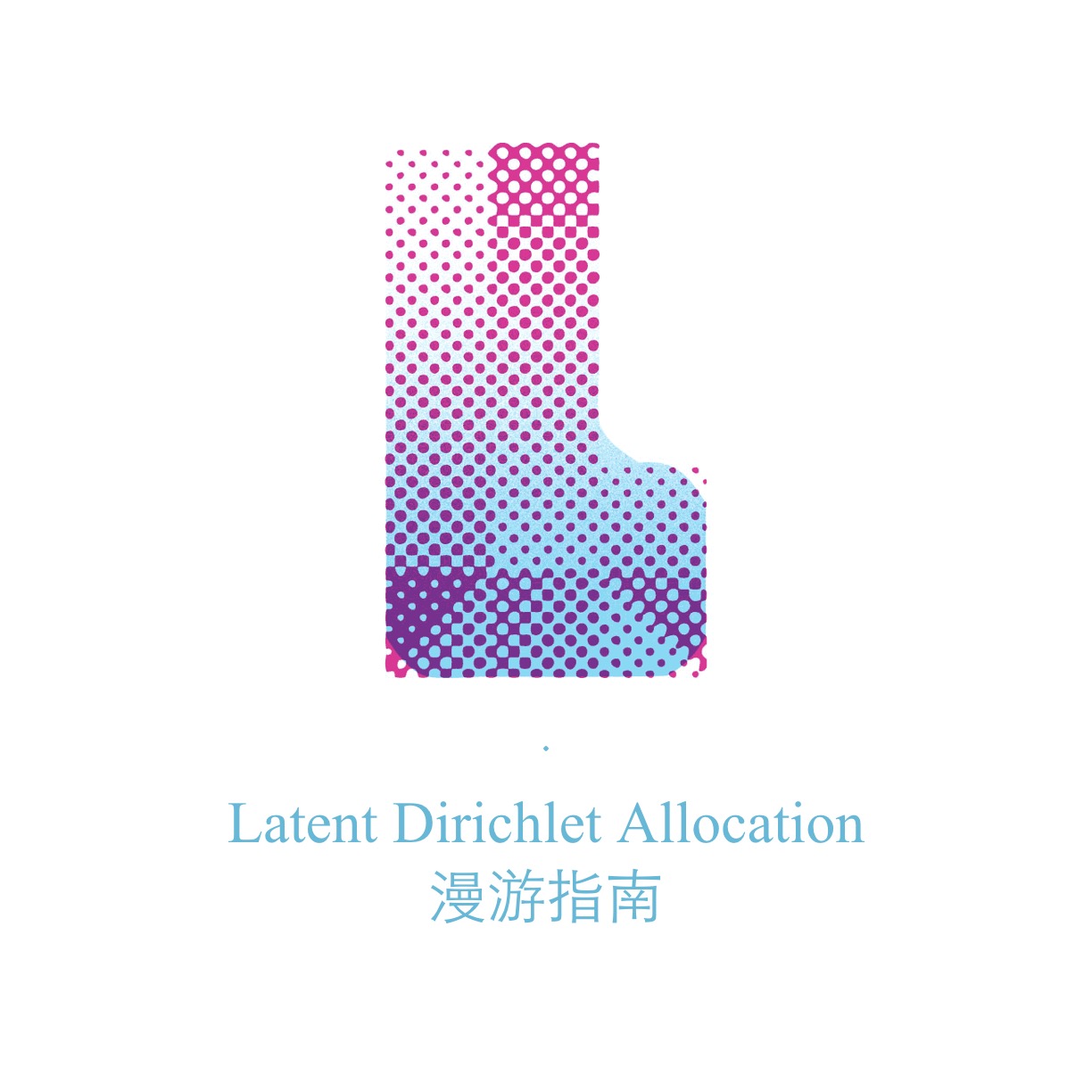
****

**马晨(**[**sharpstill@163.com**](mailto:sharpstill@163.com)**)**

**Version 1.0**

**2015年7月25日**

**前言**

LDA算法是主题模型领域非常著名的算法，值得深入研究应用，该算法也有很深刻的数学背景和技术启发。曾经有哲人说：万物皆数。我个人是个十分喜欢数学，喜欢算法，热爱技术的人，非常想从算法中寻找人工智能的永恒之道。我尤其记得19世纪的数学家赫尔曼.汉克尔说的好：

就大多数学科而言，一代人摧毁的正是另一代人所建造的，而他们所建立的也必将被另一代人所破坏。只有数学不同，每一代人都是在旧的建筑物上加进新的一层。

所以说，数学的价值还具有一种永世不灭的恒久性，其他学科的时尚潮流往往随着时代的变迁被人遗忘，那些旨在改变世界的理想，最终往往变成思想垃圾。而只有数学和算法与此不同。

我们探究前人伟大的成果时，就能体会到奥利弗.亥维赛的精辟论说：“逻辑能够很有耐性，因为它是永恒的”。

我在选择分析Latent Dirichlet Allocation(LDA)这个算法课题时，我考虑了很多因素，首先，该算法是已经被学术界和工业界广泛接受的；其次，该算法能带来更多的新技术启示；最后,该算法能为您的工作，您的研究带来最具实用性的技术启发。

LDA算法恰好满足了这个条件。

虽然网上已经有许多分析LDA算法的博客文章，但是这部作品对比网上的博文，仍旧有其**独特的价值**：

**1.这部作品理论与实践并重:**网上的同类文章非常零散，理论推导部分也缺乏**关键细节**，这部作品的每一条公式都由作者手把手为您推理（每一条公式都有详细的解释和备注），并且按照初学者的思路娓娓道来，从逻辑链条上打通算法的整个环节，让用户有醍醐灌顶的认识。并且在**实践部分**，作者以多年的工作实践经验为基础，精选了6个实现简单但又有巨大应用价值的LDA的应用方法，将成为读者未来工作实践不可多得的资料。

**2. 这部作品饱含了作者的独到见解：**这部作品最大的特色是从理论分析开始就有包含着许多作者自己独到的理解和分析，从不同角度完美解释算法的整个流程。

**3. 读者可以在这部作品各取所需：**有的工程师对于算法推导不是很感兴趣，这种情况下可以跳过前几章，直接从第4章读LDA算法怎么具体实现。如若未来有兴趣研究LDA算法的来龙去脉时，可以再来看前几章的理论推导部分。如果读者对大数据环境下的LDA感兴趣，包括怎么在Hadoop、Spark上实现LDA算法可以直接看第5章。

**4. 这部作品首次将LDA算法引入大数据时代：**大数据时代最大的特色就是信息爆炸，各种文本数据，用户生成（UGC）数据也变得非常庞大，网上查阅到的LDA算法资料大部分都是不能应对大数据环境的，这部作品的第5章深入浅出地讲解了大数据环境下怎么实现并行化的LDA算法。

**章节安排：**

第1章为相关背景介绍，介绍了算法知识的来源：从18世纪的欧拉讲到剑桥大学的David Blei。

第2章和第3章为算法的理论分析阶段：第2章为LDA算法的前置知识，为LDA做了理论工具上的准备。

在第2章力求做到关键证明不遗漏，这样就可以与后面第3章的LDA算法推导构成一个完整的推理链条！这一章的有些证明需要一些简单的微积分知识，但如果读者忘记了所有基础的微积分知识的话，那么看不懂某条证明，就请姑且相信我的推理是对的吧，跳过去往后看，日后再复习。

第3章为LDA算法推导部分，用严谨的数学推导和清晰的讲解（每个公式都做了清晰的标注），让读者认识该推理方法。

第4章为实现和应用部分，用伪代码方式庖丁解牛，讲解代码实现的精髓，然后结合作者多年的工作实践，写了该算法的几个最具实用价值的应用。

第5章为并行化，在大数据如火如荼的今天，要想大规模运行LDA算法，就要靠并行化技术了，这一章从2个算法的改进形式讲解了该技术的并行化，并且可以放在目前最流行的spark大数据引擎上运行。

本文是一个指南指引性质的作品，仍未完善完美，有许多地方未涉及到，比如变分推断推导LDA算法的过程等，未来这部作品还会完善补充，第二版会增加新的章节。本书还有些地方可能仍有疏漏，本人水平有限，如果大家发现纰漏，请及时联系我修正。**希望这部作品日臻完美。**

**作者：马晨 2015.7.7**

目录

[第1章 背景 1](#_Toc424678173)

[第2章 前置知识 5](#_Toc424678174)

[2.1 gamma函数 5](#_Toc424678175)

[2.2 二项分布(Binomial distribution) 6](#_Toc424678176)

[2.3 beta分布(beta distribution) 6](#_Toc424678177)

[2.4 多项分布(multinomial distribution) 10](#_Toc424678178)

[2.5 狄利克雷分布(dirichlet distribution) 11](#_Toc424678179)

[2.6 共轭先验分布(conjugacy prior) 12](#_Toc424678180)

[2.6.1 从二项分布到beta分布 13](#_Toc424678181)

[2.6.2 从多项分布到Dirichlet分布 15](#_Toc424678182)

[2.7 总结 17](#_Toc424678183)

[第3章 LDA推导 18](#_Toc424678184)

[3.1 unigram假设 18](#_Toc424678185)

[3.2 Latent Dirichlet Allocation Intro 20](#_Toc424678186)

[3.3 马尔可夫链 🡪 Metropolis-Hasting 🡪 Gibbs Sampling 22](#_Toc424678187)

[3.3.1 马尔可夫链(markov chain) 22](#_Toc424678188)

[3.3.2 Metropolis-Hasting算法 25](#_Toc424678189)

[3.3.3 Gibbs Sampling 26](#_Toc424678190)

[3.4 伟大的采样公式： Collapsed Gibbs Sampling采样公式推导 27](#_Toc424678191)

[3.5 总结 33](#_Toc424678192)

[第4章 实现与应用 34](#_Toc424678193)

[4.1 实现 34](#_Toc424678194)

[4.2 应用 40](#_Toc424678195)

[4.2.1 相似文档发现 40](#_Toc424678196)

[4.2.2 自动打标签 41](#_Toc424678197)

[4.2.3 LDA与LR(逻辑斯蒂回归)结合做新闻个性化推荐系统 42](#_Toc424678198)

[4.2.4 topic rank[10] 44](#_Toc424678199)

[4.2.5 word rank 46](#_Toc424678200)

[4.2.6 文章质量评分算法 49](#_Toc424678201)

[4.2.7 总结 52](#_Toc424678202)

[第5章 并行化 53](#_Toc424678203)

[5.1 AD-LDA 53](#_Toc424678204)

[5.2 spark-LDA 54](#_Toc424678205)

[5.2.1 切分块 55](#_Toc424678206)

[5.2.2 选择 56](#_Toc424678207)

[5.2.3 计算和合并 57](#_Toc424678208)

[5.2.4 总结 58](#_Toc424678209)

[附录 59](#_Toc424678210)

[参考文献 61](#_Toc424678211)

*飞机和飞鸟形态、结构和原理都不相同，但都能飞翔，人工智能未来也许如此。*

# 背景

LDA算法使用的全部知识的渊源可以追溯到18世纪的欧拉，欧拉（Leonhard Euler ，1707年4月15日～1783年9月18日），瑞士数学家。欧拉一生贡献颇丰，1734年，欧拉解决巴塞尔问题就立即出名了，巴塞尔问题如下：



(.)

这个问题困扰了几个世纪的数学家，当时的数学家只知道该级数的值小于2，但不知道具体精确值，欧拉准确的推导出该式的值=，欧拉的方法聪明而新颖，他创造性将有限多项式的观察推广到无穷级数，并假设相同的性质对于无穷级数也是成立的：



(.)

欧拉最后的发现是令人惊奇的，π这个数字在于圆周率无关的场合中出现了，这足以说明数学之中、自然之中、冥冥之中存在着某些神秘的联系。虽然以现代数学的眼光来看，欧拉的证明还不严密。但作为第一个（富有创造性的）证明，欧拉的这个证明永远有着其宝贵的价值。欧拉的另一个发现就是发现了gamma函数 ，该函数后被广泛应用于概率论，这个函数也是本文的主角之一。



图 1‑1 [Euler](http://baike.baidu.com/link?url=iIq4UAvChdqgvGhUy8cEilqX-pDPyzbvbcJ9fSLT-rym-g2EptGcyVGFtrKMsPNOz7XXZlhjvzFSAspowYbdMot3lbtoMD9PqEmNiDymj_mWDMfyYxvD314HHXLSpYx4RsywhZhqoiJM0L0ajuDzZq)

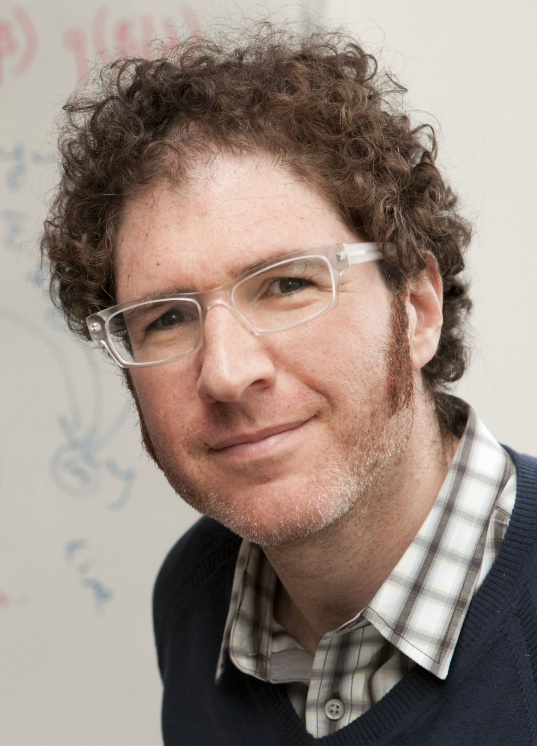
作为算法标题之一的Dirichlet， wiki一下，一个19世纪的人映入了我们的眼帘，Dirichlet（1805～1859）德国数学家，生与现德国 Duren（当时属法国），卒于哥廷根。他是解析数论的奠基者，也是现代函数观念的定义者。在本文中该数学家的主要贡献是Dirichlet分布。



