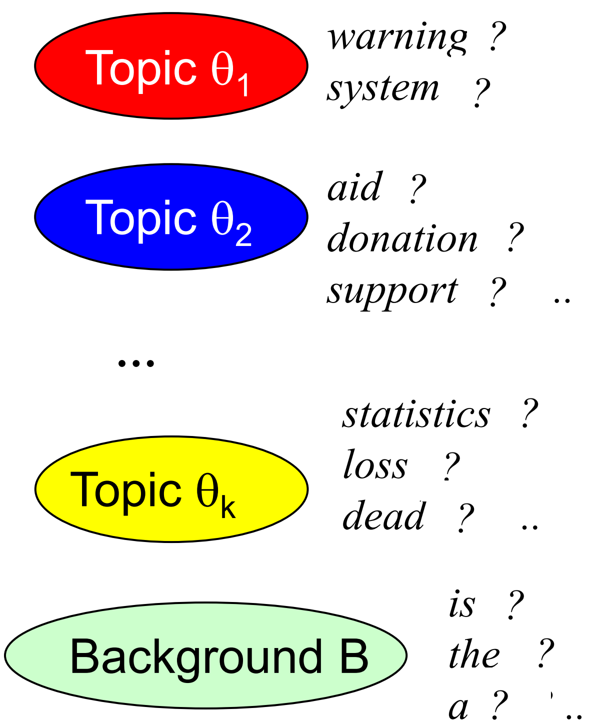
- d is *a localist representation* of (trained) documents
- LDA provides a *distributed representation*

