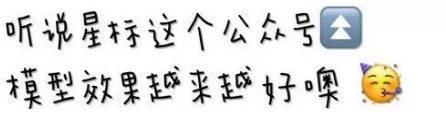
LDA主题模型 | 原理详解与代码实战

原创 kaiyuan NewBeeNLP 2020-04-09

收录于话题
#自然语言处理

55个



很久之前的LDA笔记整理，包括算法原理介绍以及简单demo实践，主要参考自July老师的<通俗理解LDA主题模型>。

1、写在前面

在机器学习领域，关于LDA有两种含义，一是**「线性判别分析（Linear Discriminant Analysis）」**，是一种经典的降维学习方法；一是本文要讲的**「隐含狄利克雷分布（Latent Dirichlet Allocation）」**，是一种概率主题模型，主要用来文本分类，在NLP领域有重要应用。LDA由Blei，David M.、Ng，Andrew Y.、Jordan于2003年提出，用来推测文档的主题分布。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后，便可以根据主题分布进行主题聚类或文本分类。

2、数学知识

第一次接触LDA的同学肯定是一头雾水的，因为相比于其他的机器学习算法，LDA模型涉及到很多数学知识与公式，这也许是LDA晦涩难懂的原因。在本小节中会介绍LDA中所需要的数学应用，对后面进一步理解LDA模型打好基础。

要理解LDA， 涉及的先验知识有以下几种：

- 一个函数：Gamma函数
- 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和LDA

- 一个采样：Gibbs sampling

2.1 Gamma函数

Gamma函数的定义为：

$$\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$$

对上式进行分部积分之后可以发现Gamma函数具有如下的性质：

$$\Gamma(x+1) = x\Gamma(x)$$

可知Gamma函数可以看成是阶乘函数在实数上的推广：

$$\Gamma(n) = (n-1)!$$

2.2二项分布

要解释二项分布，首先要搞清楚**「伯努利分布」**（又称为两点分布或者0-1分布），它是一个离散型的随机分布，其中的随机变量只有两种取值，{0,1}。而二项分布就是对伯努利分布重复n次。举个栗子，把投掷一次硬币这个试验认为是伯努利分布，则投掷n次硬币的试验就可以被认为是二项分布。二项分布的概率密度公式为：

$$P(K=k) = \binom{n}{k} p^k (1-p)^{n-k}$$

2.3 多项分布

从二项分布到多项分布，只是从随机变量的取值由两种扩展为多维。多项分布是指单次实验中的随机变量的取值不再是0-1的，而是有多重离散值（1,2,3....当然也不一定是整数）。举个栗子，投掷有六个面的骰子试验。多项分布的概率密度公式为：

$$P(x_1, x_2, \ldots, x_k; n, p_1, p_2, \ldots, p_k) = \frac{n!}{x_1! \ldots x_k!} {p_1}^{x_1} \ldots {p_k}^{x_k}$$

其中x1，x2...是指随机变量的取值，而p1，p2...是指取到相对应随机变量的概率。

2.4 共轭先验分布

在贝叶斯概率理论中，如果**「后验概率P(θ|x)和先验概率p(θ)满足同样的分布律」**，那么，先验分布和后验分布被叫做共轭分布，同时，先验分布叫做似然函数的共轭先验分布。

$$P(\theta|x) = \frac{P(\theta, x)}{P(x)} \tag{1}$$

共轭的意思是，以Beta分布和二项式分布为例，数据符合二项分布的时候，参数的先验分布和后验分布都能保持Beta分布的形式，这种形式不变的好处是，我们能够在先验分布中赋予参数很明确的物理意义，这个物理意义可以延续到后续分布中进行解释，同时从先验变换到后验过程中从数据中补充的知识也容易有物理解释。

2.5 Beta分布

对于参数a>0,b>0，取值范围为[0,1]的随机变量x的概率密度函数为：

$$f(x;\alpha,\beta)=\frac{1}{B(\alpha,\beta)}x^{\alpha-1}(1-x)^{\beta-1} \quad (2)$$

其中，

$$\frac{1}{B(\alpha,\beta)}=\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \quad (3)$$

对比二项分布公式可以发现：Beta分布是二项分布的共轭先验分布。

2.6 Dirichlet分布

Dirichlet分布是Beta分布在高维上的推广，其公式为：

$$\frac{1}{B(\alpha,\beta)}=\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \quad (4)$$

其中：

$$B(\alpha)=\frac{\prod_{i=1}^{k}\Gamma(\alpha^i)}{\Gamma(\sum_{i=1}^{k}\alpha^i)},\sum_{i=1}^{k}x^i=1 \quad (5)$$

根据Beta分布、二项分布、Dirichlet分布、多项式分布的公式，我们可以得出结论：Beta分布是二项式分布的共轭先验分布，而狄利克雷(Dirichlet)分布是多项式分布的共轭分布。

2.7 MCMC和Gibbs sampling

在现实应用中，我们很多时候很难精确求出精确的概率分布，常常采用近似推断方法。近似推断方法大致可分为两大类：第一类是采样(Sampling)，通过使用随机化方法完成近似；第二类是使用确定性近似完成近似推断，典型代表为变分推断(variational inference).