**图 1‑2** [**Peter Gustav Lejeune Dirichlet**](http://en.wikipedia.org/wiki/Peter_Gustav_Lejeune_Dirichlet)

但是这还不是故事的全部，说到底19世纪的时候还没有发明计算机，LDA应该不是这哥们发明的，于是继续search，最后查明英国剑桥大学的David M.Blei是最初LDA论文的作者。Blei同学借用了Dirichlet Distribution，而创造了Latent Dirichlet Allocation。

下面这张照片的blei以PLSA（LDA之前的另一个概率模型）为基础，加上了贝叶斯先验，从而诞生了LDA算法，LDA算法最初的论文使用的是变分EM方法训练（Variational Inference）。该方法较为复杂，而且最后训练出的topic主题非全局最优分布，而是局部最优分布。后期发明了Collapsed Gibbs Sampling方法，推导和使用都较为简洁。Blei及其LDA算法正式的介绍如下：



**图 1‑3** [**David Blei**](http://en.wikipedia.org/wiki/David_Blei)

Latent Dirichlet Allocation是Blei等人于2003年提出的基于概率模型的主题模型算法，LDA是一种非监督机器学习技术，可以用来识别大规模文档集或语料库中的潜在隐藏的主题信息。该方法假设每个词是由背后的一个潜在隐藏的主题中抽取出来。

对于语料库中的每篇文档，LDA定义了如下生成过程（generative process）：

1. 对每一篇文档，从主题分布中抽取一个主题；

2. 从上述被抽到的主题所对应的单词分布中抽取一个单词；

3. 重复上述过程直至遍历文档中的每一个单词。

LDA认为每篇文章是由多个主题mix混合而成的，而每个主题可以由多个词的概率表征。所以整个程序的输入和输出如下所示：

|  |
| --- |
| **LDA算法的输入与输出** |
| **算法输入：**分词后的文章集（通常为一篇文章一行）  主题数K，超参数α和β |
| **算法输出：**1.每篇文章的各个词被指定(assign)的主题编号：tassign-model.txt  2.每篇文章的主题概率分布θ：theta-model.txt  3.每个主题下的词概率分布φ：phi-model.txt  4.程序中词语word的id映射表：wordmap.txt  5.每个主题下φ概率排序从高到低top n特征词：twords.txt |

如果你想使用一下LDA算法，我建议你从Gibbs LDA++代码起步使用。（<http://gibbslda.sourceforge.net/>），你用一下，就会发现该算法使用方式还算傻瓜，并且生成的结果文件也挺规则，根据manual一看便懂。输入分词后的文件，一个文章一行，输出其中看到每个主题规则文件.twords如下格式所示：

|  |  |
| --- | --- |
| Topic 0th: | Topic 1th: |
| 食品 0.022937  不合格 0.006634  加多宝 0.006634  检测 0.006433  包装 0.004403  儿童 0.004038  抽检 0.004016  王老吉 0.003759  批次 0.003483  检验 0.003379  红罐 0.003342  样品 0.003312 | 农业 0.022957  农村 0.022404  农民 0.014940  土地 0.013734  粮食 0.007909  农产品 0.006644  耕地 0.005823  集体 0.005067  农户 0.004852  种植 0.004674  流转 0.004177  农业部 0.003693 |

# 前置知识

本章所描述的工具和线索在后期LDA算法的采样公式推导中会全部明了，关于为什么需要使用这些知识要素，这里面有很长的渊源和历史，比如在概率论和数理统计中，gamma函数被广泛使用，而在最终的LDA采样公式中，你会发现，gamma函数被神奇地约去消失了。我们在后面的章节中可以看到，LDA算法精妙之处在于用令人屏息的洞察力作为纽带，将全部零散的部件组合在了一起。

## gamma函数

所谓的gamma函数其实就是阶乘的函数形式，直接给出



(.)

可以直接算出

* 也可以算出



(.)

* 接下来验证



也正因为如此，

(.)

## 二项分布(Binomial distribution)

在概率论中，二项分布即重复n次独立的伯努利试验。在每次试验中只有两种可能的结果（成功/失败），每次成功的概率为p，而且两种结果发生与否互相对立，并且相互独立，与其它各次试验结果无关，事件发生与否的概率在每一次独立试验中都保持不变，则这一系列试验总称为n重伯努利实验，当试验次数为1时，二项分布就是伯努利分布。

在给出二项分布之前，我们来做一个例子，假设你在玩CS这个游戏，你拿着狙击枪，敌人出现你打中敌人的概率是p，打不中敌人的概率是1-p，那么敌人**第一次出现你没打中而第二次出现你打中的概率是 (1-p)⋅p。如果敌人出现了n次，而你打中了其中的k次，而不确定具体在哪k次（第1次，还是第4次），这样从n次中任取k次的次数是****，而这不确定的k次打中敌人的概率是：****，通过这个例子我们便得知了二项分布的概率。**

二项分布的概率密度函数是：



(.)

## beta分布(beta distribution)

在概率论中，beta分布是指一组定义在区间(0,1)的连续概率分布，有两个参数和，且>0。

Beta分布的概率密度函数是：



(.)

随机变量X服从参数为的beta分布通常写作：。

这个式子中分母的函数称为B函数。

* 这里我们来证明一个重要的公式，该公式中的关系在LDA算法Gibbs Sampling采样公式中也有使用，这个关系也就是**B函数和Gamma函数的关系**（该公式也被称为第一型欧拉积分）：



(.)

**证明方法1：**





**证明方法2:**



t

s

t=s

t=-s



这便是坐标变换后新坐标的积分区间



* Beta分布的期望

对于后面2.6节提到的Dirichlet分布也有相似结论：



(.)

这个结论在LDA算法做完Gibbs sampling后，估计和时用到。

## 多项分布(multinomial distribution)

多项分布[[1]](#footnote-1)是二项分布的推广扩展，在n次独立试验中每次只输出k种结果中的一个，且每种结果都有一个确定的概率p。多项分布给出了在多种输出状态的情况下，关于成功次数的各种组合的概率。

举个例子，投掷n次骰子，这个骰子共有6种结果输出，且1点出现概率为p1，2点出现概率p2，…多项分布给出了在n次试验中，骰子1点出现x1次，2点出现x2次,3点出现x3次，…，6点出现x6次。这个结果组合的概率为：



(.)

也可以用gamma函数表示（这个写法的形式和Dirichlet分布相似）：



(.)

下面我们通过一个例题加深对多项分布的印象:

* 问题：



## 狄利克雷分布(dirichlet distribution)

dirichlet分布是beta分布在多项情况下的推广，也是多项分布的共轭先验分布（共轭先验分布在2.6节讲）。dirichlet分布的概率密度函数如下：



(.)

二项分布和多项分布很相似，Beta分布和Dirichlet 分布很相似，而至于“Beta分布是二项式分布的共轭先验概率分布，而狄利克雷分布（Dirichlet分布）是多项式分布的共轭先验概率分布”这点在下文中说明。

另一个重要的公式是:

(.)

这个公式的结构和证明**相似于**上文中“B函数和Gamma函数的关系—公式(2.6)”，这里略去证明。

## 共轭先验分布(conjugacy prior)

In Bayesian probability theory, if the posterior distributions p(θ|x) are in the same family as the prior probability distribution p(θ), the prior and posterior are then called conjugate distributions, and the prior is called a conjugate prior for the likelihood function.[[2]](#footnote-2)

所谓的共轭，只是我们选取(choose)一个函数作为似然函数(likelihood function)的prior probability distribution，使得后验分布函数[[3]](#footnote-3)(posterior distributions)和先验分布函数形式一致。比如Beta分布是二项式分布的共轭先验概率分布，而狄利克雷分布(Dirichlet分布）是多项式分布的共轭先验概率分布。为什么要这样做呢？这得从贝叶斯估计[4]谈起：

根据贝叶斯规则，后验分布=似然函数\*先验分布



(.)

*参数估计是一个重要的话题。对于典型的离散型随机变量分布：二项式分布，多项式分布；典型的连续型随机变量分布：正态分布。他们都可以看着是参数分布，因为他们的函数形式都被一小部分的参数控制，比如正态分布的均值和方差，二项式分布事件发生的概率等。因此，给定一堆观测数据集（假定数据满足独立同分布），我们需要有一个解决方案来确定这些参数值的大小，以便能够利用分布模型来做密度估计。这就是参数估计！*

*对于参数估计，一直存在两个学派的不同解决方案。一是频率学派解决方案：通过某些优化准则（比如似然函数）来选择特定参数值；二是贝叶斯学派解决方案：假定参数服从一个先验分布，通过观测到的数据，使用贝叶斯理论计算对应的后验分布。先验和后验的选择满足共轭，这些分布都是指数簇分布的例子。[[4]](#footnote-4)*

简而言之，假设参数也是变量而非常量，而且在做试验前已经服从某个分布p()（来源于以前做试验数据计算得到，或来自于人们的主观经验），然后现在做新试验去更新这个分布假设。如果不知道最大似然估计(Maximum Likelihood)的概念，参见附录。

### 从二项分布到beta分布

注意二项分布概率密度函数为，将参数去掉，变成形式：，再加上归一化因子B(α,β)（注意这个归一化因子含有k和n，但绝不含p），变为beta分布：，beta分布前一项B函数是确保beta分布是归一化(normalized)。

* 求证：beta分布确实是二项分布的共轭先验分布

证明：



(.)

这就可以看到后验分布(post distribution)又变为beta分布，也就是和先验(prior distribution)一致了，因此我们称之为共轭(conjugacy)。

观察到后验和先验都是beta分布，但变为了，超参数变了。如果以后有新增的观测值，后验分布又可作为先验分布来进行计算。具体来讲，在某一个时间点，有一个观测值，此时可以得到后验，之后，每一个观测值的到来，都以之前的后验作为先验，乘以似然函数后，得到修正后的新后验。在这每一步中，其实我们不需要管什么似然函数，我们可以将后验分布看作是以代表x=1出现“次数”的参数s和代表x=0出现“次数”的参数f为参数的beta分布：当有一个新的x=1的观测量到来的时候，s=α+1,f=β，即α的值相应的加1;否则s=α,f=β+1即β的值加1。所以这也就是超参数(α,β)又被称之为伪计数(pseudo count)的原因。

我们可以把上面性质所表示的方法看作为序列方法（sequential approach），该方法是贝叶斯观点中很自然得到的学习方法。他非常适合实时学习场景。在某一时刻，有一个观测数据，或是一小批量数据，在下一批观测数据到来之前我们就可以丢弃他们，因为我们可以在一开始的小批量数据中得到我们的后验分布模型，当有新的一批数据到来时，只需要更新这个模型就够了。一个实时学习应用场景是：在所有数据到来之前，预测就必须通过之前稳定到达的一部分数据流来做出预测。注意到，因为这种序列方法不需要将所有数据都载入内存，因此他在大数据的应用将非常有效。当然，之前频率学派所使用的最大化似然函数方法也可以转换成这种序列式方法的。

另外，我们做先验分布的目的是估计参数，比如投掷硬币试验，我们需要根据已有的观测数据，估计下一次试验硬币的正面结果概率是多少。



(.)

这正是beta分布的均值，结合beta分布的均值公式可以得到：

(.)

### 从多项分布到Dirichlet分布

通过观察多项式分布的形式（），我们选取先验分布的形式为（保留带概率pk的项）：



(.)

这里α1, α2,… αk代表这个分布的超参数（或伪计数），。由于有这两个条件的限制(用μk等价于pk​)，因此{μk}之上的分布是K-1维度：

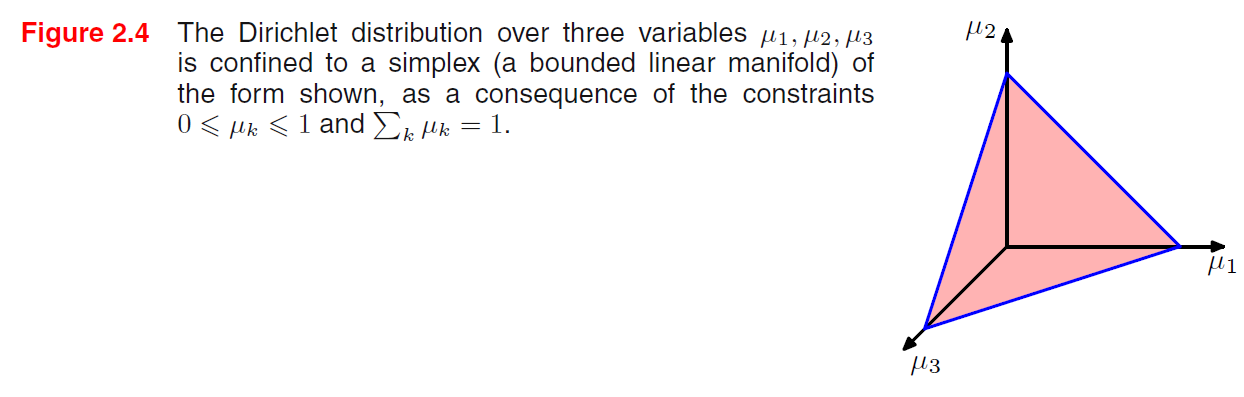


图 2‑1 节选自《Pattern Recognition and Machine Learning》[7] p77

注：上面这张图的含义是假设一共有3个概率，这3个概率μ的范围被限制在红色框所围的坐标轴内。

将(2.16)式加入系数归一化(normalized),便可得到Dirchlet分布的概率密度表达式：



由于，这里mk是:做n次实验输出k编号结果的次数，这个mk也叫充分统计量。更上一节一样，这里的mk就是根据新试验去更新超参数。

* 求证：dirichlet分布确实是多项分布的共轭先验分布

证明:

 (.)