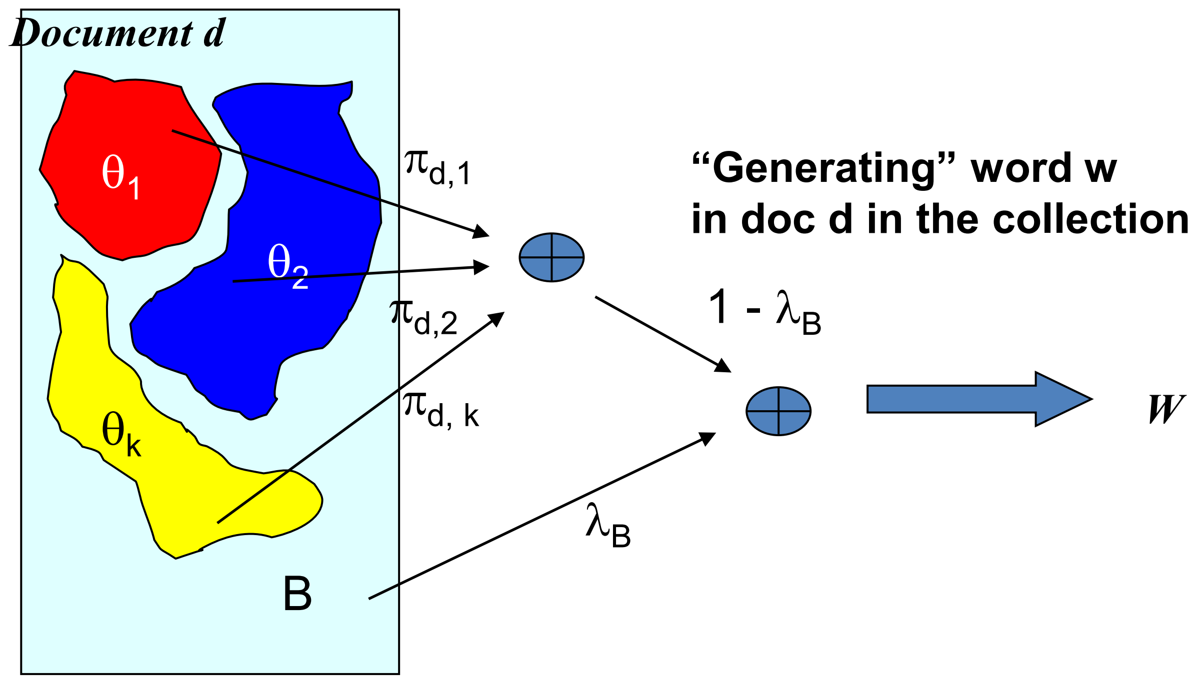
2.2 PLSA as a Mixture Model

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

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

PLSA是一种混合模型，需要使用两层概率对整个样本空间建模





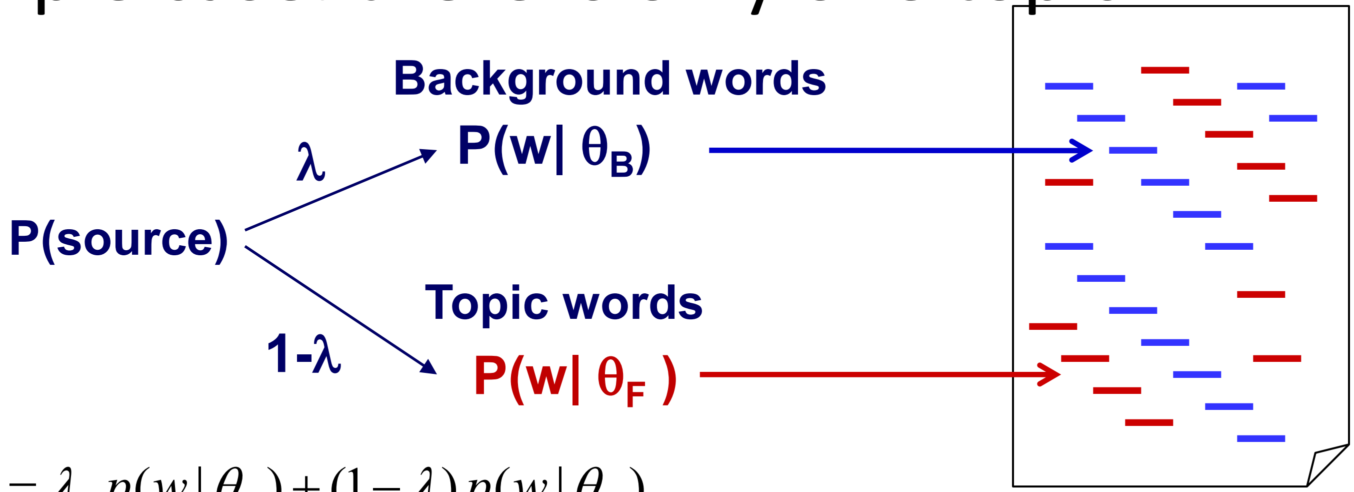
Parameters:

λ_B=noise-level (manually set)

θ's and π's are estimated with Maximum Likelihood

Special Case: Model-based Feedback

- Simple case: there is only one topic



$$p_d(w) = \lambda_B p(w|\theta_B) + (1-\lambda) p(w|\theta_F)$$

$$\log p(d) = \sum_{w \in V} c(w,d) \log[\lambda_B p(w|\theta_B) + (1-\lambda) p(w|\theta_F)]$$

Maximum Likelihood: $\theta_F = \arg\max_{\theta} \sum_{d} \log p(d)$

What about there are *k* topics?

How to Estimate θ_j: EM Algorithm

Known Background $p(w \mid B)$

the 0.2
a 0.1
we 0.01
to 0.02
…

Unknown topic model $p(w|\theta_1)$=?

"Text mining"

…
text =?
mining =?
association =?
word =?
…

…

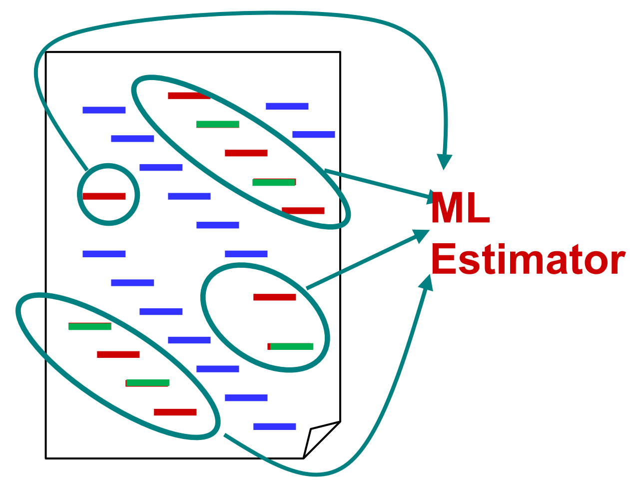
Unknown topic model $p(w|\theta_2)$=?

"information retrieval"

…
information =?
retrieval =?
query =?
document =?
…

Suppose, we know the identity of each word ...

Observed Doc(s)



A General Introduction to EM

Data: X (observed) + H(hidden) Parameter: θ

"Incomplete" likelihood: $L(\theta) = \log p(X|\theta)$
"Complete" likelihood: $L_c(\theta) = \log p(X,H|\theta)$

EM tries to iteratively maximize the incomplete likelihood:

Starting with an initial guess $\theta^{(0)}$,

1. E-step: compute the expectation of the complete likelihood

$$Q(\theta;\theta^{(n-1)}) = E_{\theta^{(n-1)}}[L_c(\theta) \mid X] = \sum_{h_i} p(H = h_i \mid X, \theta^{(n-1)}) \log P(X, h_i)$$

2. M-step: compute $\theta^{(n)}$ by maximizing the Q-function

$$\theta^{(n)} = \arg\max_\theta Q(\theta;\theta^{(n-1)}) = \arg\max_\theta \sum_{h_i} p(H = h_i \mid X, \theta^{(n-1)}) \log P(X, h_i)$$

Convergence Guarantee

Goal: maximizing "Incomplete" likelihood: $L(\theta) = \log p(X|\theta)$

I.e., choosing $\theta^{(n)}$, **so that** $L(\theta^{(n)}) - L(\theta^{(n-1)}) \geq 0$

Note that, since $p(X,H|\theta) = p(H|X,\theta)\, P(X|\theta)$, $L(\theta) = L_c(\theta) - \log p(H|X,\theta)$

$$L(\theta^{(n)}) - L(\theta^{(n-1)}) = L_c(\theta^{(n)}) - L_c(\theta^{(n-1)}) + \log [p(H|X,\theta^{(n-1)}) / p(H|X,\theta^{(n)})]$$