3、文本建模

ok，介绍完数学知识，下面来到了本文的重点，LDA模型。在这之前，会循序渐进地介绍几个基础模型：**「Unigram model」**、**「mixture of unigrams model」**，以及跟LDA最为接近的**「pLSA」**模型。

为了后续描述方便，首先定义一些变量：

- w表示词，V表示所有单词的个数（固定值）
- z表示主题，k是主题的个数（预先给定，固定值）
- $D=(\mathbf{w}_1,\cdots,\mathbf{w}_M)$表示语料库，其中$M$是语料库中的文档数（固定值）
- $\mathbf{w}=(w_1,w_2,\cdots,w_N)$表示文档，其中$N$表示一个文档中的词数（随机变量）

3.1 Unigram Model

在Unigram Model中，我们采用**「词袋模型」**，假设了文档之间相互独立，文档中的词汇之间相互独立。假设我们的词典中一共有 V个词 ，那么最简单的 Unigram Model 就是认为上帝是按照如下的游戏规则产生文本的：

1. 上帝只有一个骰子，这个骰子有V面，每个面对应一个词，各个面的概率不一；
2. 每抛掷一次骰子，抛出的面就对应的产生一个词；如果一篇文档中N个词，就独立的抛掷n次骰子产生n个词；

用公式来表示的话也就是

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

3.2 Mixture of Unigram Model

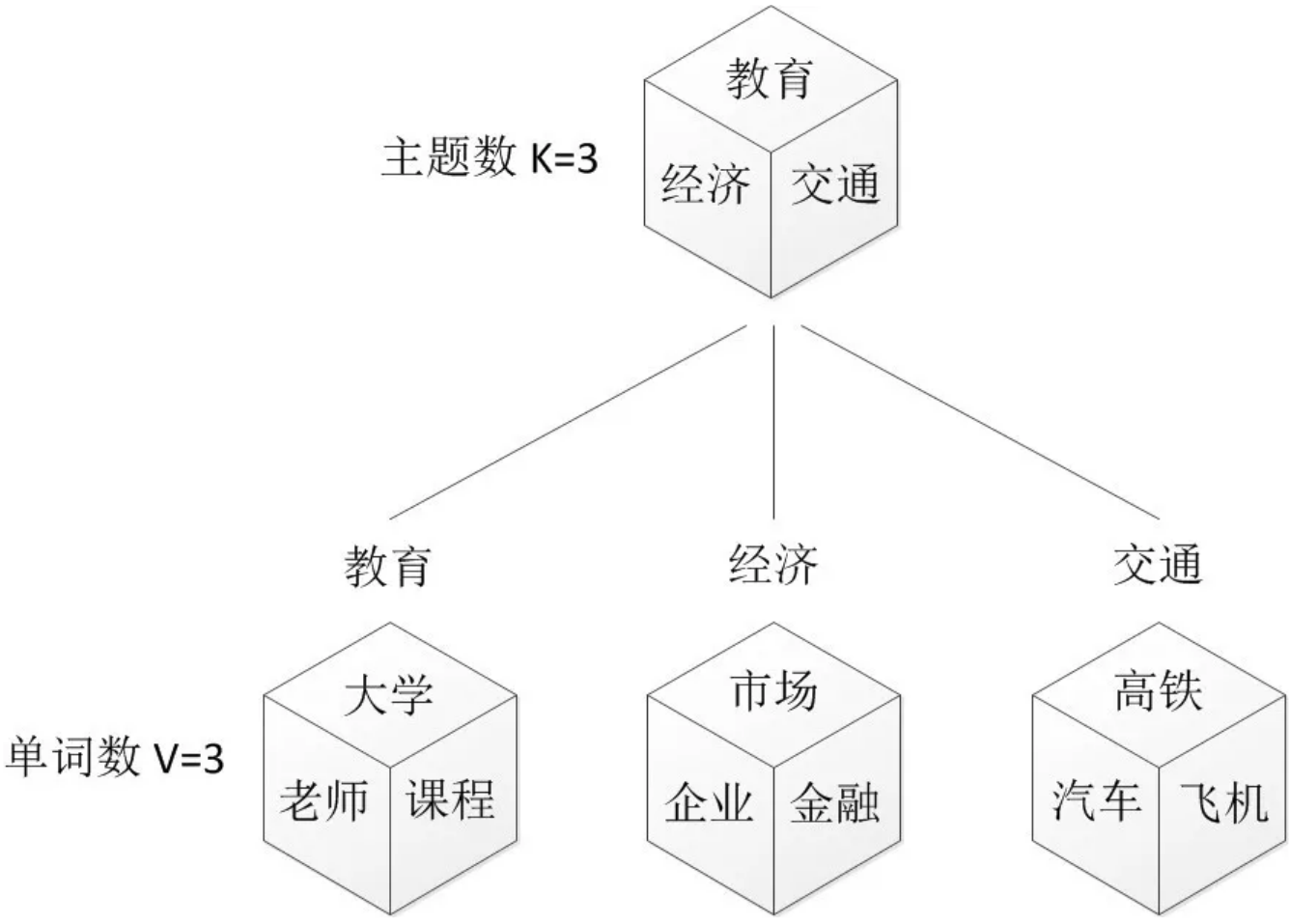
相比于unigram model，mixture of unigram model引入了一个**“主题”**参数作为中间量来链接文档和词语。