## 总结

1. 贝叶斯学派采用给参数赋予先验分布，并使得先验与后验共轭，通过求后验均值来得到参数的估计，频率学派通过某个优化准则比如最大化似然函数来求得参数的估计；不管是哪个学派思想，都要用到似然函数，注意到频率学派所使用的似然函数是N次伯努利实验下的似然函数，但贝叶斯学派所使用的似然函数是二项式分布形式的似然函数（二项式分布是N次伯努利实验中出现事件A的次数的分布）。

2. 当拥有无限数据量时（beta分布式中s和f都趋向于无穷，dirichlet分布式中m趋向于无穷），贝叶斯方法和频率学派方法所得到的参数估计是一致的。当在有限的数据量下，贝叶斯学派的参数后验均值的大小介于先验均值和频率学派方法得到参数估计。比如在抛硬币实验中，当数据量有限时，先验均值为0.5，后验均值将会比先验大，比频率学派得到参数估计小。

3. 随着观测数据的增多，后验分布曲线越来越陡峭（越来越集中），即方差越来越小（后验方差总比前验方差小），当数据量无穷大时，方差趋近于0，即随着数据越来越多，后验的不确定性在减小。

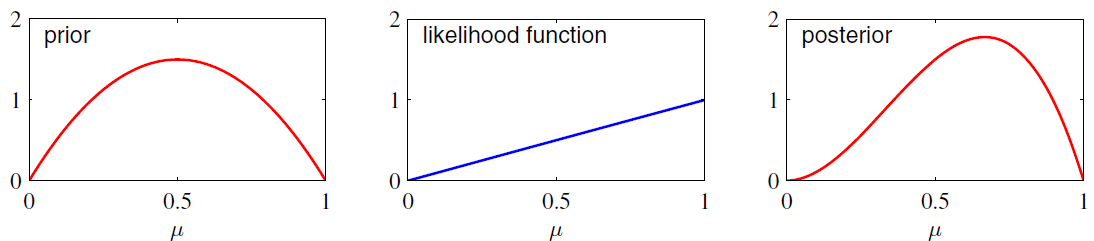


图 2‑2 后验分布曲线

# LDA推导

## unigram假设

假设有N个服从i.i.d(独立同分布)的单词从一个多项式分布(multinomial)中抽取，在N个词中，我们关注vi的发生次数为n(t)，那么则记为w~Mult(w|p)，该文档生成的概率是



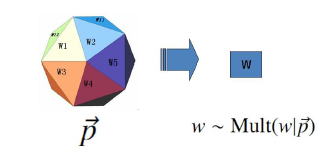
(.)

其中nt为该单词在文章中出现的次数,V为字典中的单词个数。在这里其中每个词的概率。由于这种方法不考虑文章内单词间的顺序，因此被称之为词袋模型（bag of words）。



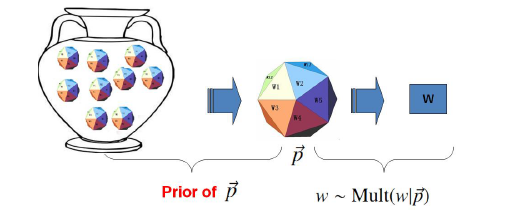
**图 3‑1基于unigram的盘子图[[5]](#footnote-5)**

这个生成文档的概率可以用投掷骰子的游戏来模拟：上帝拿出一枚骰子，骰子有V个面，每个面代表一个词，每个面的概率是，投掷N次骰子每个面产生的次数分别是次，然后计算生成语料库的概率。



**图 3‑2上帝投掷V个面的骰子生成文档**

现在来引入Dirichlet分布作为多项分布的先验分布，则文本单词概率。贝叶斯学派下，我们不知道上帝到底用哪个骰子来投掷，所以从一个服从Dirichlet分布的坛子中抽取一个骰子，然后投掷生成文档：



**图 3‑3贝叶斯观点下Unigram Model**

根据dirichlet分布函数密度公式，我们得到了超参数(hyper parameter)（或伪计数(pseudo count)）α的似然函数：

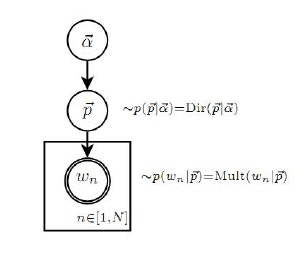


(.)

因为，所以推出的后验分布：



(.)



**图 3‑4 Unigram Model的概率图模型[[6]](#footnote-6)**

**在贝叶斯框架下，参数p可以得到估计，因为我们已经有了后验分布，**根据第2章的公式(2.7)，**于是：**



(.)

也就是说对于每一个pi，我们做如下参数估计即可：

上述参数估计很直观：每个单词产生的概率估计值是对应事件的先验的伪计数和数据中的计数的和在整体计数中的比例。进一步可以产生整体文本语料概率为：[3]



(.)

## Latent Dirichlet Allocation Intro

让我们再回首第一章背景中的那句话：

对于语料库中的每篇文档，LDA定义了如下生成过程（generative process）：

1. 对每一篇文档，从主题分布中抽取一个主题；

2. 从上述被抽到的主题所对应的单词分布中抽取一个单词；

3. 重复上述过程直至遍历文档中的每一个单词。

什么，没看懂？爱因斯坦曾说：上帝不扔骰子。我们这里假设我们的机器上帝是扔骰子的，所以我换一种投掷骰子的说法解释这个过程。

LDA的逻辑与3.1节生成文档的逻辑类似，这里我引用《LDA数学八卦》的一幅图如下说明LDA如何生成文档：

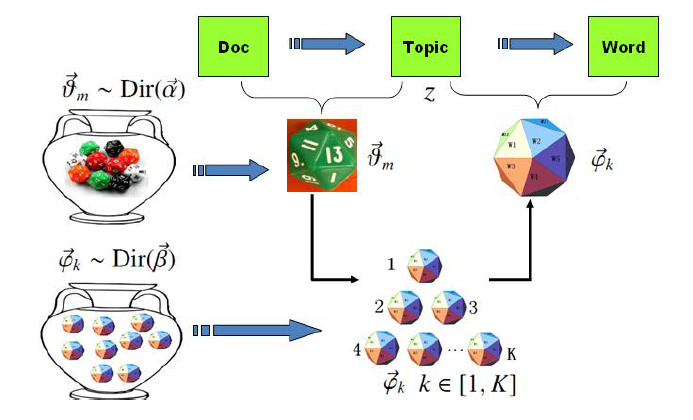


图 3‑5 LDA模型[[7]](#footnote-7)

每次生成一篇新的文档前，上帝从服从的坛子中抽取出一个doc->topic骰子，然后重复以下步骤：

1. 投掷这个doc->topic骰子，得到一个topic编号z。
2. 从服从分布的坛子里共K个topic->word骰子中选择编号为z的那个，投掷这枚骰子，于是得到一个词。

更为详细的过程如下图所示：

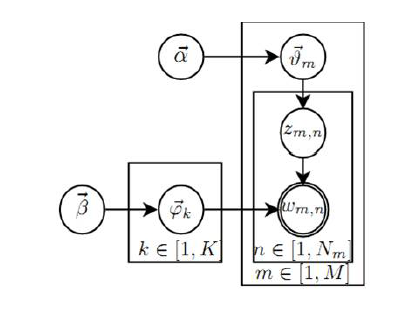


图 3‑6 LDA图模型表示

回到上面的那个过程，更为正式的表述如下：

1. ，这个过程表示在生成第**m**篇文档的时候，先从第一个坛子中抽中一个doc-topic骰子，然后投掷这枚骰子生成了文档中第**n**个词的**主题(topic)编号**。[[8]](#footnote-8)此为Dirichlet- Multinomial共轭。
2. ，这个过程表述了如下动作生成语料中第m篇文档的第n个词：在上帝手头的K个topic-word骰子中，挑选编号为k=zm,n的那个骰子进行投掷，然后生成word:。[[9]](#footnote-9)此为Dirichlet- Multinomial共轭。

## 马尔可夫链 🡪 Metropolis-Hasting 🡪 Gibbs Sampling

在正式推导LDA的Gibbs Sampling采样公式之前，读者有必要了解为什么需要这样推导，做到知其然知其所以然。[[10]](#footnote-10)

### 马尔可夫链(markov chain)

马尔可夫链条通俗说就是根据一个转移概率矩阵去转移的随机过程（ 马尔可夫过程）[[11]](#footnote-11)，该随机过程在Page Rank算法中也有使用。如下图所示：

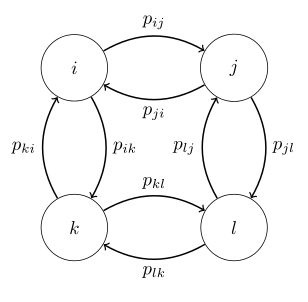
[](http://en.wikipedia.org/wiki/File:Kolmogorov_criterion_dtmc.svg)

图 3‑7马尔可夫转移图

通俗理解：这里的每个圆环类似一个岛屿(状态)，比如i到j的概率是pij，每个节点的出度概率之和=1。现在假设我要根据这个图去转移，那么首先需要把这张图“翻译”成如下的矩阵：



即该矩阵的第i行第j列pij表示从i岛屿走到j岛屿的概率。

这就是转移矩阵，我现在身处的位置用一个向量表示π0=(i,j,k,l)。

现在假设我站在i岛屿，我的位置向量π0=(1,0,0,0)，第一次转移，也就是相当于π1=π0\* P=[pii, pij, pik, pij]，相乘后向量π1每一项为小于1的小数。就是说我有Pii的机会比率留在原始i岛屿，有Pij的机会到达j岛屿，……；第二次转移时，以我刚才的位置向量为基础得到π2=π1\* P；以此类推……

**有这么一种情况**，我的位置向量在若干次转移后会达到一个稳定状态，再转移π向量也不变化了，这个状态称之为平稳分布状态π\*，这个情况需要满足一个条件：这就是**Detailed Balance**。

下面通过一个例子深刻认识**Detailed Balance**：

首先要明确：一个马尔可夫链条要成为reversible markov chain 才有可能达到Detailed Balance条件，何为reversible markov chain?也就是满足Detailed Balance的 马尔可夫链条。是不是有点鸡生蛋蛋生鸡的味道。换一种说法，reversible markov chain也就是其转移概率满足Kolmogorov’s criterion的链条，这个Kolmogorov’s criterion即表示以下含义，在图3.7中：。

也就说一个封闭的环中，一个方向的概率连乘积=反过来方向的概率连乘积。

例子：我们用假设构造这样一个转移矩阵：

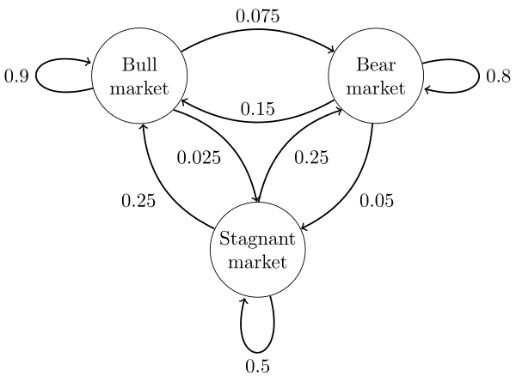


图 3‑8 detail balance的转移矩阵图

先构造出转移矩阵，

假设我们的初始向量为π=(1, 0, 0)，现在用计算机软件计算矩阵乘法得到π \* P 1000次后的stationary distribution = (0.625, 0.3125, 0.0625)。注意这个平稳分布有且只有一个（**是唯一的**）。

所谓的**detailed balance**就是：



(.)

这里的π是平稳分布(stationary distribution)的那个π\*。代到这个例子里：



所以得出结论，detailed balance成立。

有了detailed balance， 马尔可夫链就可以收敛[[12]](#footnote-12)了，我们可以根据detailed balance去sampling生成一些点，使这些点收敛到stationary distribution。因此这些点就是满足这个stationary distribution概率分布的点。收敛在这里称之为burning-in，在burning-in之前的若干迭代步骤生成的点被抛弃掉。下面我们再来验证下：

由于，因此两边同时对i求积分，也就是：

最后一步由于矩阵的每一行和=1（每个节点出度概率之和=1）。因此这个推导暗示了π\*=π\*⋅P

在这个例子里，我们取j=2，也就是，则



另外这样的 马尔可夫链还需要满足两个性质：1.irreducible， 马尔可夫链的所有状态节点需要可以彼此通信，不能有割裂的孤岛。2.aperiodic，非周期性，链条不会在特定的周期内在两个节点来回循环。

### Metropolis-Hasting算法

结合LDA算法，有了上述detail balance条件，受到这个平稳分布不再变化的启发，我们的**终极目标自然是：要使用一个 马尔可夫链条，sample[[13]](#footnote-13)出一系列的状态点，使其最终的平稳分布状态就是我们给定的那个联合概率分布（该联合概率就是LDA里的文档集被生成(generate)的概率）。**

**MH算法的目的：**是根据一个需求的(desired distribution)概率分布**P(x)**生成一系列样本状态点（因此，这个算法可以生成任意的概率分布）。为了达到这个目的，该算法使用 马尔可夫过程去到达一个平稳分布π(x)(stationary distribution)，以使π(x)=P(x)。

**MH算法推导：为了达到这个平稳分布，有两个条件需要满足：(1)满足detail balance：**（2）该平稳分布必须唯一（Ergodicity可遍历[[14]](#footnote-14)）。

MH算法的方式是设计一个马尔可夫过程（通过构造转移概率）来满足上述两个条件，用P(x)代替π(x)后，现在重点来分析detail balance：



我们的方法是进一步将转移概率分解为两个子步骤：proposal distribution(建议概率)和acceptance distribution（接受概率）。**建议概率**是说我们给出状态x后转移到x’的条件概率，而**接受概率**是接受状态x’的条件概率。

所以转移概率，代入公式(3.6)（detail balance），并整理可得：，也就是说这样我们得到了一个接受比率：“从状态x到x’的接受概率”与“x’到x的接受概率”的比率（接受率）。如果这个比率大于1，则我们按照建议概率转移到x’，否则停留在x原地不动（拒绝接受建议）。Metropolis的选择是进一步将上述接受率改为如下形式（比率大于1就=1，上限为1）：



(.)

接着就可以写出MH算法：

|  |  |
| --- | --- |
| **算法3.1 Metropolis-Hasting算法步骤** | |
| 1  2  3  4  5  6  7  8  9  10  11 | 初始化，随机选择一个初始状态点x  for i=0 to N-1 do //第i个sample点为x(i)  根据随机选择一个新状态x’  计算接受率α  生成0~1之间随机数u  if( or u<α) then{ //若α<1，按照α为概率接受x’  x(i+1) = x’  }else{  x(i+1) = x(i)  } |

MH算法非常简单，但要小心设计建议概率g，其实有时候考虑MH算法的特殊情况（Gibbs Sampling is a special case of MH）让问题变得更简单。

### Gibbs Sampling

考虑我们目标要得到的是一个维度是n的多维的概率分布，**如果采用设置多维概率分布P里的完全条件概率(full conditionals)作为建议概率(proposal)**，那么接受率就会始终=1，一直接受x’。让我们来证明这一点[15]：

full conditionals的公式如下：



(.)