Taking <u>expectation w.r.t.</u> <u>$p(H|X,\theta^{(n-1)})$,</u>

$$\underline{L(\theta^{(n)}) - L(\theta^{(n-1)})} = \underline{\mathbf{Q(\theta^{(n)}; \theta^{(n-1)}) - Q(\theta^{(n-1)}; \theta^{(n-1)})}} + \underline{\mathbf{D(p(H|X,\theta^{(n-1)}) || p(H|X,\theta^{(n)}))}}$$

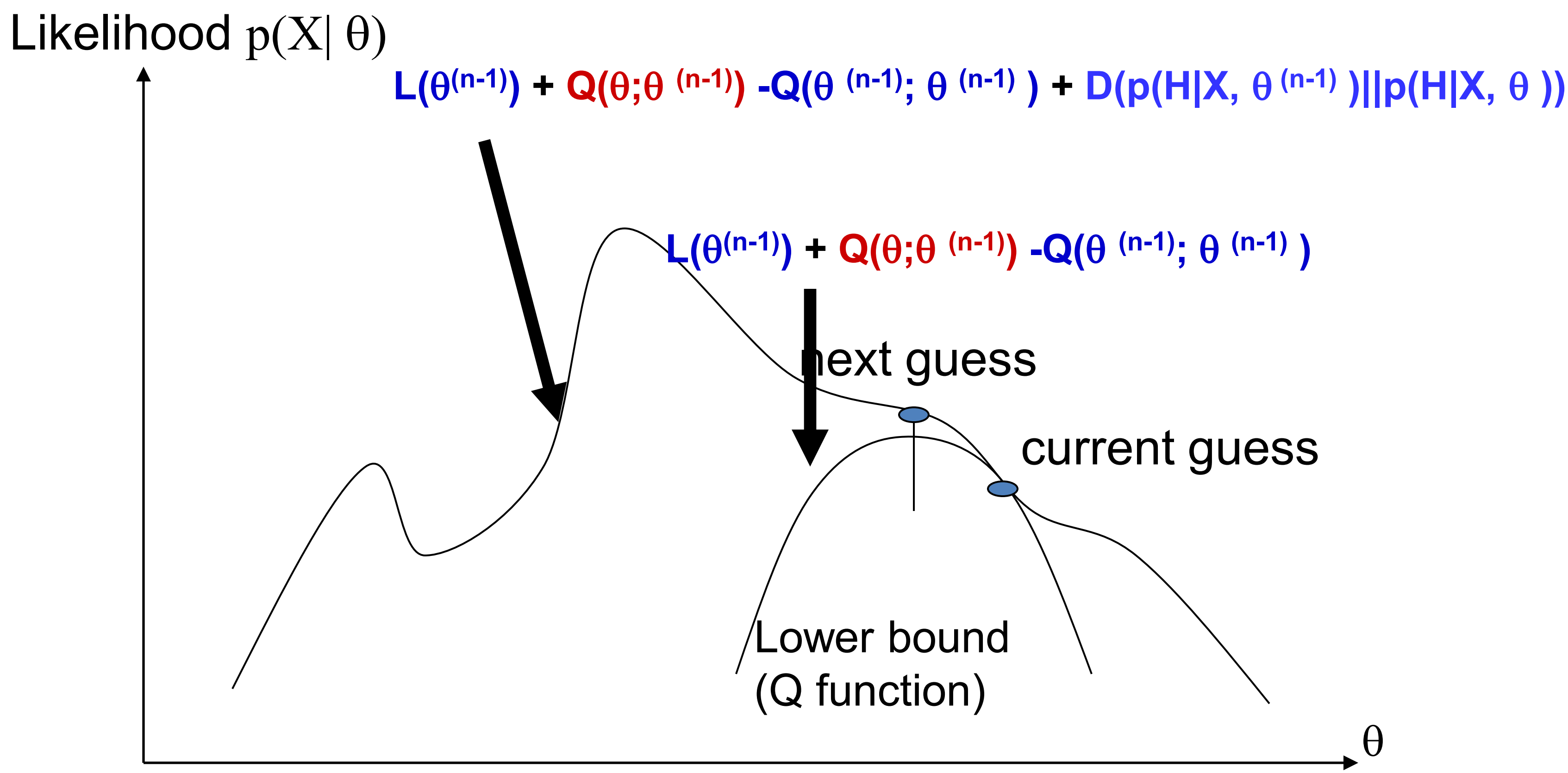
Doesn't contain H

EM chooses $\theta^{(n)}$ to maximize Q

KL-divergence, always non-negative

Therefore, $L(\theta^{(n)}) \geq L(\theta^{(n-1)})$!