该模型的生成过程是：给某个文档先选择一个主题，再根据该主题生成文档，该文档中的所有词都来自一个主题。假设主题有$z_1, \ldots, z_k$，生成文档的概率为：

$$p(\mathbf{w}) = p(z_1)\prod_{n=1}^{N} p(w_n|z_1) + \cdots + p(z_k)\prod_{n=1}^{N} p(w_n|z_k) = \sum_{z} p(z)\prod_{n=1}^{N} p(w_n|z)$$

3.3 pLSA

在上面的Mixture of unigrams model中，我们假定一篇文档只有一个主题生成，可实际中，一篇文章往往有**「多个主题」**，只是这多个主题各自在文档中出现的概率大小不一样。比如介绍一个国家的文档中，往往会分别从教育、经济、交通等多个主题进行介绍。那么在pLSA中，文档是怎样被生成的呢？



我们可以用以上的骰子模型来模拟PLSA生成一片文档的过程：

1. 现有两种类型的骰子，一种是doc-topic骰子，每个doc-topic骰子有K个面，每个面一个topic的编号；一种是topic-word骰子，每个topic-word骰子有V个面，每个面对应一个词；
2. 现有K个topic-word骰子（对应doc-topic骰子的K个面），每个骰子有一个编号，编号从1到K；
3. 生成每篇文档之前，先为这篇文章制造一个特定的doc-topic骰子，重复如下过程生成文档中的词：
    - 投掷这个doc-topic骰子，得到一个topic编号z；
    - 选择K个topic-word骰子中编号为z的那个，投掷这个骰子，得到一个词；

重复至完成一篇文档所需的词语数。在这个过程中，我们并未关注词和词之间的出现顺序，所以pLSA是一种词袋方法。

在这里，我们定义：

- $P(d_i)$表示海量文档中某篇文档被选中的概率。
- $P(w_j|d_i)$表示词在给定文档中出现的概率。
- $P(z_k|d_i)$表示具体某个主题在给定文档下出现的概率。
- $P(w_j|z_k)$表示具体某个词在给定主题下出现的概率，与主题关系越密切的词，其条件概率越大。

利用上述定义好的概率，我们可以按照如下的步骤得到‘文档-词语’的生成模型：

1. 按照概率选择一篇文档
2. 选定文档后，从主题分布中按照概率选择一个隐含的主题类别
3. 选定后，从词分布中按照概率选择一个词

简而言之，pLSA的生成文档过程可以理解为**「先选定文档生成主题，再确定主题生成词语」**。

但是如果我们现在的情况是，已知一篇文档，想要确定这个已存在的文档其主题分布是什么样的。这便是主题建模（文档生成模型的逆过程）的目的：自动地返现文档几种的主题分布。即文档d和单词w是可被观测到的，但是主题确实隐藏的。

由上分析，对于任意一篇给定文档，其$P(w_j|d_i)$是可以计算的。从而可以根据大量已知文档的文档-词语信息$P(w_j|d_i)$，训练出文档-主题$P(z_k|d_i)$和主题-词语$P(w_j|z_k)$，如下公式所示：

$$P(w_j|d_i) = \sum_{k=1}^{K} P(w_j|z_k)P(z_k|d_i)$$

故可以得到每个词语的生成概率为：

$$\begin{aligned} P(d_i, w_j) &= P(d_i)P(w_j|d_i) \\ &= P(d_i)\sum_{k=1}^{K} P(w_j|z_k)P(z_k|d_i) \end{aligned}$$

由于$P(d_i)$可事先计算求出，而$\mathrm{P}(w_j|z_k)$和$P(z_k|d_i)$未知，所以$\theta = (P(w_j|z_k), P(z_k|d_i))$就是我们要估计的参数（值），通俗点说，就是要最大化这个θ。