**所以一旦在联合概率的full conditionals可用时，可以采用n维向量里轮流每一个维度循环的方式来迭代达到平稳状态。**

而为了得到full conditionals，就**要先写出联合概率（这也就引出了下一节的文本生成的联合概率）**，再用公式(3.8)的方式求得。

因此通用的Gibbs Sampling迭代算法如下所示：

|  |  |
| --- | --- |
| **算法3.2 Gibbs Sampling算法步骤** | |
| 1  2  3  4  5  6  7  8  9  10  11 | 初始化，随机选择一个初始状态点x1:n(0)  for i=0 to N-1 do{ //第i个sample点为x(i),x的下标j为x的第j维  sample  sample  …  sample  …  sample |

## 伟大的采样公式： Collapsed Gibbs Sampling采样公式推导

现在我们想从上述过程推导出Gibbs Sampling的采样公式，在做Gibbs Sampling公式推导之前最重要的一件事是写出**整个文本训练集生成的联合概率**（也就是上述3.2节过程乘起来）：



(.)

注意到第一项不包含α，第二项不包含β，所以两项可以分开处理。先来看第一项：也就是说给定topic的情况下的多项分布的似然函数（参照式(3.1)）：



(.)

这里的W代表语料中的所有词数，这些W个词在带有topic=zi的条件下的多次独立“多项分布”试验中被产生，我们现在将这一项分解为两个连乘，一个over topic，一个over vocabulary[[15]](#footnote-15)：



(.)

这里我们用代表topic k下单词t被观测到的次数，最终目标函数由对求积分得到。

这个式子的解释：首先不考虑超参数β，而是假设已知参数，这个就是那个K\*V维的矩阵，表示从每一个topic产生词的概率，然后把积分掉，仿照公式(3.5)的推理，再利用公式(2.11)，可以求出第一部分表达式了[[16]](#footnote-16)：



(.)

这个式子中的向量即代表：第k个topic下单词的分布情况，=(该topic下第1个单词的个数,该topic下第2个单词的个数,...)。我们发现这里对φ的积分的处理：φ被gamma函数integrating out而消失了[[17]](#footnote-17)。这里积分的变量是φ的所有可能值的集合，有点类似于边缘概率在离散分布上的处理，假设我们有概率p(a,b,c)，可以计算在所有c的可能值上求和，而仅计算

这个过程属于Dirichlet- Multinomial共轭结构。可以仿照这个推理过程，**也可以被推理得到**，将他重写为两个的乘积：



(.)

这里表示m th文档，表示m th文档下第k号主题词数（topic k被指派给词的计数）。为了integrating out ，[[18]](#footnote-18)仿照公式(3.12)的推理，同样利用公式(2.11)，我们做如下推导[[19]](#footnote-19)：



(.)

这里即代表：第m篇文档中的主题分布情况，=(1号主题词数,2号主题词数,...)。这也属于Dirichlet- Multinomial共轭结构。

所以联合分布可以得到：



(.)

有了联合概率分布，紧接着在此基础上就可以根据公式(3.8)推导full conditionals ，而现在我们已经讨论了推导最后的Collapsed Gibbs Sampling采样公式推导的全部要素，**唯有两点尚缺，这两点就是以下的事实**：

1. Gamma约去

利用



也就是说分子和分母里的gamma函数只差1时，如果分母较大，分子gamma函数内数字为分母-1，整个式子就等于。

1. 连乘号约去

由于full conditionals中包含分子分母，分子和分母的**唯一差别**只在与当前采样的第m篇文档第i个单词，所以其他无关乘积因子分子和分母皆因相等而约去。

当前第m篇文档的主题采样时：约去后为

当前第k号主题的主题采样时：约去后为

Collapsed Gibbs Sampling采样公式推导如下[[20]](#footnote-20)：







(.)

这一推导将永远是MCMC历史上的经典之作，它简洁、优雅，其关键之处还在于他利用了以下事实：Gamma函数在最终的公式中神奇地消失了。

gamma函数的出现有**两个作用**：

1. 换掉积分

2. 在分子分母同时出现gamma函数时利用gamma是阶乘的特性，而约去。

做完主题采样后，根据期望公式(3.4)就可以得到θmat(doc->topic)和φmat(topic->word)两个**重要的矩阵**：



(.)



(.)

## 总结

我们可以清楚的看到，在整个LDA迭代过程中，只是不断地对每个单词的topic编号重新指定，而(doc->topic)和(topic->word)两个概率向量是迭代训练完成后通过期望公式计算出来的。

**训练后如何验证模型质量的好坏，验证结果的正确与否(或者验证训练是否已经收敛)呢？这里使用了一个perplexity的公式：**



(.)

Perplexity的意义：b可以设置为2或e，其中H(q)就是该概率分布的熵。当概率q的K平均分布的时候，带入上式可以得到q的perplexity值=K。公式里的xi为测试文本，可以是句子或者文本，N是测试集的大小（用来归一化），对于未知分布q，**perplexity的值越小，说明模型越好。**

 我们以LDA简明扼要的盘子表述法结束本章。



图 3‑9 LDA的盘子图[[21]](#footnote-21)

这个图模型表示法也称作“盘子表示法”（plate notation）。图中的阴影圆圈表示可观测变量（observed variable），非阴影圆圈表示潜在变量（latent variable），箭头表示两变量间的条件依赖性（conditional dependency），方框表示重复抽样，重复次数在方框的右下角。

# 实现与应用

根据前面的推导，LDA的代码实现近在眼前，而Gibbs Sampling的LDA代码实现非常简洁，成为了机器学习界的典型案例。这一章，我分析java版本的JGibbsLDA代码(或Gibbs LDA++)后，庖丁解牛，分析该代码实现。

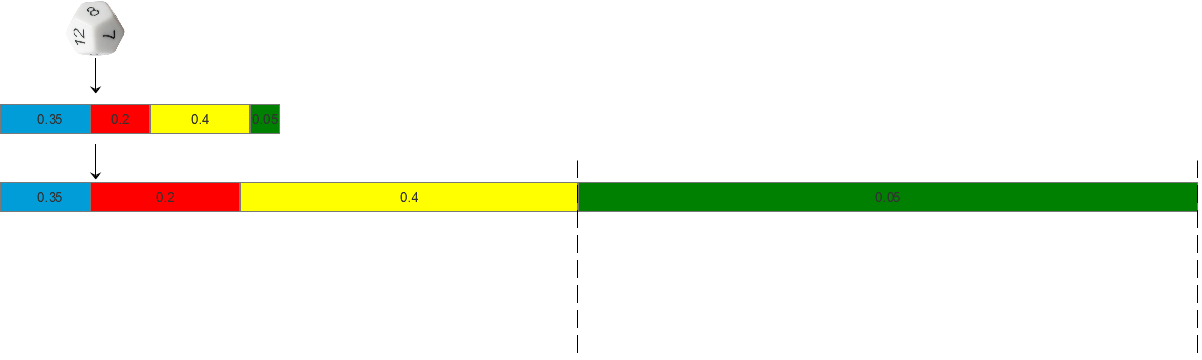
## 实现

在分析LDA的代码实现之前，我们先来实现一个小算法（该小算法可以作为面试题），题目如下：

**投掷骰子程序**：写出一个函数，骰子有K个面，各面概率不一，给定各面的概率，**请写出投掷骰子后，骰子被投掷到的面编号的输出**。这里可将骰子抽象为一个array数组(double类型数组)，这个数组内的元素为代表骰子各面的概率，函数逻辑为根据数组中各个index上的double浮点数概率，输出随机投掷到的那个array index编号：

|  |  |
| --- | --- |
| **算法4.1 Cumulative algorithm** | |
| 输入：double array[],array每项代表概率  输出：随机投掷到的array index | |
| 1  2  3  4  5  6  7  8  9  10  11 | double p[] = new double[length];  copy array’s element to p  for index from 1 to array.length do{  p[index] += p[index-1]; //逐项累积  }  double u = random double in [0~ 1] \* p[last element]  for index from 0 to p.length-1 do{  if (p[index] > u){  break  }  return index |

算法4.1在gibbs sampling中起到重要作用。该方法被称为“累积法”，将数组内概率先逐项累积，最后用该p数组范围内的均匀分布随机浮点数来随机投掷后，即可得到index值。



For循环逐项比对,直到第一项大于u

K面骰子各面概率不均

0~1的均匀随机数\*last elment size (scale up)

**double u**

Cumulative method

图 ‑1 cumulative method

我们紧接着来实现Collapsed Gibbs sampling的代码。[[22]](#footnote-22)

根据前面的推导采样公式(3.16)可以写出[[23]](#footnote-23)：

|  |  |
| --- | --- |
| **算法4.2 LDA Collapsed Gibbs Sampling** | |
| 输入：文档集(分词后)，K(主题数)，α，β，iter\_number(迭代次数)  输出：(doc->topic)和(topic->word)、tassign文件(topic assignment） | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | **申请几个统计量（int数组（或二维数组））：**   |  | | --- | | nw[][]:number of instances of word/term i assigned to topic j，size V x K  nwsum[]:total number of words assigned to topic j，size K  nd[][]:第i篇文档里被指定第j个主题词的次数， size M x K  ndsum[]:total number of words in document i，size M  z[][]: 是int二维数组[[24]](#footnote-24)，topic assignment for each word，  size M x per\_doc\_word\_len表示第m篇文档第n个word被指定的topic index |   **Initial阶段：**  for (m, doc) in doc\_set{ //m是doc编号  for word in doc{ //word均有wordid  topic\_index = random int from [0~ K-1]; 初始化阶段随机指定  z[m][n] = topic\_index; //将随机产生的主题存入z二维数组变量中  nw[word\_id][topic\_index] ++; //为相应的统计量+1  nwsum[topic\_index]++;  nd[m][ topic\_index]++;  ndsum[m]++;  }  }  //Initial初始化结束，后面的迭代使用了-1🡪采样公式重新分配🡪+1三重奏。  **Collasped Gibbs Sampling迭代阶段：（每一轮迭代有三重for循环）**  for iter in iter\_number{ //迭代iter\_number次  for (m, doc) in doc\_set{ //m是doc编号  for word in doc{  t=从z[m][n]中取得当前word的主题编号(初始化来自随机)  令nw[word\_id][t]、nwsum[t]、nd[m][ t]三个统计量均-1  double p[] = new double p[K]  for k in [0,1,2,…K-1]{ //从0到K-1号每个主题计算概率    //按照上述公式生成每个主题的概率存到临时概率数组p中    }  new\_t = 输入数组p到算法4.1 cumulative method随机投掷  令nw[word\_id][new\_t]、nwsum[new\_t]、nd[m][new\_t]三个统计量均+1  }  }  }  **输出阶段（也可以在迭代一半时候即可输出）：**  根据z数组输出文件tassign.txt  根据nw、nwsum、nd、ndsum套用公式(3.4)生成文件theta.txt和文件phi.txt  文件theta格式:大矩阵文件，M行K列  文件phi格式:大矩阵文件，K行V列 |

下面讨论几个实现时的注意点：

**1. read old train file:**所谓的model文件就是算法4.2输出的这几个矩阵文件，如果输出了这几个文件，中断训练后，下次再继续的时候：nw、nwsum、nd、ndsum这四个统计量均从tassign文件中可以读取得到。

**2. predict:**另外，如果已经训练过一个model了，如果有一篇新文档，需要对其predict（预测）新文档上的主题分布，可以利用已经训练的trn\_nw和trn\_nwsum用以下公式推断：



注意看这个公式，trn\_nw和trn\_nwsum相当于起到了伪计数(pseudo count)的作用，公式后一项因子只与**当前文档**的主题词数计数有关，故不必加入train的计数。此外，predict在不管是初始化还是迭代时只修改new\_nw和new\_nwsum统计量，其余采样过程都一样。[[25]](#footnote-25)