Another way of looking at EM



E-step = computing the lower bound
M-step = maximizing the lower bound

Parameter Estimation

E-Step:
Word w in doc d is generated
- **from cluster j**
- **from background**

Application of Bayes rule

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

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

M-Step:
Re-estimate
- **mixing weights**
- **cluster LM**

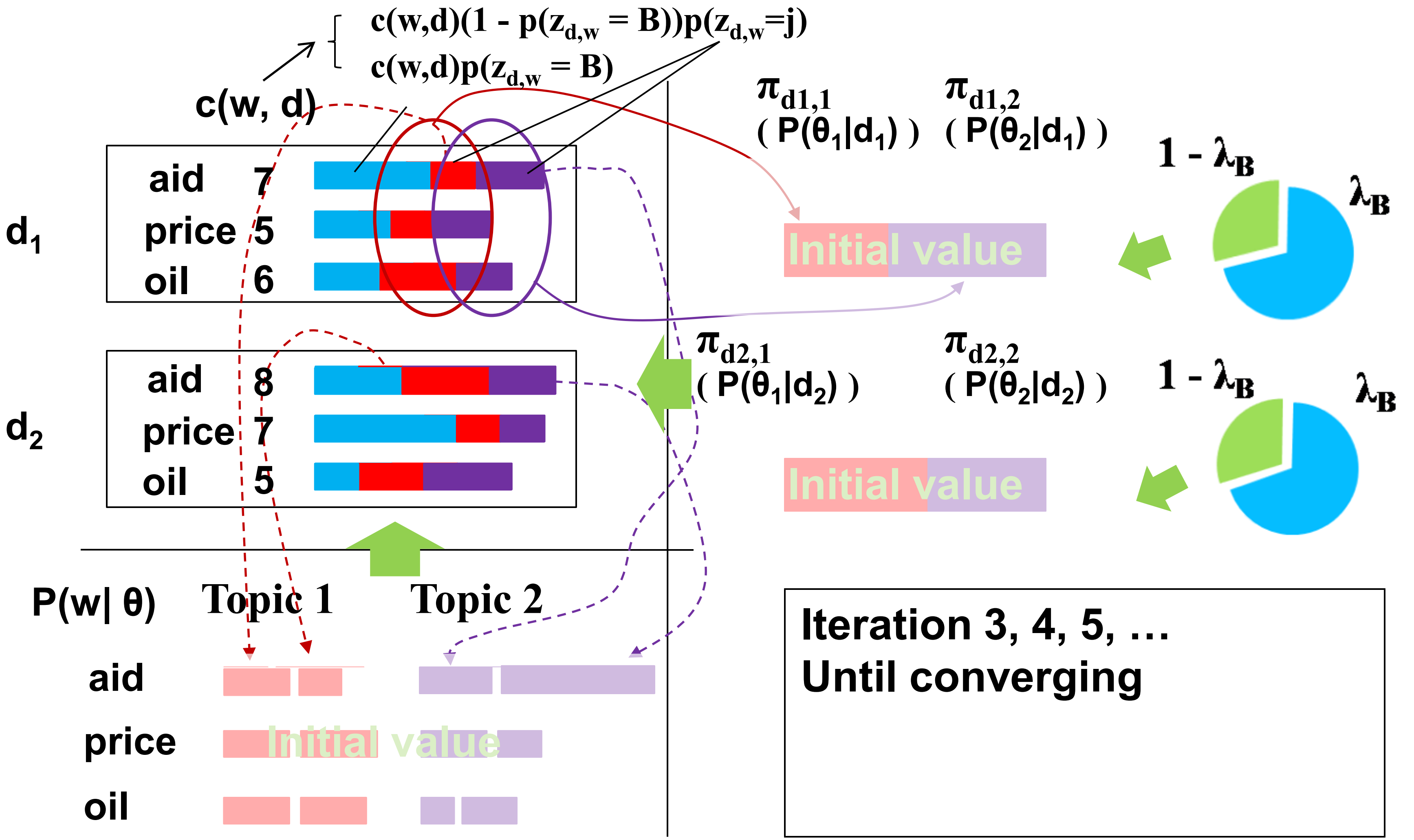
$$\pi_{d,j}^{(n+1)}=\frac{\sum_{w\in V}c(w,d)(1-p(z_{d,w}=B))p(z_{d,w}=j)}{\sum_{j'}\sum_{w\in V}c(w,d)(1-p(z_{d,w}=B))p(z_{d,w}=j')}$$

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

Sum over all docs
(in multiple collections)
m = 1 if one collection

Fractional counts contributing to
- **using cluster j in generating d**
- **generating w from cluster j**