用什么方法进行估计呢，常用的参数估计方法有极大似然估计MLE、最大后验证估计MAP、贝叶斯估计等等。因为该待估计的参数中含有隐变量z，所以我们可以考虑EM算法。

关于EM算法，主要是可以分为E-step和M-step，这里就不再详细论述。可以参考PRML或者李航老师的小蓝书。

3.4、 LDA模型

从pLSA模型的分析中可以看出，pLSA模型的样本随机，参数虽未知但固定，属于**「频率派思想」**。

而在LDA模型中，样本固定，参数未知但不固定，是个随机变量，服从一定的分布，所以LDA模型属于**「贝叶斯派」**。

以上就是pLSA和LDA主要的区别，所以可以认为LDA就是在PLSA的基础上套上了贝叶斯框架（具体来说就是多加了两个先验参数）

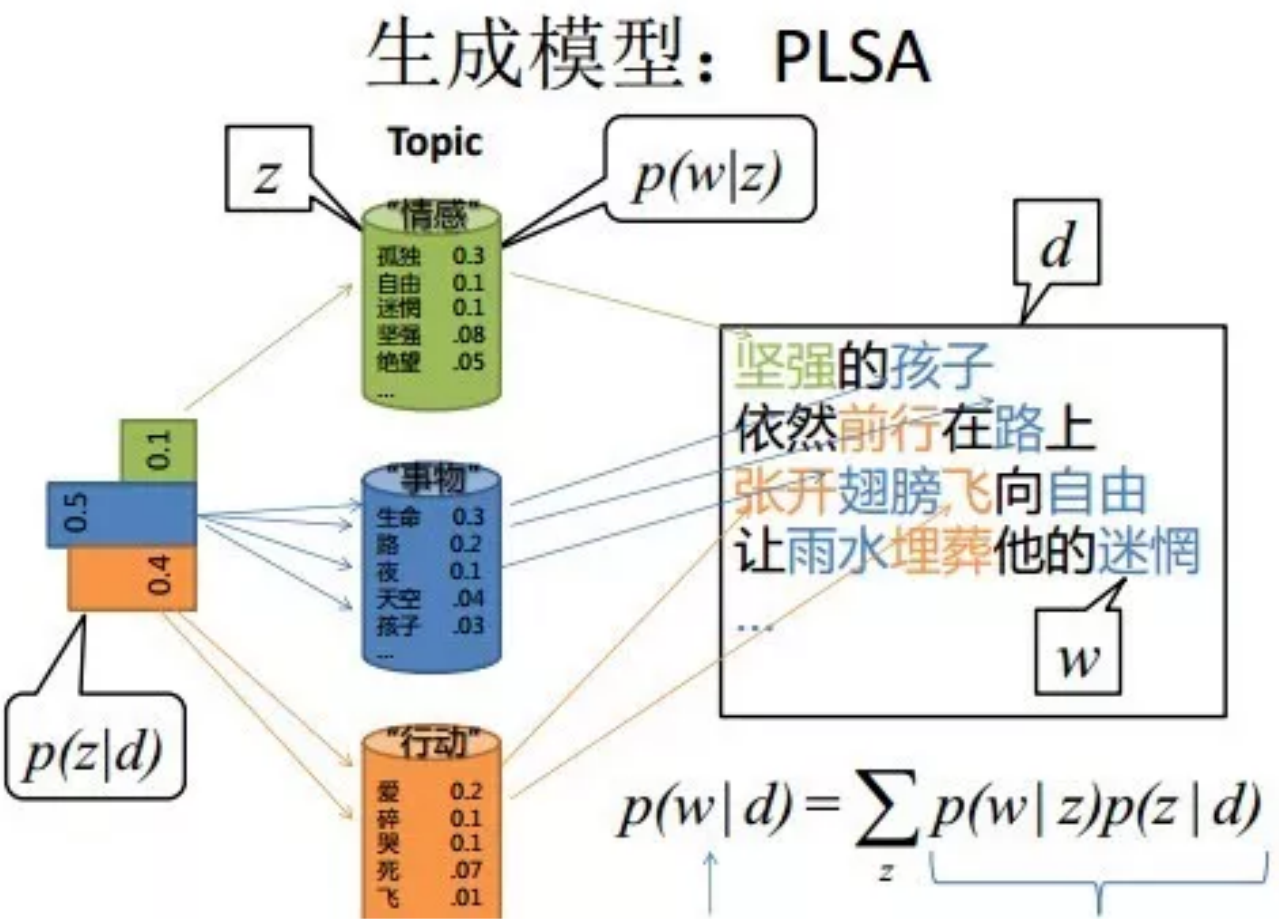
废话不多说，直接上LDA模型生成文档的套路：

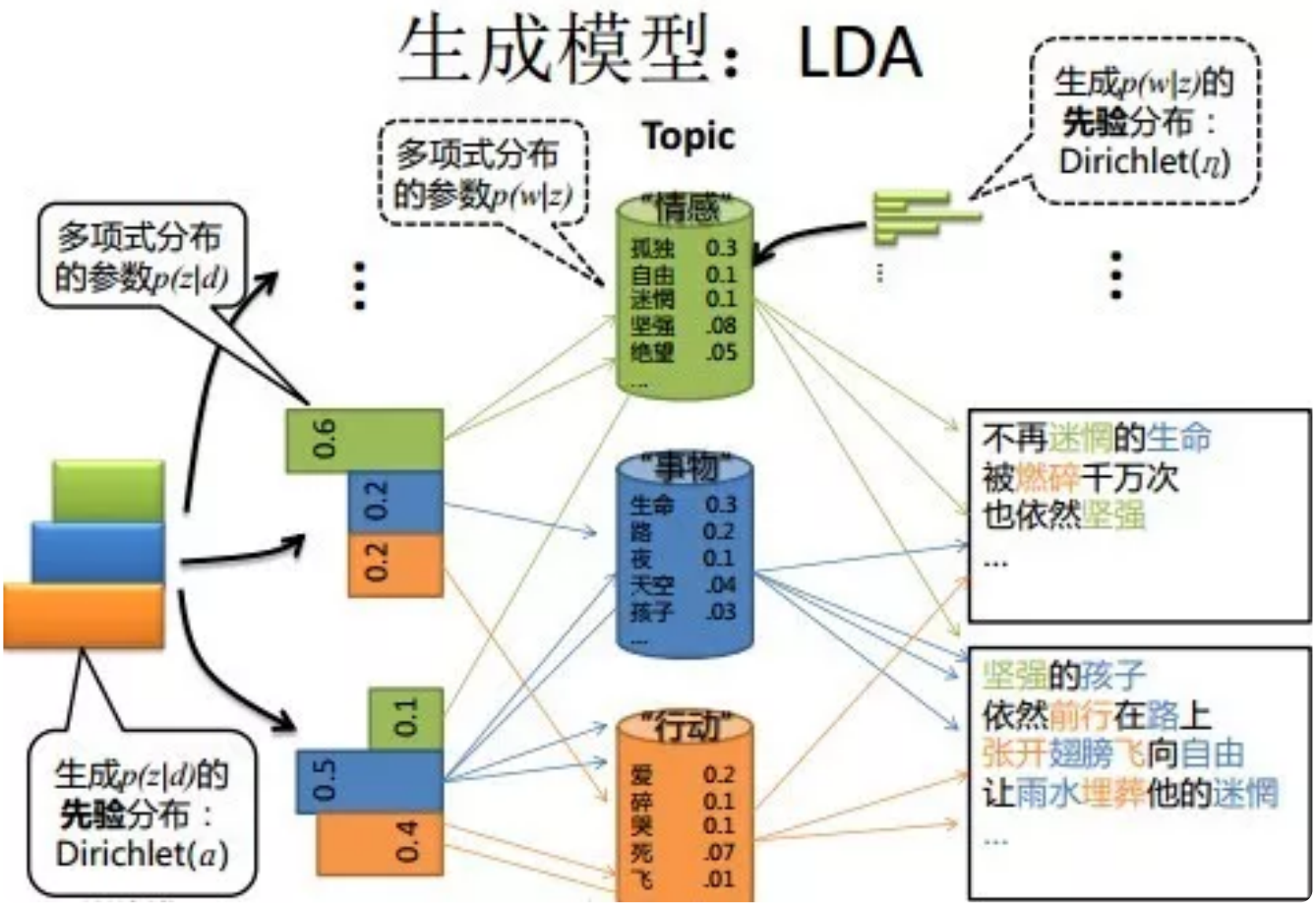
1. 按照先验概率$P(d_i)$选择一篇文档d_i；
2. 从狄利克雷分布（即Dirichlet分布）中取样生成文档d_i的主题分布θ_i，换言之，主题分布θ_i由超参数为α的Dirichlet分布生成；
3. 从主题的多项式分布中θ_i取样生成文档d_i第j个词的主题$z_{i,j}$；
4. 从超参数为β的狄利克雷分布（即Dirichlet分布）中取样生成主题$z_{i,j}$对应的词语分布$\phi_{z_{i,j}}$，换言之，词语分布$\phi_{z_{i,j}}$由参数为β的Dirichlet分布生成；