另外，-1 🡪 重新分配topic 🡪 +1三重奏我将其形象比喻为在采样每个单词前，从各个统计量中**抠掉**该单词（及其主题），然后重新分配。

再来用两张图，来说明：-1 🡪 重新分配topic🡪 +1三重奏：

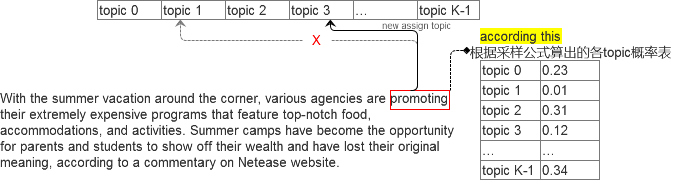


图 4‑2 -1 🡪 reassign topic 🡪 +1

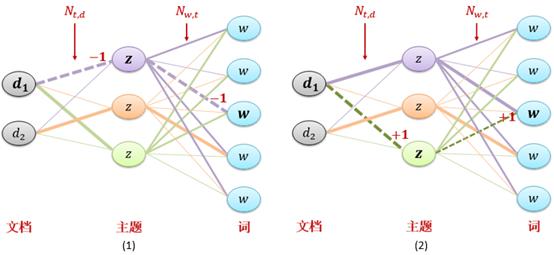
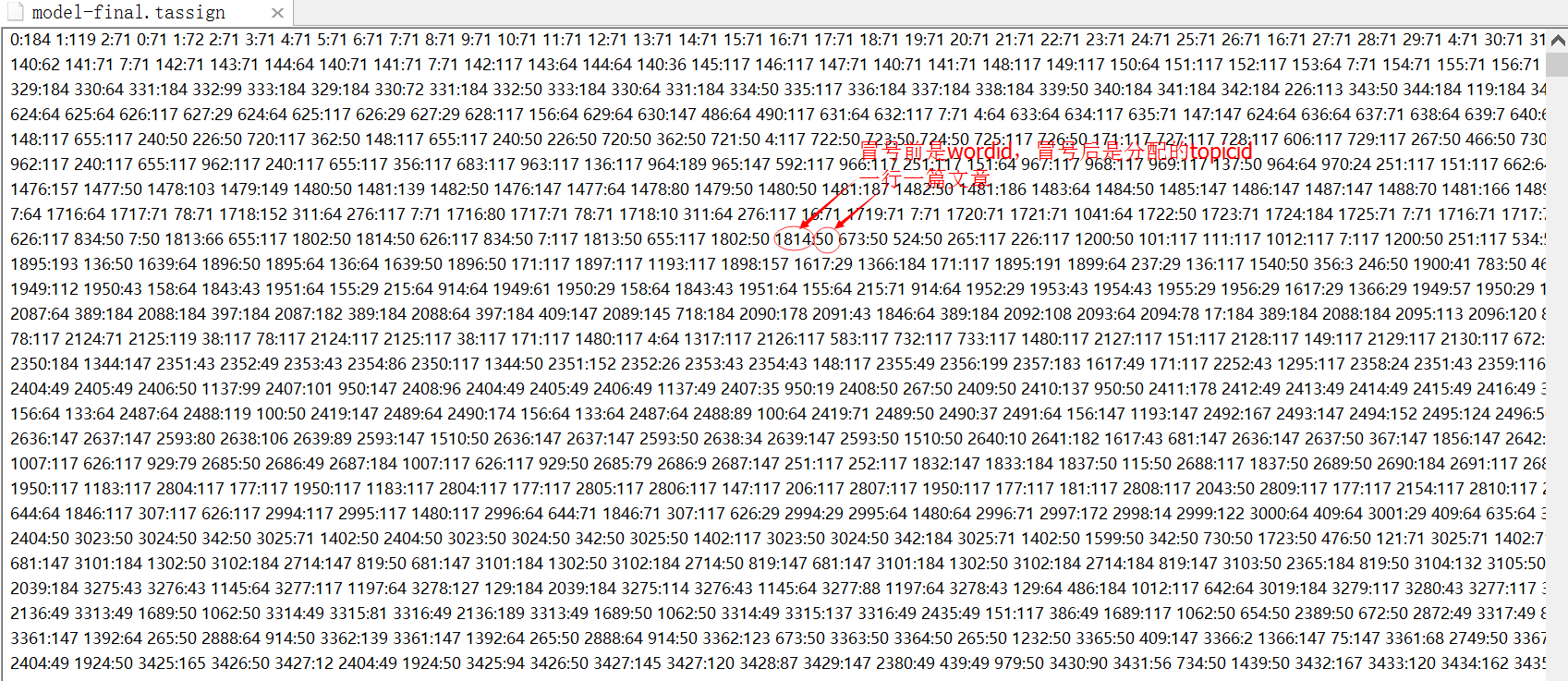


图 4‑3 -1 🡪 reassign topic 🡪 +1

JGibbs LDA或Gibbs LDA++输出的**文件样例**如下：

**1. tassign文件**（一行一个doc，冒号前是wordid,冒号后是topicid）



**2. theta文件**（矩阵文件：行号=doc idx，列号=topic idx）

|  |  |
| --- | --- |
|  | topic0 topic1 topic2 topic3 topic4 topic5 topic6 topic7 topic8 topic9 topic10 topic11 topic12 topic13 topic14 … |
| Doc1  Doc2  Doc3  … |  |

**3. phi文件**（矩阵文件：行号=topic idx，列号=word idx）

|  |  |
| --- | --- |
|  | word0 word1 word2 word3 word4 word5 word6 word7 word8 word9 word10 word11 word12 … |
| topic0  topic1  topic2  … |  |

**4. wordmap文件**（word🡪wordid）



**算法复杂度分析：**

**1. 时间复杂度**：Collapsed Gibbs Sampling LDA最大的特点是每次迭代内部要经历三重for循环，因此每轮迭代经历的循环数是doc\_num \* per\_word\_num \* K = corpus\_word\_count \*K，所以最终的时间复杂度=O(iter\_num \* corpus\_word\_count \* K)，这就说明了你启动时设置的主题数K越多，算法运行就越耗时间。

**2. 空间复杂度**：算法运行中需要几个统计量，耗费空间最多的有两个nw和nd，其中nw占用内存空间最大=主题数K \* vocabulary\_size(词典中unique单词数)，所以算法的空间复杂度=O(K \* V)。例如，考虑K=300，V=10万词，且都用int（4字节）来存储，二维数组nw需要多大的内存呢？



**参数设置：**

**topic number K：**许多读者问，如何设置主题个数，其实现在没有特别好的办法[[26]](#footnote-26)，只有cross validation，我的建议是一开始不要设置太大而逐步增大实验，也可以结合后文的topic rank方法，多个k值同时训练介绍后为topic打分排序，人工校验，看看垃圾topic是否较多。这个参数同时也跟文章数量有关，可以通过一个思想实验来验证：设想两个极端情况：如果仅有一篇文章做训练，则设置几百个topic不合适，如果将好几亿篇文章拿来做topic model，则仅仅设置很少topic也是不合适的。

**α、β超参数：**这个超参数其实作为伪计数，不是很影响算法的结果。

**迭代次数iter\_number：**gibbs sampling前面1000次甚至更多可以抛弃，这样会达到更好的效果。

## 应用

算法实用才更有价值，我将LDA称之为万能工具，在许多文本分析的角度都可以应用（甚至推荐领域），下面我们通过几个实用例子来谈谈如何应用LDA算法。选取应用的原则是：利用基本的LDA算法的输出即可做出、无需改造原有LDA代码的，这也是对读者要求较小的应用方式。

### 相似文档发现

这个方法可以被用作新闻推荐中，正文详情页的“相关推荐”，该方法所述的相似文档是指的“主题层面”上的相似，这就比其他的基于word来挖掘的相似度更有意义，整个方法非常简单：

1. **Hellinger distance**

该距离是度量两个概率分布差异的距离，如下所述（当然你也可以用KL距离等）：



(.)

由于前面的是常数，不影响排序，因此两个文章的相似度可以用下面的距离做出：



(.)

**(2)算法步骤**

由于LDA可以将文章表征成多主题分布的混合，所以利用上述公式，这样仅仅用了一个文件：theta文件，就可以分析出主题层面上的距离。

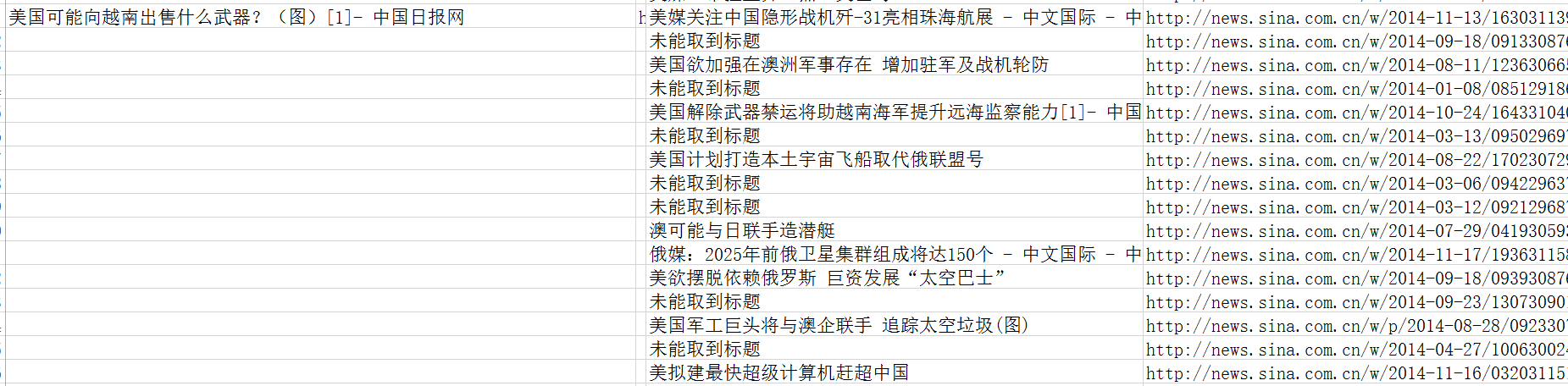
输入：LDA训练后的theta文件

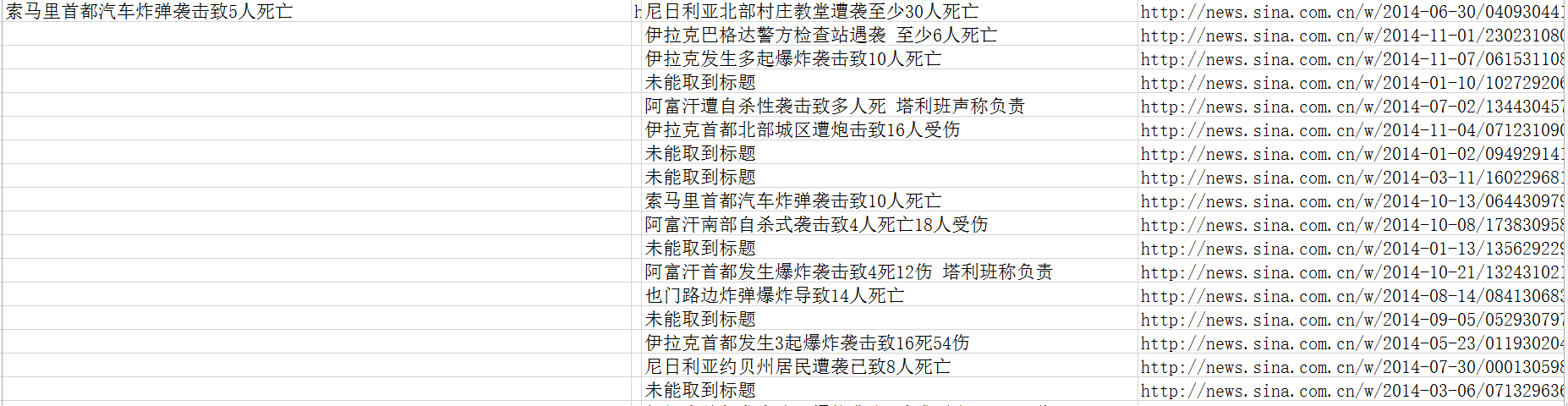
**步骤：**1. 读取LDA训练后的theta文件

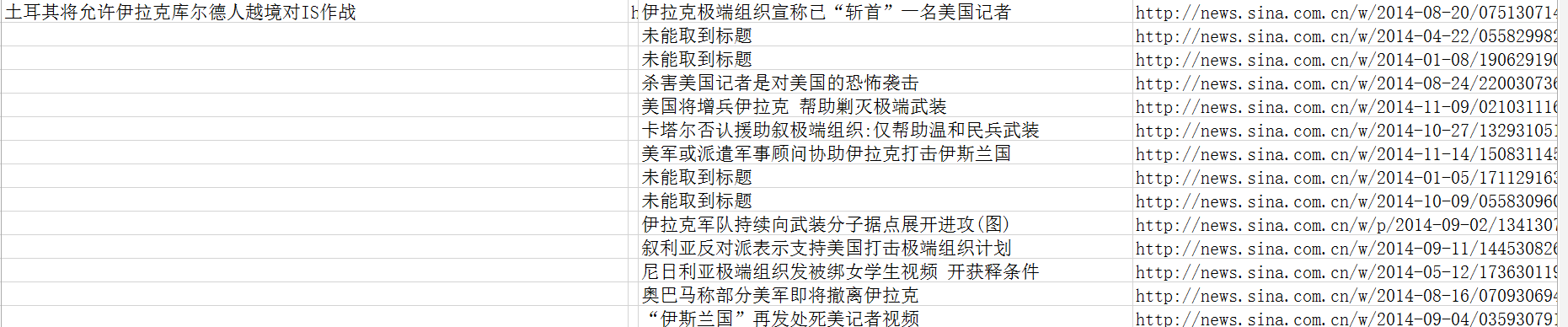
2. 为每个doc分别计算其他doc与该doc的主题距离（用上面的公式），再按照距离从小到大排序，输出每个doc最相关的前n个主题文章。

**(3) 结果演示**

比如ISIS和极端组织并字面上并不相等，但表达意思一致，在这种情况下，依靠基于LDA的办法就可以成功为其推荐出极端组织的文章，部分结果如下所示（截图左边为“正文页主文章”，右边一系列为与之主题相似的文章）：







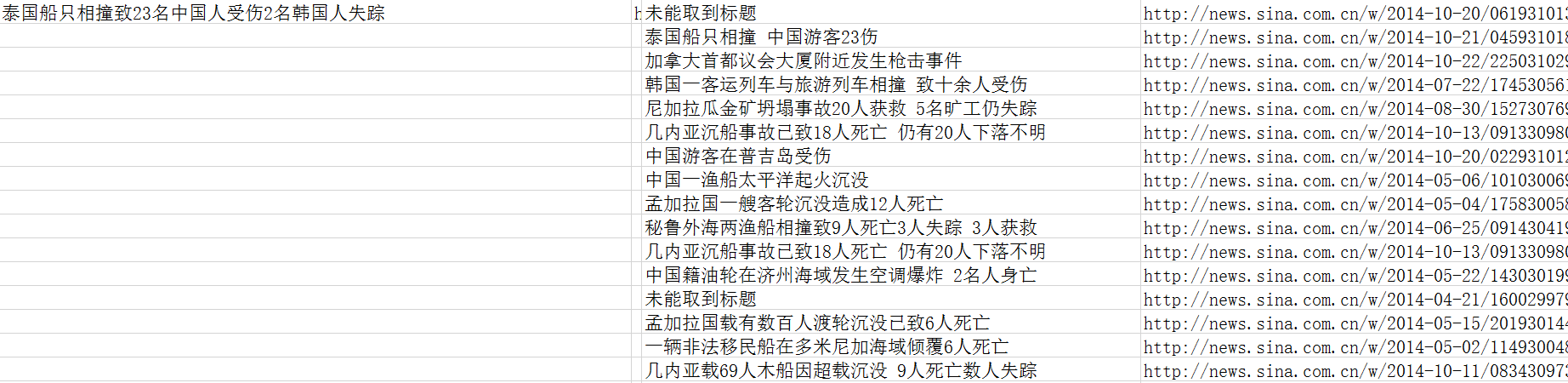


图 4‑4主题相似的文章推荐

### 自动打标签

利用LDA的训练结果做文章的简易自动打标签同样可以得到不错的结果，该方法的实现也异常简单：

算法输入：theta文件（doc🡪topic），phi文件（topic🡪word）

算法步骤：

1.分词（可以考虑使用stanford segment）

2.词性标注（可以考虑使用stanford chinese POS tagging）

3.去掉副词、介词等非实意词。

4.LDA训练

5.通过读取theta文件找出每篇文章概率最大那个主题，获得主题编号。

6.读取phi文件里每个主题下word概率，读入内存。

7.根据该文章最大主题编号找出**该文章下**该概率最大主题编号下的概率最大n个word词（max top n），（换句话说：该文章最大主题下的最大概率的n个词）作为该文章标签输出。