How the Algorithm Works



Iteration 3, 4, 5, …
Until converging

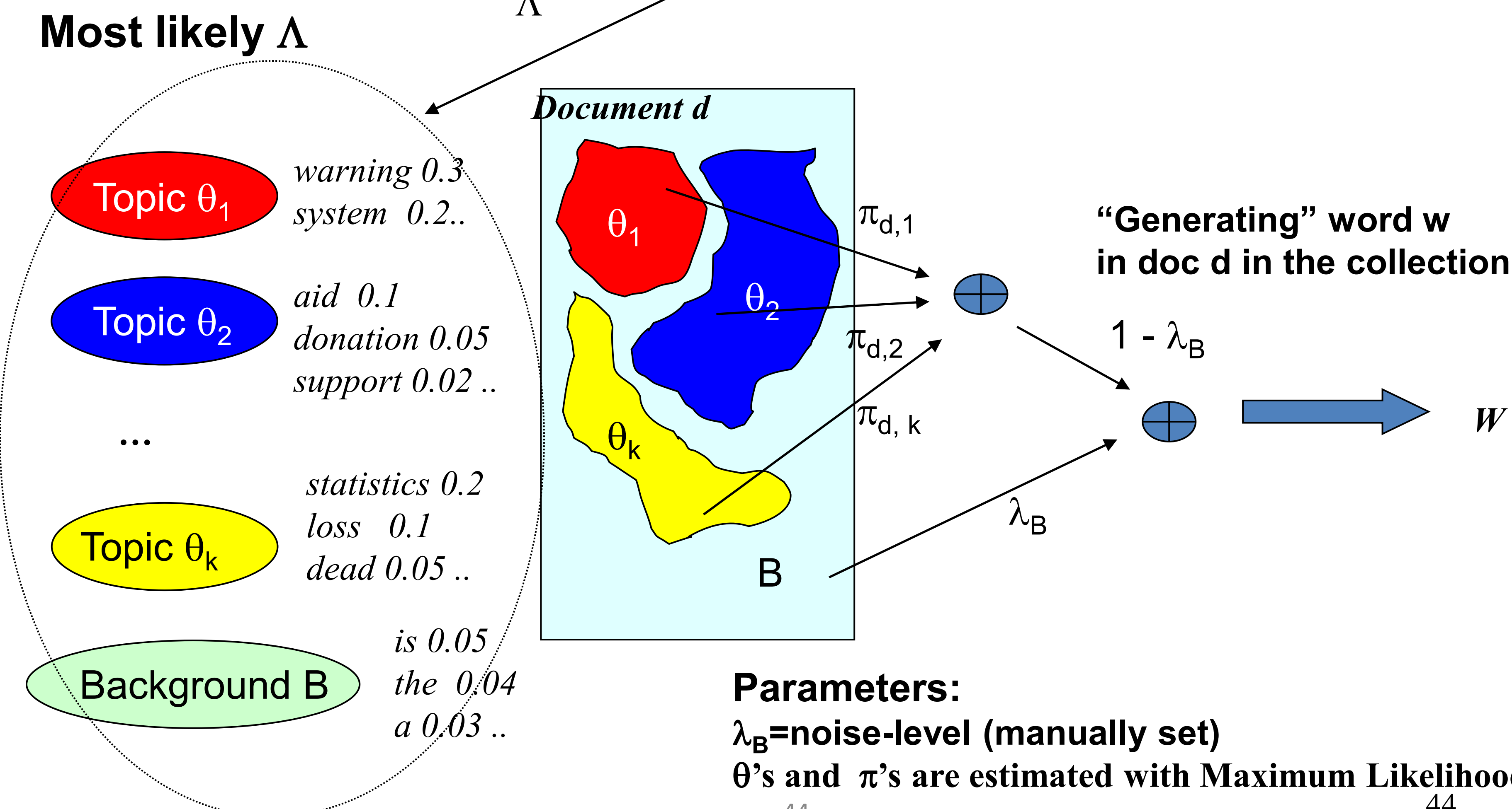
PLSA with Prior Knowledge

- There are different ways of choosing aspects (topics)
  - Google = Google News + Google Map + Google scholar, …
  - Google = Google US + Google France + Google China, …
- Users have some domain knowledge in mind, e.g.,
  - We expect to see “retrieval models” as a topic in IR.
  - We want to show the aspects of “history” and “statistics” for Youtube
- A flexible way to incorporate such knowledge as priors of PLSA model
- In Bayesian, it’s your “belief” on the topic distributions

43

Adding Prior

$$\Lambda^* = \arg\max_{\Lambda} \; p(\Lambda)\, p(Data \mid \Lambda)$$



Parameters:

λ_B=noise-level (manually set)

θ's and π's are estimated with Maximum Likelihood

44

Adding Prior as Pseudo Counts

Known Background $p(w \mid B)$

the 0.2
a 0.1
we 0.01
to 0.02
…

Unknown topic model $p(w|\theta_1)$=?

"Text mining"