5. 从词语的多项式分布$\phi_{z_{i,j}}$中采样最终生成词语$w_{i,j}$

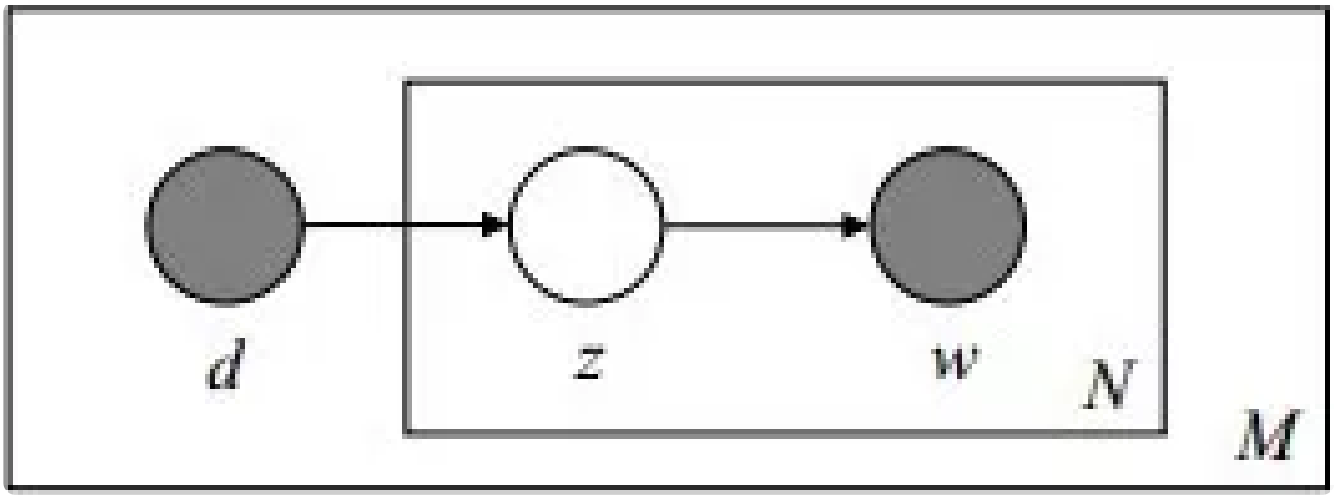
可以看出，LDA 在 PLSA 的基础上，为主题分布和词分布分别加了两个 Dirichlet先验。

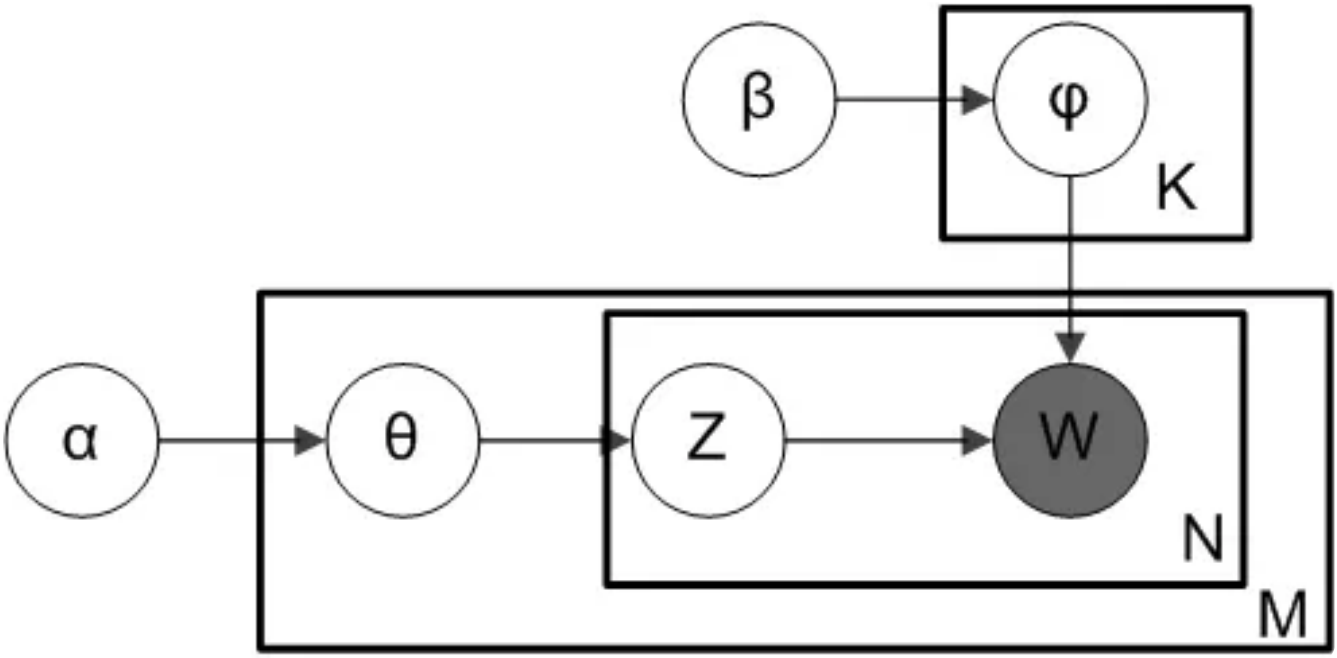
我们仍然以上面骰子模型举例说明，在PLSA中，我们会以固定的概率来抽取一个主题词，比如0.5的概率抽取教育这个主题词，然后根据抽取出来的主题词，找其对应的词分布，再根据词分布，抽取一个词汇。由此，可以看出PLSA中，主题分布和词分布都是唯一确定的。但是，在LDA中，主题分布和词分布是不确定的，LDA的作者们采用的是贝叶斯派的思想，认为它们应该服从一个分布，主题分布和词分布都是多项式分布，因为多项式分布和狄利克雷分布是共轭结构，在LDA中主题分布和词分布使用了Dirichlet分布作为它们的共轭先验分布。所以，也就有了一句广为流传的话 -- LDA 就是 PLSA 的贝叶斯化版本。下面两张图片很好的体现了两者的区别：