输出结果样例如下所示，可以看到，由于是同一个主题下最大概率的n个词，因此仅从标签表达意义就可以看出文章大体是讲什么的。



图 4‑5基于LDA打标签

### LDA与LR(逻辑斯蒂回归)结合做新闻个性化推荐系统

常见的个性化新闻推荐手段有基于关键词的，基于新闻频道栏目的等，但是基于关键词的由于粒度较细，不太容易满足推荐系统的“多样性”，而基于喜爱栏目（比如NBA）的个性化推荐粒度又太粗，所以就比较有必要研究基于主题的推荐系统了。

下面我分别介绍两种方法为用户推荐用户喜爱主题的新闻文章。

**方法1：LDA + LR**

这个方法基于这样的假设：用户从新闻列表页点击的新闻时，肯定是对该新闻较为喜欢，而上下相邻的几条新闻未点击，肯定用户的眼睛看到了而未点：表明不喜欢。

**算法步骤：**

1.收集点击日志：

用户在一个新闻网站上（比如一个门户）通常的点击情况如下图所示，我们需要收集下用户在列表页点击的文章和用户未点击的前几个和后几个文章（收集时需要做标识区别）。

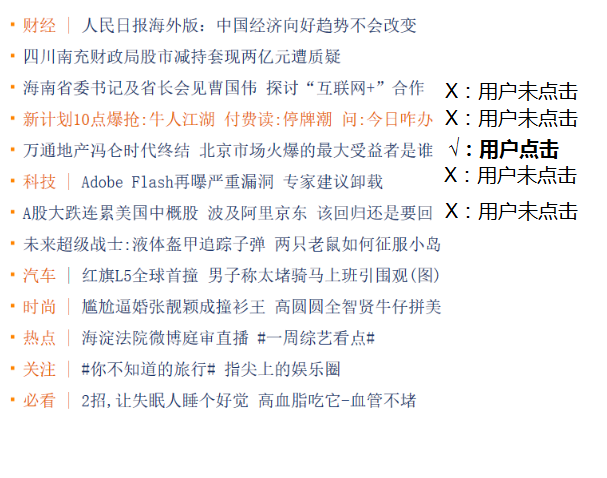


图 4‑6需要收集的用户点击的item条目和上下几条未点击

2. 使用LDA为新闻文章做主题维度特征提取



图 4‑7将文章用LDA特征提取(theta文件)

3. LR算法使用log点击数据做训练[[27]](#footnote-27)

将文章通过LDA训练后降维为K个主题（比如200个主题后），每个主题的概率为一个小于1的浮点小数。在利用前面的点击log数据，就可以将**每个用户点击标为正样本，用户未点击为负样本**。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **docid** | **用户点击** | **Topic1** | **Topic2** | **Topic3** | **…** | **TopicK** |
| 842 | 0 | 0.12 | 0.02 | 0.42 | … | 0.2 |
| 886 | 0 | 0.42 | 0.15 | 0.08 | … | 0.03 |
| 822 | 0 | 0.14 | 0.52 | 0.05 | … | 0.26 |
| 981 | 1 | 0.05 | 0.2 | 0.17 | … | 0.47 |
| 417 | 0 | 0.49 | 0.3 | 0.17 | … | 0.23 |

表 4‑1某一个用户的LR训练数据(1为点击，0为未点击)

紧接着就可以使用LR训练出该用户对K个topic里每个topic上的喜好权重weight了，再用这个weight向量对**每篇新的新闻文章**使用线性加权公式: doc\_score = w1 \* topic1 + w2 \* topic2 + …，从大到小按照score排序后，可以立即为用户提供个性化推荐。

**注意**：考虑到通常很多网站用户粘性没有那么大，一个用户的点击记录不可能很多，可以先使用用户聚类算法将用户聚类成一组一组，然后**一组用户作为训练集**训练topic喜好程度的weight权重。

**方法2：user profile记录喜好topic法**

算法步骤：

**1.提取topic：**文章LDA训练后的theta文件，提取每篇文章概率最大的前3个topic主题

**2.save topic🡪user profile：**当用户访问文章A后，就把文章A的top 3的topic贴到用户兴趣档案里，并将概率分值相加，如果过了1天用户没有访问网站或访问这个topic的文章，就按照该topic乘以0.8衰减，直至衰减到0。

**3.产生推荐：**下次用户再次访问网站时，从用户兴趣档案找出用户最感兴趣（分值最大）的k个topic，然后选取这几个topic下热门的文章为用户推荐。

### topic rank[10]

由于LDA输出的主题编号是随机的，主题也是未经排序的，因此可以开发一个为每个主题打分的算法，将显著的、有特色的主题打分高往前排，而将垃圾的、意义不明的主题往后排，这种排序算法也是很有意义的。

算法步骤：

该算法总体而言核心思路就是计算当前topic的概率分布与**垃圾分布**的距离。

**1.垃圾topic分布分为两种，准备2个距离：**

(1) topic🡪doc：由于LDA算法运行出的theta文件是doc🡪topic的概率分布，我们需要转换成topic🡪doc的概率分布，所以k号topic的第m篇文章的概率为，(分母归一化)，Topic在doc上的概率向量（其中D是语料集的文章总数），衡量其与下面这个背景噪音向量的距离（使用KL divergence、1-余弦、1-皮尔逊相关度等），距离越远分值越高，主题越有特色。

(2) topic🡪word：使用phi文件的topic🡪word的概率分布，计算当前topic的phi向量与向量的距离。

**2. 加权排序**

将上述几个当前topic向量与垃圾向量的加权求和后排序。

final\_score= a \* scoretopic->doc + b \* scoretopic->word即可（\*符号为乘法）。

为了验证这个算法，我专门设计了一个页面，并且为每个主题人工标注了一个名字，如下截图：

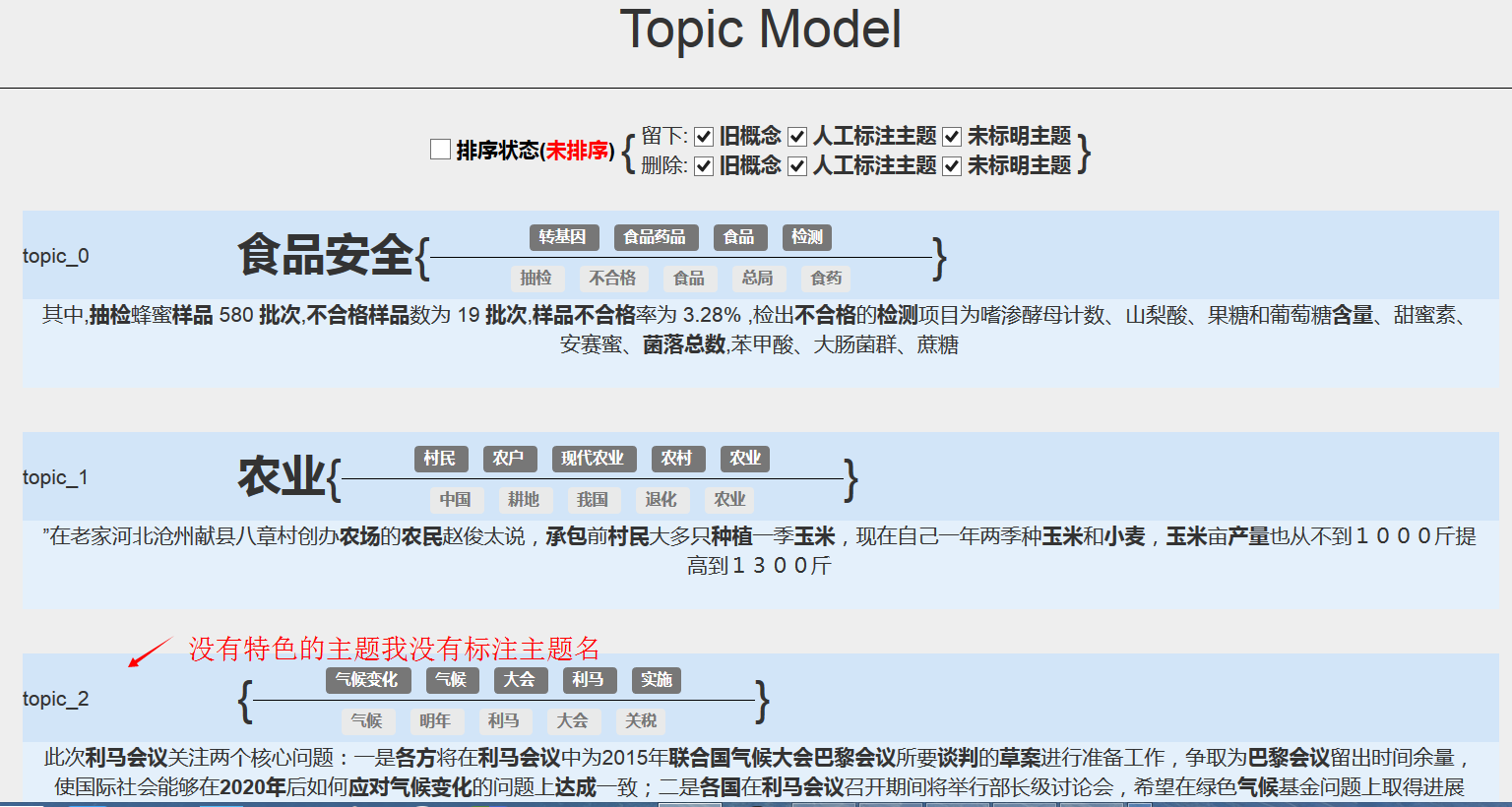
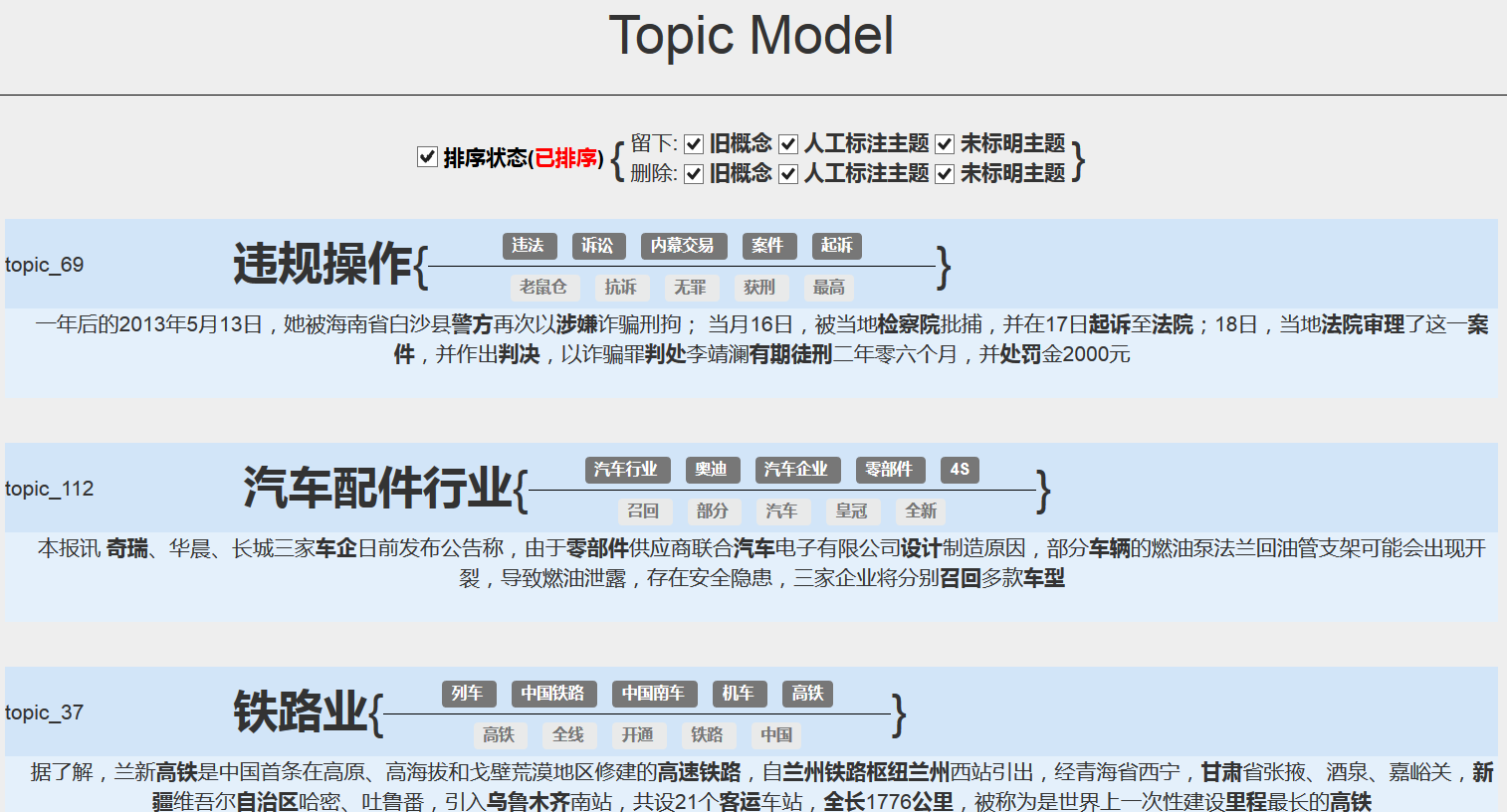