…
text =?
mining =?
association =?
word =?
…

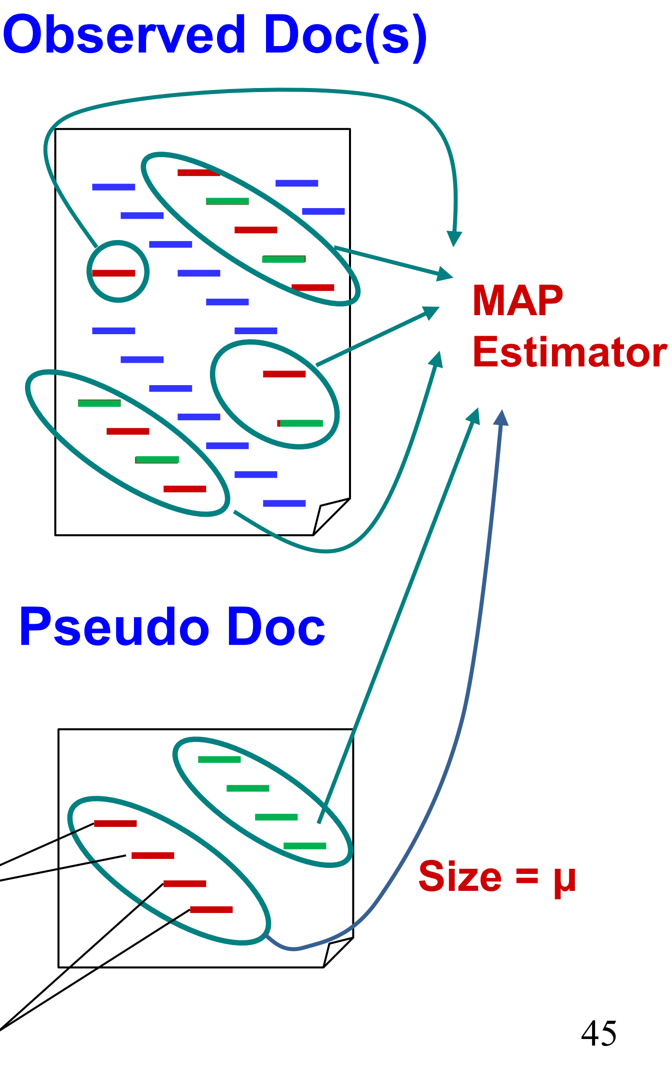
…

Unknown topic model $p(w|\theta_2)$=?

"information retrieval"

…
information =?
retrieval =?
query =?
document =?
…

Suppose, we know the identity of each word ...



Maximum A Posterior (MAP) Estimation

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

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

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

$$p^{(n+1)}(w\,|\,\theta_j)=\frac{\sum_{i=1}^{m}\sum_{d\in C_i}c(w,d)(1-p(z_{d,w}=B))\,p(z_{d,w}=j)\;+\mu p(w|\theta'_j)}{\sum_{w'\in V}\sum_{i=1}^{m}\sum_{d\in C_i}c(w',d)(1-p(z_{d,w'}=B))\,p(z_{d,w'}=j)\;+\mu}$$

Pseudo counts of w from prior θ'

Sum of all pseudo counts

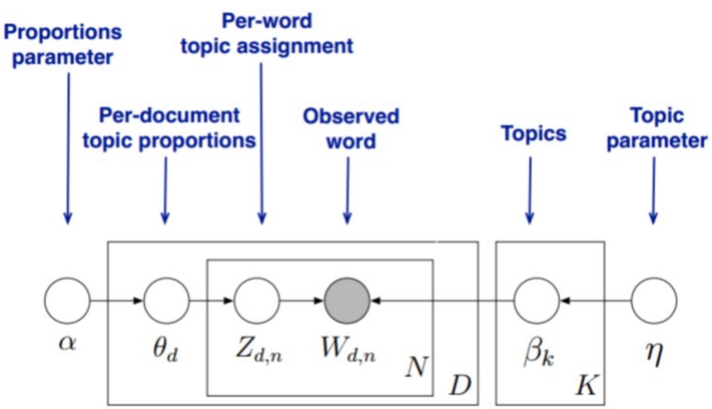
What if μ=0? What if μ=+∞?

PLSA的优点缺点分析

- **优点：**PLSA可以解决了同义词和多义词的问题，利用了强化的期望最大化算法（EM）训练隐含类（潜在类）。而且相对了LSA，有了坚实的统计学基础。

- **缺点：**随着document和term 个数的增加，pLSA模型也线性增加，变得越来越庞大，也就是说pLSA中训练参数的值会随着文档的数目线性递增。而且，pLSA可以生成其所在数据集的的文档的模型，但却不能生成新文档的模型。

2.2.Basic Topic Model: LDA



"Arts"	"Budgets"	"Children"	"Education"
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

The following slides about LDA are taken from Michael C. Mozer's course lecture
http://www.cs.colorado.edu/~mozer/courses/ProbabilisticModels/
以一定的概率取主题，再以一定的额概率选取主题下的某个单词，不断重复两步，最终生成文章

2.2 LDA：概率主题模型：隐狄利克雷分布（Latent Dirichlet Allocation，简称LDA）LDA: Motivation

- "Documents have no generative probabilistic semantics"
    - i.e., document is just a symbol
- Model has many parameters
    - linear in number of documents
    - need heuristic methods to prevent overfitting
- Cannot generalize to new documents
- 需要根据一篇文档确定一篇文档的主题分布

•是一种主题模型，它可以将文档集中每篇文档的主题以概率分布的形式给出，从而通过分析一些文档抽取出它们的主题（分布）出来后，便可以根据主题（分布）进行主题聚类或文本分类。同时，它是一种典型的词袋模型，即一篇文档是由一组词构成，词与词之间没有先后顺序的关系。

•词袋模型Vocabulary of |V| words

•Document is a collection of words from vocabulary.