在PLSA和LDA的两篇论文中，使用了下面的图片来解释模型，它们也很好的对比了PLSA和LDA的不同之处。





上图是PLSA，下图是LDA，其中，阴影圆圈表示可观测变量，非阴影圆圈表示隐变量，箭头表示两变量间的条件依赖性，方框表示重复抽样，方框右下角表示重复抽样次数，Φ表示词分布，Θ表示主题分布， 是主题分布Θ的先验分布（即Dirichlet 分布）的参数，是词分布Φ的先验分布（即Dirichlet 分布）的参数，N表示文档的单词总数，M表示文档的总数。

所以，对于一篇文档d中的每一个单词，LDA根据先验知识确定某篇文档的主题分布θ，然后从该文档所对应的多项分布（主题分布）θ中抽取一个主题z，接着根据先验知识确定当前主题的词语分布ϕ，然后从主题z所对应的多项分布（词分布）ϕ中抽取一个单词w。然后将这个过程重复N次，就产生了文档d。

LDA参数估计：Gibbs采样

类似于pLSA，LDA的原始论文中是用的变分-EM算法估计未知参数，后来发现另一种估计LDA未知参数的方法更好，这种方法就是：Gibbs Sampling，有时叫Gibbs采样或Gibbs抽样，都一个意思。Gibbs抽样是马尔可夫链蒙特卡尔理论（MCMC）中用来获取一系列近似等于指定多维概率分布（比如2个或者多个随机变量的联合概率分布）观察样本的算法。

LDA训练

1. 对语料库中的每篇文档中的每个词汇w，随机的赋予一个topic编号z
2. 重新扫描语料库，对每个词，使用Gibbs Sampling公式对其采样，求出它的topic，在语料中更新
3. 重复步骤2，直到Gibbs Sampling收敛
4. 统计语料库的topic-word共现频率矩阵，该矩阵就是LDA的模型；

根据这个topic-word频率矩阵，我们可以计算每一个$p(word|topic)$概率，从而算出模型参数 ，这就是那 K 个 topic-word 骰子。而语料库中的文档对应的骰子参数$\theta_1,\cdots,\theta_M$在以上训练过程中也是可以计算出来的，只要在 Gibbs Sampling 收敛之后，统计每篇文档中的 topic 的频率分布，我们就可以计算每一个 p(topic|doc) 概率，于是就可以计算出每一个θ_m。由于参数θ_m是和训练语料中的每篇文档相关的，对于我们理解新的文档并无用处，所以工程上最终存储 LDA 模型时候一般没有必要保留。通常，在 LDA 模型训练的过程中，我们是取 Gibbs Sampling 收敛之后的 n 个迭代的结果进行平均来做参数估计，这样模型质量更高.

4.LDA主题模型实战

上面讲了那么多的LDA原理，尽量理解，有时间可以自己从底层开始写LDA框架，但是现在已经有很多成熟的LDA给我们写好了，也就咩必要重复造轮子。下面我们利用gensim提供的LDA接口来看一下主题模型的效果。（像这种试验性质的跑算法，推荐用notebook，可以实时看到每一步的结果）

In [1]:
```
# 输入文本数据
doc = ['因为森林人即将换代，这套系统没必要装在一款即将换代的车型上，因为肯定会影响价格。',
       '斯柯达要说质量，似乎比大众要好一点，价格也低一些，用料完全一样。我听说过野帝，但没听说过你说这车。',
       '摩雷的听爱卓和优特声听人声都非常OK，而且价格不算高。DLS的低频应该是不用担心的，建议后门装，不加低音的前提下！可以加好友聊配置方案',
       '那个貌似是设置载重胎压用的，也就是载几个人或者是拖拖车的胎压用的，原车有带内置胎压计无奈不显示数值也无适配的独立的显示屏，我加了一个外置
       '初看以为是中国有柴油版手动森林人呢。吓我一跳，原来是外国，油耗确实很低，怎么做到的？路况很好吗？还是基本都是长途？我的纯市区和两公里内短
```