图 4‑8 rank排序前的主题



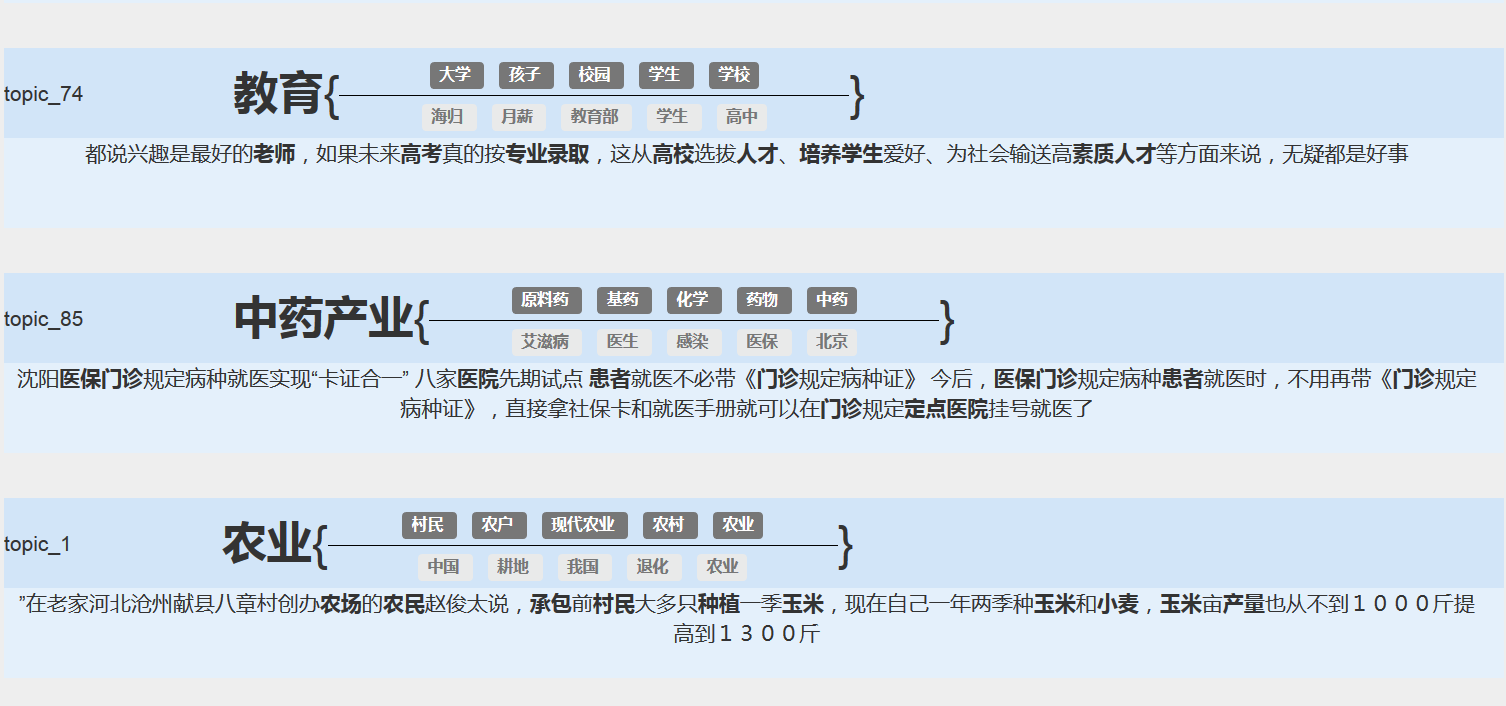


图 4‑9排序后的前几位高分topic



图 4‑10 排序后低分的垃圾topic

可以看到在排序后，出人意料地将人工标注的名称的特色topic打分较高（排序后的前几位均有名称标注），而排序后分值最低的垃圾topic则普遍没有标注名称，这说明人工挑选的topic与算法打分结果相一致。

### word rank

仿照topic rank的思路，就可以做出word rank[[28]](#footnote-28)，首先考虑下word rank有什么用，在哪些场景有使用，再给出基于LDA的word rank算法

word rank的应用场景之一：**协助文本分类器**

比如在淘宝网搜索时，经常需要“分类搜索”的功能，如下截图：

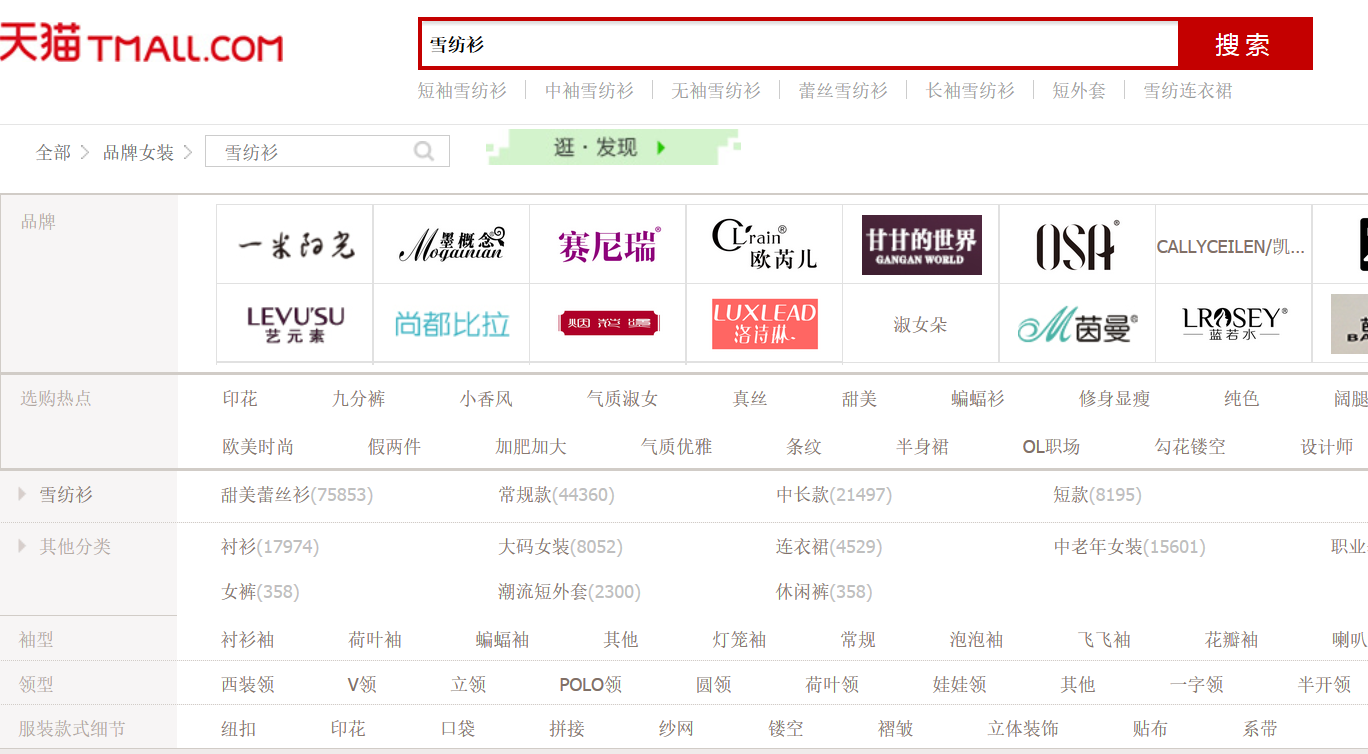


图 4‑11分类搜索

现在我们需要按照店家给出的商品标题描述分类，但是，如果你仔细观察店家给出的商品标题，会发现如下情况：



图 4‑12商品的标题描述会误导搜索分类

如果我按照传统的分词后TF-IDF提取word特征的方式分类，就会出问题，比如上面这个例子中，店家为了增加他们被搜索命中的机会，通常在标题上填写很多重复冗余无用的信息，比如图上的标题中“套头”这个词的意思是：没有扣子或者拉链的，必须从头上套着穿的。但是这个词是不能用作分类的依据的，搜索时我只想按照商品的主要特征词来分类，而非“套头”。

如果有一个方法可以获取到每个分类下的主要特征词，而忽略每个分类下冗余词，那么就可以正确的分类了。所以**步骤**是：1.根据印花、真丝等类别词匹配(match)找出相应的商品的文本；2.**每个分类**执行word rank算法，为每个词打分，过滤掉分值最低的垃圾词；3.word rank的分值作为SVM/LR等分类器的特征依据。

下面的算法步骤详述了word rank算法：

1.计算word在每个主题上的概率分布，得到向量

读取phi文件，由于phi中的概率值是topic🡪word 的，而我们需要的是word🡪topic的反向关系，因此使用，其中每一项 ，也即是将 phi 文件矩阵转置后归一化(这里的大写K是主题总数，c是当前词)。

2.计算垃圾/噪音word向量

，每一项都是1/K的平均值。

3.使用KL距离度量两者距离

可以采用1-余弦相似度，或KL距离，或几种距离度量的平均值度量两者距离，最后均一化，距离越大的分值越高。

我以汽车新闻类型的文章为案例，用该算法输出样例：

|  |  |
| --- | --- |
| word rank得分较高的word | word rank得分较低的word |
| 瑞虎 0.919845109474  专车 0.919889049627  奔腾 0.920182327547  吉利 0.920315761625  菲翔 0.920482668896  凯迪拉克 0.920863099045  长城 0.920909053849  本田 0.920911274423  福克斯 0.921148048364  宝骏 0.921434482516  荣威 0.921925139295  路虎 0.922897265021  瑞风 0.922900276469  捷豹 0.923057646905  景逸 0.923665401422  海马 0.924589687164  奔驰 0.924722328098  沃尔沃 0.92505266906  南通 0.92513072146  东风标致 0.925149629944  哈弗 0.925476645704  雷克萨斯 0.926134979376  马自达 0.926283860082  起亚 0.926606816176  奥迪 0.926743050409  昆山 0.927461877372 | 悍马 0.0163962476521  加倍 0.0181429299963  韦伯 0.0183433086821  充充电 0.0185650744705  假定 0.0187398834494  程惊雷 0.0188122305783  六百 0.019123897586  投运 0.0191464581692  一臂之力 0.0191952400808  欧阳明 0.0194425138855  发卡 0.0194874355798  教育机构 0.0195078269565  交卷 0.0195195159147  污染天气 0.0195711744352  于海霞 0.0195859600484  源点 0.0195859600484  暂住证 0.0195920128674  一概而论 0.0196311823389  刘理萌 0.0196311823389  长信 0.0196311823389  语气 0.0196637204583  娄格嘉 0.0197160360731  购物网站 0.0197221613537  私企 0.0197550919891  文德 0.0197550919891  奸商 0.0197611161447 |

表 4‑2 word rank汽车文章例子

可以看到得分较高的word大部分均为汽车品牌，大部分的得分就较为准确，经过测试，平均可以达到80%的准确度，但得分低的里面“悍马”的分就太低了，这就表明了此算法仍有优化空间（留给读者，可以结合topic rank的score进行优化）。

### 文章质量评分算法

这是迄今为止，我发现的基于原始的LDA算法训练结果就能产出的最强大的技术之一。说的通俗点，这个文章质量评分算法就好比“计算机程序自动为高考作文判分”。其实将topic rank思路扩展开，便可以延伸出此技术。

比如博文推荐方面，由于博文质量由用户自主撰写而成，良莠不齐，因此需要一个博文质量自动评分系统。在经过LDA算法对文本的训练后，可以根据topic主题在当前文章的分布情况以及文章中各个词word在主题topic上的概率分布信息开发出一个新的基于LDA的文本质量评估算法。这个算法对比以往的基于SVM分类器等方法有**几个关键优势**：1.该算法是无监督的，不需标注数据，2.该算法可以为每篇文章产生一个分值，而这是二分类的分类器做不到的。

该算法的步骤是：

1.先找出该文档生成的概率分布即如下公式：



(.)

在这个公式中等式左边的向量是指该文档的生成概率，其中每一项指的这个文档的一个词在该主题上产生的概率 \* 文档中该主题的概率。这个式子里的即，而即，注意到w1~wN是指从词典中编号1直到N的所有词，如果该文档d中不含单词编号i的词，则向量中编号i位置的概率为0。

2.生成这个向量后，与“众文档平均概率向量[[29]](#footnote-29)”（Mass Article probability vector）的概率进行距离度量，其中，“众文档平均概率向量”这个向量由如下公式得出：



(.)

其中p(k)表示k编号主题在整个语料库（corpus）中的概率，可以由如下公式得到：



(.)

其中，θd,k表示每个文档中主题k的概率值。分子是指语料库(corpus)的所有文字的k编号主题，而分母的部分相当于是语料库所有文章的所有θ之和。

3.使用KL、cosine、pearson correlation coefficient三种相似度/距离度量标准公式（criterion）衡量上面这两个向量之间的距离。



(.)



(.)



(.)

距离算出来后，再使用



(.)

来将三种不同度量标准的距离重新归一化到0~1之间。越接近“众文档平均概率向量”的文章质量就越高，反之质量就越垃圾。最后使用



(.)