- N words in document
- $w = (w_1, \ldots, w_N)$

•*Latent topics*

- *random variable z, with values 1, ..., k*

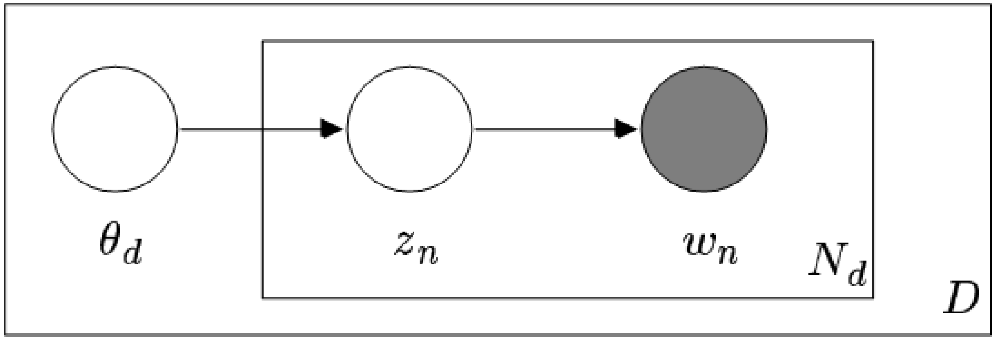
•Like topic model, document is generated by sampling a topic from a mixture and then sampling a word from a mixture.

- But topic model assumes a fixed mixture of topics (multinomial distribution) for each document.
- LDA assumes a *random mixture of topics (Dirichlet distribution) for each topic.*

LDA可以分为以下5个步骤

- 一个函数：
  - gamma函数：独立同分布 $(X_1,\ldots X_n)\sim U(0,1)$,$x_{(k)}$的分布是$beta(k,n-k+1)$
- 四个分布：
  - 二项分布、多项分布、beta分布、Dirichlet分布
- 一个概念和一个理念：
  - 共轭先验和贝叶斯框架
- 两个模型：
  - pLSA、LDA
- 一个采样：
  - Gibbs采样

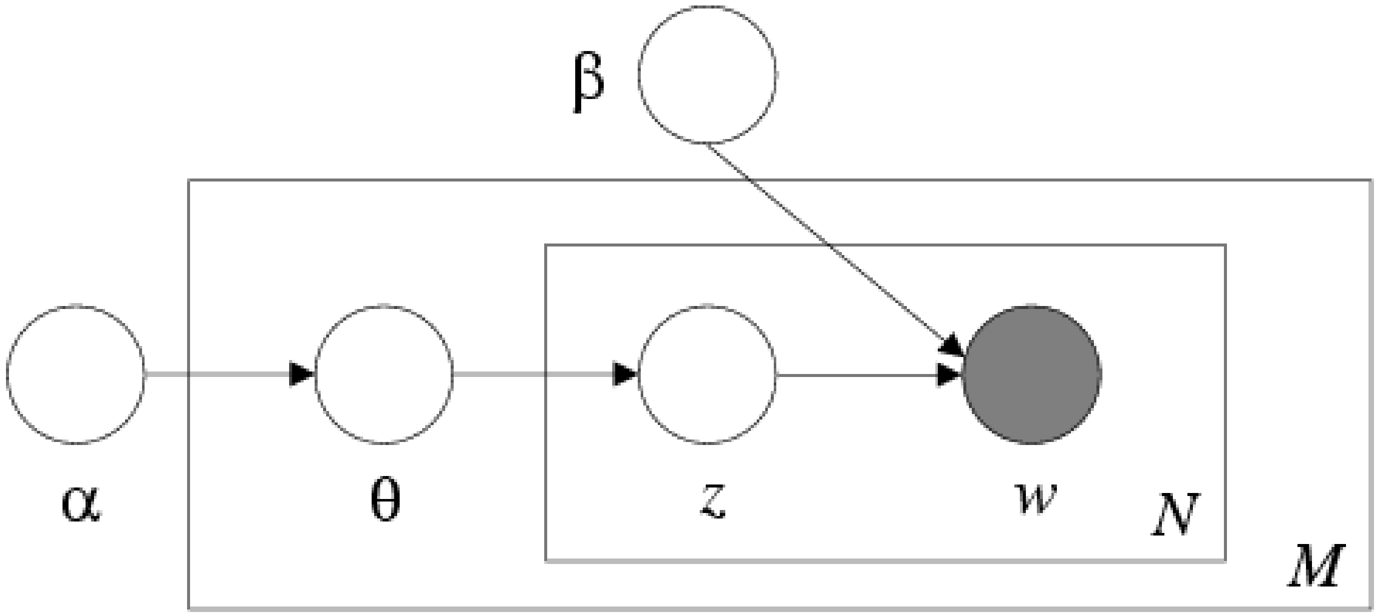
Generative Model



- "Plates" indicate looping structure
  - Outer plate replicated for each document
  - Inner plate replicated for each word
  - Same conditional distributions apply for each replicate
- Document probability

$$p(\mathbf{w}) = \int_\theta \left(\prod_{n=1}^{N} \sum_{z_n=1}^{k} p(w_n | z_n; \beta) p(z_n | \theta) \right) p(\theta; \alpha) d\theta$$