In [2]:
```
# 分词
import jieba
doc = [jieba.lcut(s) for s in doc]

print(doc)
```

```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\ADMINI~1\AppData\Local\Temp\jieba.cache
Loading model cost 0.807 seconds.
Prefix dict has been built succesfully.
```

```
[['因为', '森林', '人', '即将', '换代', '，', '这套', '系统', '没', '必要', '装在', '一款', '即将', '换代', '的', '车型', '上', '，', '因
为', '肯定', '会', '影响', '价格', '。'], ['斯柯达', '要说', '质量', '，', '似乎', '比', '大众', '要', '好', '一点', '，', '价格', '也',
'低', '一些', '，', '用料', '完全', '一样', '。', '我', '听说', '过野帝', '，', '但', '没听说过', '你', '说', '这车', '。'], ['摩雷', '的',
'听', '爱卓', '和', '优特', '声', '听', '人声', '都', '非常', 'OK', '，', '而且', '价格', '不算', '高', '。', 'DLS', '的', '低频', '应该',
'是', '不用', '担心', '的', '，', '建议', '后门', '装', '，', '不', '加', '低音', '的', '前提', '下', '！', '可以', '加', '好友', '聊', '配
置', '方案'], ['那个', '貌似', '是', '设置', '载重', '胎压', '用', '的', '，', '也', '就是', '载', '几个', '人', '或者', '是', '拖', '拖车',
'的', '胎', '压用', '的', '，', '原车', '有', '带', '内置', '胎压计', '无奈', '不', '显示', '数值', '也', '无', '适配', '的', '独立', '的',
'显示屏', '，', '我加', '了', '一个', '外置', '的', '固特异', '的', '胎', '压计', '（', '之前', '老', '车上', '拆下', '的', '）', '，', '可
以', '显示', '数值', '总', '感觉', '安心', '一点'], ['初看', '以为', '是', '中国', '有', '柴油', '版', '手动', '森林', '人', '呢', '。',
'吓', '我', '一', '跳', '，', '原来', '是', '外国', '，', '油耗', '确实', '很', '低', '，', '怎么', '做到', '的', '？', '路况', '很', '好',
'吗', '？', '还是', '基本', '都', '是', '长途', '？', '我', '的', '纯', '市区', '和', '两公里', '内', '短途', '。', '油耗', '9', '点', '4',
'。']]
```

In [3]:
```
# 去停用词
sw_path = 'D:/NLP/data/stop_words/HIT.txt'
stop_words = []
with open(sw_path, 'r') as f:
    for line in f:
        line = line.strip()
        stop_words.append(line)
new_doc = []
for sentence in doc:
    new_s = []
    for w in sentence:
        if w not in stop_words:
            new_s.append(w)
    new_doc.append(new_s)
print(new_doc)
```

```
[['森林', '人', '即将', '换代', '这套', '系统', '没', '必要', '装在', '一款', '即将', '换代', '车型', '上', '肯定', '会', '影响', '价格'],
['斯柯达', '要说', '质量', '似乎', '大众', '好', '一点', '价格', '低', '用料', '完全', '听说', '过野帝', '没听说过', '说', '这车'], ['摩雷',
'听', '爱卓', '优特', '声', '听', '人声', '都', '非常', 'OK', '价格', '不算', '高', 'DLS', '低频', '应该', '不用', '担心', '建议', '后门',
'装', '不', '加', '低音', '前提', '下', '加', '好友', '聊', '配置', '方案'], ['貌似', '设置', '载重', '胎压', '载', '几个', '人', '拖', '拖
车', '胎', '压用', '原车', '带', '内置', '胎压计', '无奈', '不', '显示', '数值', '无', '适配', '独立', '显示屏', '我加', '外置', '固特异',
'胎', '压计', '之前', '老', '车上', '拆下', '显示', '数值', '总', '感觉', '安心', '一点'], ['初看', '以为', '中国', '柴油', '版', '手动',
'森林', '人', '跳', '原来', '外国', '油耗', '确实', '很', '低', '做到', '路况', '很', '好', '基本', '都', '长途', '纯', '市区', '两公里',
'内', '短途', '油耗', '9', '点', '4']]
```

In [4]:
```
# 文本向量化
import gensim
from gensim import corpora
#doc = [s.split() for s in doc]
dictionary = corpora.Dictionary(new_doc)
DT = [dictionary.doc2bow(s) for s in new_doc]
```

```
g:\python\install\lib\site-packages\gensim\utils.py:1212: UserWarning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

In [5]:
```
# gensim LDA API: https://radimrehurek.com/gensim/models/ldamodel.html
Lda = gensim.models.ldamodel.LdaModel
ldamodel = Lda(DT, num_topics=3, id2word = dictionary, passes=50)
```

In [6]:
```
# 输出主题结果，这里取了三个值，对于一对一文本分类其实可以只输出top1
print(ldamodel.print_topics(num_topics=3, num_words=3))
```

```
[(0, '0.019*"价格" + 0.019*"数值" + 0.019*"胎"'), (1, '0.009*"价格" + 0.009*"森林" + 0.009*"一点"'), (2, '0.027*"油耗" + 0.027*"很" + 0.027
*"低"')]
```


- END -

往期推荐

关于逻辑回归，面试官们都怎么问

2020-04-03

关于SVM，面试官们都怎么问

2020-03-27

【Kick Algorithm】十大排序算法及其Python实现

2020-02-28