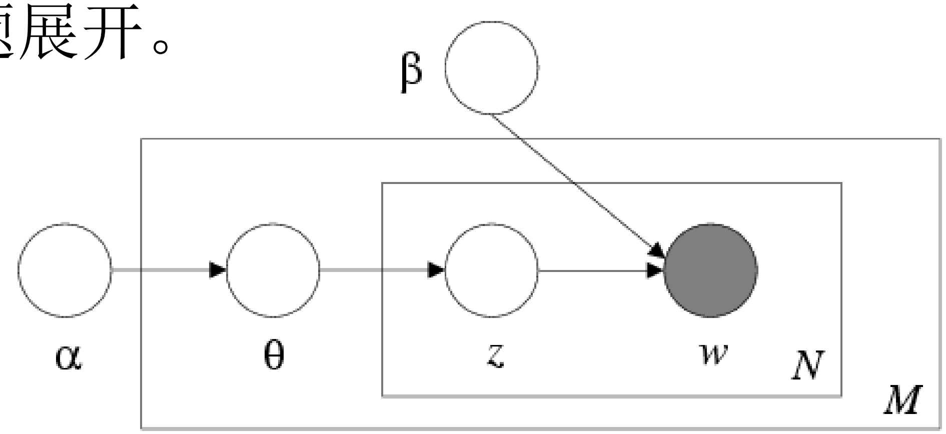
Fancier Version



$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} \cdots \theta_k^{\alpha_k - 1}$$

Inference

LDA的目标是给定一篇文章，推测这篇文章的主题分布，也就是要找到每一篇文档的主题分布和每一个主题中词的分布。在LDA模型中，我们需要先假定一个主题数目K，所有的分布都基于K个主题展开。



$$p(\theta,\mathbf{z}\mid\mathbf{w},\alpha,\beta)=\frac{p(\theta,\mathbf{z},\mathbf{w}|\alpha,\beta)}{p(\mathbf{w}|\alpha,\beta)}$$

$$p(\theta,\mathbf{z},\mathbf{w}|\alpha,\beta)=p(\theta|\alpha)\prod_{n=1}^{N}p(z_n|\theta)p(w_n|z_n,\beta)$$

$$p(\mathbf{w}|\alpha,\beta)=\int p(\theta|\alpha)\left(\prod_{n=1}^{N}\sum_{z_n}p(z_n|\theta)p(w_n|z_n,\beta)\right)d^k\theta$$

Inference

- In general, this formula is intractable:

$$p(\mathbf{w}|\alpha,\beta) = \int p(\theta|\alpha)\left(\prod_{n=1}^{N}\sum_{z_n} p(z_n|\theta)\,p(w_n|z_n,\beta)\right)d^k\theta$$

- Expanded version:

1 if w_n is the j'th vocab word

$$p(\mathbf{w}\,|\,\alpha,\beta) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)}\int\left(\prod_{i=1}^{k}\theta_i^{\alpha_i-1}\right)\left(\prod_{n=1}^{N}\sum_{i=1}^{k}\prod_{j=1}^{V}(\theta_i\beta_{ij})^{w_n^j}\right)d\theta$$

Variational Approximation

- Computing log likelihood and introducing Jensen's inequality: log(E[x]) >= E[log(x)]

$$\begin{aligned}\log p(\mathbf{w};\alpha,\beta) &= \log \int_\theta \sum_{\mathbf{z}} p(\mathbf{w}|\mathbf{z};\beta)p(\mathbf{z}|\theta)p(\theta;\alpha)\frac{q(\theta,\mathbf{z};\gamma,\phi)}{q(\theta,\mathbf{z};\gamma,\phi)}d\theta \\ &\geq \mathrm{E}_q[\log p(\mathbf{w}|\mathbf{z};\beta)+\log p(\mathbf{z}|\theta)+\log p(\theta;\alpha)-\log q(\theta,\mathbf{z};\gamma,\phi)]\end{aligned}$$

- Find variational distribution *q such that the above equation is computable.*
  - *q parameterized by γ and* φ_n
  - *Maximize bound with respect to γ and* φ_n *to obtain best approximation to p(w | α, β)*
  - *Lead to variational EM algorithm*
- *Sampling algorithms (e.g., Gibbs sampling) are also common*

Summary: PLSA vs. LDA

- LDA adds a Dirichlet distribution on top of PLSA to regularize the model
- Estimation of LDA is more complicated than PLSA
- LDA is a generative model, while PLSA isn't
- PLSA is more likely to over-fit the data than LDA
- Which one to use?
  - If you need generalization capacity, LDA
  - If you want to mine topics from a collection, PLSA may be better (we want overfitting!)

聚类在自然语言中的应用

- **探测数据分析（exploratory data analysis）**
  – 例如词性标注，将相似的词作为同一种词性，对前置词比较有效
  – 对this和the 这种语法语义特征不一致的词，不总分在一组的词不适合
- **概化（generalization）**
  – 等价类，可以使用相同的上下文环境，解决数据稀疏问题
  – 同时聚类是学习的一种方法（推理Friday 的前置词）

作业（202012.17）

1. 请针对已收集的文档通过与查询Q1匹配，进行匹配排序。请考虑文档的长度和停词等因素：

   Q1=“水上赛事因天气因素被中断”

2. 请与Q2=“水上赛事因风力级别过高被中断”的词频分布比较，最为匹配的是哪些文档。
3. 请根据2中匹配程度较高的文档作为训练文档，提炼主题模型，指出这些文档中主要有几个主题，每个主题的结构比例和主要关键词的分布构成是怎样的。