返回这三种距离的平均值作为对文档的评分分值（区间0~1之间）。

先以旅游博客作为打分测试集，整个质量打分算法运行结果如下文件所示：

|  |  |  |  |
| --- | --- | --- | --- |
| **旅游博文评分较高的10篇文章** | | | |
| **网址url** | **评分** | **阅读量** | **文章标题** |
| http://blog.sina.com.cn/s/blog\_53bd0d1701015f4l.html | 0.998066008917 | 61 | 山西徒步路线 |
| http://blog.sina.com.cn/s/blog\_69cb346901017r65.html | 0.991291044954 | 697 | 关于海洋航行者号“神户”“长崎”“济州岛”上岸观光的攻略之毛毛雨 |
| http://blog.sina.com.cn/s/blog\_65376d0501013ikj.html | 0.981205704494 | 17 | 最全全国旅游景点逃票大攻略 |
| http://blog.sina.com.cn/s/blog\_459154f301017xz3.html | 0.979939083754 | 189 | 【心无羁，行不止】独家奉送新鲜出炉西藏10DAYS跟团+拉萨自由行攻略-近3万字吐血整理你读完绝不会失望 |
| http://blog.sina.com.cn/s/blog\_6d4f6b220101b8cx.html | 0.979059053598 | 216 | 东欧德奥匈捷斯五国十日游记[无图] |
| http://blog.sina.com.cn/s/blog\_45e5ae5a01014mry.html | 0.978861793044 | 110 | 《康藏朝圣之旅》&nbsp;第三篇&nbsp;魂牵拉姆纳错&nbsp;&nbsp;（转载） |
| http://blog.sina.com.cn/s/blog\_3d4ceed401017a48.html | 0.976334174062 | 278 | 2012莫斯科、圣彼得堡七日游之流水账 |
| http://blog.sina.com.cn/s/blog\_4a0ab71d01014per.html | 0.975317724569 | 526 | 美国两周游游记（纽约、新泽西、堪萨斯、拉斯维加斯、洛杉矶） |
| http://blog.sina.com.cn/s/blog\_51688d58010168sk.html | 0.974945350427 | 69 | 2012年十月金秋，走入绚彩额济纳,绝美胡杨林 |
| http://blog.sina.com.cn/s/blog\_3e1b470f010179ve.html | 0.972611234889 | 191 | 泰国-大马一月游 |

表 4‑3评分较高的10篇文章

|  |  |  |  |
| --- | --- | --- | --- |
| **旅游博文评分较低的10篇文章** | | | |
| **网址url** | **评分** | **阅读量** | **文章标题** |
| http://blog.sina.com.cn/s/blog\_a2621cf20100yri6.html | 0.0825872 | 0 | 米径3公分30元/株 |
| http://blog.sina.com.cn/s/blog\_a259d83e01014lkt.html | 0.0818101 | 2 | 米径2-3-4-5-6-8-10-12公分 |
| http://blog.sina.com.cn/s/blog\_a2d9fcda01012twf.html | 0.0817654 | 0 | 16-60-190-260-360-760-1200元/株 |
| http://blog.sina.com.cn/s/blog\_7308354b01018uzp.html | 0.08163 | 437 | 神道八百万神 |
| http://blog.sina.com.cn/s/blog\_9e05cefd01013gch.html | 0.0816215 | 5 | 2-3分枝地径2公分高180公分4元/株 |
| http://blog.sina.com.cn/s/blog\_a50c721201015smj.html | 0.0683244 | 3 | [原创]www.pc-ge.com湖南阳光板批发&nbsp;阳光板价格阳光板价格PC阳光板 |
| http://blog.sina.com.cn/s/blog\_71663f9201016cr7.html | 0.06791 | 474 | (23)四川律师通信录&nbsp;Szechwan&nbsp;Lawyers&nbsp;Contacts |
| http://blog.sina.com.cn/s/blog\_4d56fe5b01012ypz.html | 0.0612046 | 59 | 解放前测绘资料目录-甘肃 |
| http://blog.sina.com.cn/s/blog\_a54beb4601015n17.html | 0.0595885 | 4 | [转载]www.pc-ge.com湖南阳光板批发&nbsp;2 |
| http://blog.sina.com.cn/s/blog\_a574e2c801013dgd.html | 0.0220331 | 20 | 一色的青瓦，飞檐碉兽，古香古色，看起来如同沷墨山水 |

表 4‑4评分较低的10篇文章

可以看到打分最低的“垃圾文章”有以下若干种类型：

1．批发货物的牛皮癣广告，如《[原创]www.pc-ge.com湖南阳光板批发 阳光板价格阳光板价格PC阳光板》；

2. 各种名词列举的博文，如<http://blog.sina.com.cn/s/blog_4a708d1301014gvx.html>，《互为近义词的成语大全》；还有《河南花木报价表，各种花木价格表》；

3. 简单列举的景点，如<http://blog.sina.com.cn/s/blog_577b695a01017bbd.html>，《中国国家级自然景区名录》；

4. 航班和汽车等时刻表，如<http://blog.sina.com.cn/s/blog_5776fab60100zeue.html>，《成都双流国际机场完整航班时刻表》等。

**实时接口（online learning）：**

由于LDA训练需要时间，如果我们需要一个实时接口，实时为新文章打分，可以采用以下策略：先将LDA的model训练完成后，将**nw（稀疏矩阵）、平均概率向量、wordmap、距离最大值和最小值（用于均一化）存于类似redis的数据库cache下来，此为train模型的记忆。**在线上文章分词后实时传送给接口后，连接读取redis中的模型记忆，使用前面介绍的predict方法为新文章预测主题分布后，再按照上面批量打分的方法衡量距离打分即可。

另外可以进一步挖掘一下文章的“阅读点击量”和“质量评分”之间是否存在关系，但是博客文章的点击阅读量(PV)经常与作者知名度、作者粉丝多少有关，因此可以按照区间平均点击阅读量(PV)来看两者的关系：

|  |  |  |
| --- | --- | --- |
| 分值区间 | 平均pv | 文章数 |
| 0.9~1.0 | 449.841 | 390 |
| 0.8~0.9 | 588.2538 | 2975 |
| 0.7~0.8 | 399.3124 | 3073 |
| 0.6~0.7 | 341.1599 | 2277 |
| 0.5~0.6 | 167.3782 | 3038 |
| 0.4~0.5 | 149.2586 | 1655 |
| 0.3~0.4 | 137.2095 | 797 |
| 0.2~0.3 | 71.75649 | 501 |
| 0.1~0.2 | 76.86979 | 192 |
| 0~0.1 | 58 | 20 |

表 4‑5区间平均PV与文章质量打分的关系

图 4‑13区间平均PV与文章质量打分的关系

可以从上图看到，按照每0.1为一个区间进行统计文章的平均pv，会发现存在一定关系的，即：文章打分高的点击阅读量相对来说更高，文章打分最低的点击阅读量相对来说更低。基于这一点，可以有理由将文章质量打分算法融入推荐系统，过滤掉垃圾文章，提高点击率。

当然该方法也可用于“没有标注训练集的文本垃圾过滤”等场景中。

### 总结

除了上面提到的应用外，LDA还有许许多多其他的应用，限于篇幅有限，就不一一展示了，算法优化无止境，上面提到的应用都有很多优化余地，这些就留给读者完成。

# 并行化

通过上一章的实现可以看到，LDA算法运行的时间复杂度还是比较高的，跟主题数K和训练语料数据量都有关，如果主题数过多，更是会造成训练时间的大幅上涨。因此如果训练文本语料数据或主题数过多时，有必要开发并行计算版本的LDA。

使用Gibbs Sampling版本的LDA开发并行化，存在着一个**主要难题**，就是nw、nwsum、nd、ndsum这几个统计量的并行更新问题，更具体地说，如果简单地将文章集(doc set)拆分成若干份并行计算，A进程和B进程同时启动时，由于每次重新指定主题(reassign topic)后，都会修改掉统计量（-1 🡪 重新分配topic 🡪 +1），因此就会造成修改读写冲突，破坏统计量的一致性。**3个依赖**：(1)从列上看：不同文章并行化时，某文章的单词a“依赖于”另一篇文章相同单词a采样后修改的nw和nwsum；(2)从行上看：同一篇文章的后一个单词“依赖于”前一个单词采样后修改的nd和ndsum。(3)从主题层面看，同一个主题的后一次采样“依赖于”同一个主题前一次采样的nwsum。

## AD-LDA

为了解决这个问题，David Newman等人提出了AD-LDA算法[12]，这种方法将文章分散到不同的机器上（**按行拆分**），该算法由于可以被转换成map-reduce操作，因此早期的LDA并行化实现都采用AD-LDA，该算法的核心是提出了global update操作（解决依赖1），整个步骤如下所示：

|  |
| --- |
| 算法5.1 AD-LDA算法 |
| 算法步骤：   1. 将训练集中的D篇文章分散到P个机器上。并按照以下规则将统计量分散到各个机器上：  |  | | --- | | nd、ndsum：这两个统计量仅与文章有关，拆分到各个机器上；  nw、nwsum：这两个统计量被完整copy到各个机器上，而不是拆分。  我们用nw\_p、nwsum\_p代表各个机器上的copy版本，注意到这个数组与全局占用内存一致。 |  1. 在各个机器p上，使用nw\_p,nwsum\_p,nd\_p,ndsum\_p执行原始单机LDA算法（各个机器互不知晓对方存在），注意几个统计量都被替换为了copy版本，即(nw\_p=nw，nwsum\_p=nwsum)，各机器中的统计量（nw\_p等）独立修改，其余过程与单机LDA算法（算法4.2）完全一致。 2. merge back:在各个机器分布式Gibbs Sampling运行结束后，z\_p数组已经被指定topic完成。执行global update操作：   [[30]](#footnote-30)  nw是所有进程在Gibbs Sampling之前的统计量（全局nw），在各个机器上迭代结束后，nw\_p被合并回了全局统计量nw，紧接着，全局nwsum(topic’s word count)也可以被计算出来：     1. 一轮迭代结束，转第2步 |

注意到这个算法被看作单机版本的Gibbs Sampling的近似(approximate)，因为在各个机器上不同进程启动Gibbs Sampling时，nw\_p和nwsum\_p互不知晓其他机器的存在而进行采样，就会造成前面提到的修改冲突，破坏统计量一致性，所以说这一步其实已经产生了误差，不能完全等同于单机版的Gibbs sampling，但其后的global update在每一轮迭代后都将这个问题尽可能地修复了。

**问题：**内存问题——AD-LDA算法仍然未能解决耗费内存最大空间的统计量nw的问题，因此如果词典vocabulary和topic数量过多时，仍然在每个机器上都会耗费较大内存，而由于单台机器的内存有限，尤其在Hadoop等对每个job内存限制时更不可接受。

## spark-LDA

由于存在上述问题，邱卓林研究了一个基于spark的LDA分布式方法[[31]](#footnote-31)，该方法虽然题目是基于spark的，但其实是个LDA的分布式Gibbs Sampling的一般性的原理，可以被实现在任意并行框架中（例如MPI），该方法克服了AD-LDA占用内存大的问题，并且误差比AD-LDA更小。算法用到的变量如下：

|  |
| --- |
| X：数据集  XP ：切分后的第p份子数据集  nd：全局document🡪topic count统计量  nd\_i：nd的第i部分  nw\_j：nw的第j部分  nwsum：topic k被指定的单词总数  nwsum\_p：在第p份子块中sampling后重新统计的topic k的单词总数 |

表 5‑1 spark-LDA变量含义

算法步骤如下：

### 切分块

Spark-LDA类似于AD-LDA，也是将原始训练数据进行了切分后分散到各个机器上去执行，但是进一步考虑了列切分，由于LDA的采样中可以**不考虑**同一篇文章中单词出现的顺序，所以只要没有统计量更新依赖，就可以把文章像切蛋糕一样切分开并行执行。故这个算法的**关键思想**在于训练语料文本矩阵中**同一行(同一篇文章)**或**同一列(同一个单词)**的word都会产生**统计量更新依赖**，所以**同一行或同一列**的数据不能被同时sampling采样。（这里的行列是指每篇文章排序后的wordid）如下所示：

|  |  |  |  |
| --- | --- | --- | --- |
| **doc编号** | **文章原始内容** | **转换成wordid后** | **对wordid排序** |
| doc m | He published several books, including Chromic Salts Technology, Rejuvenating Chemical Industry through Science and Technology, Soft Science and Reform, Large Linear Target Programming and Application, Research in China's Economic Development and Reform and Economic Reform and Development in China. | 306 310 482 326 497 403 400 323 375 394 392 371 415 343 417 316 329 440 417 484 436 305 444 392 463 312 315 425 | 305 306 310 312 315 316 323 326 329 343 371 375 392 392 394 400 403 415 417 417 425 436 440 444 463 482 484 497 |
| doc n | In recent years, he devoted himself to the use of complexity science to study issues relating to the development and reform of China, made enormous efforts to explore and explain the characteristics and law of development of the fictitious economy and actively studied and promoted the development of venture capital in China. | 444 312 384 306 458 372 471 424 466 359 406 415 490 476 496 312 343 315 379 393 454 454 447 361 397 325 336 482 374 495 338 492 393 425 | 306 312 312 315 325 336 338 343 359 361 372 374 379 384 393 393 397 406 415 424 425 444 447 454 454 458 466 471 476 482 490 492 495 496 |

表 5‑2先将原始训练文章转换为排序后的wordid

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Doc\_m | 305 | 306 | 310 | 312 | 315 | 316 | 323 | 326 | 329 | 343 | 371 | 375 | 391 | 392 | 394 | 400 | 403 | 415 | 417 |
| Doc\_p |  | 306 | 310 | 312 |  |  | 323 |  | 329 | 343 |  |  |  | 392 |  |  | 403 |  | … |
| Doc\_q |  | 306 | 310 |  |  |  | 323 |  |  |  |  | 375 | 391 |  |  | 400 | 403 | 415 | … |

表 5‑3每篇文章排序后wordid，原训练语料🡪矩阵

上表是将原始语料转换为排序后(sorted)的矩阵后（并且对齐后），同一行和同一列的wordid在sampling时会产生统计量依赖。空的格子表示该文章没有这个词而被对齐留空。以上表中323为例，**画十字线**的上的单词均不可被同时并行执行sampling。

经过分析，整个算法的步骤：可以先将原始数据X划分为P\*P的份，这里的P是我们设置并发数，如下图所示的P=3，每一份数据都相应地标注了坐标，注意到由于十字线上(同一行列)不可以被同时执行，因此下图的执行分为几个阶段执行，对角线上的(0,0)、(1,1)、(2,2)这3块数据先被同时执行，而每一块内部执行单机版的Gibbs Sampling，这三块并行执行结束后，再执行另外不在十字线上的3块(0,1)、(1,2)、(2,0)，这三块执行结束后最后执行(0,2)、(1,0)、(2,1)三块（如下图右，三行行内并行计算，行间串行计算）：

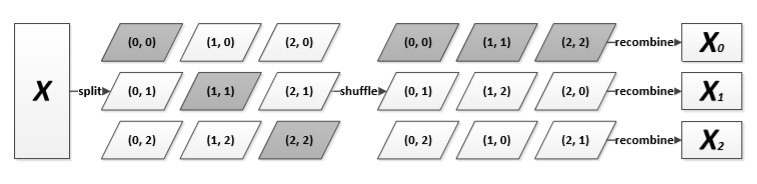


图 5‑1 spark-LDA切分流程，并发度3

这里的“一块”的概念是包括多行和多列的，我们希望每块的数量都均等，这样在每台机器上才能负载均衡。但在切分时候由于“对齐”会造成空的word格子的原因，所以，简单地按照列数均分和行数均分来切分是不行的。为了尽可能每块均等，可以分割多次，然后挑选出各个块之间**差距的最大值最小**的那一次。

**切分策略：**(1)首先将数据集X按照列切分成均等的P份，如果包含的单词数不均等，可以随机交换列再切分来保证均等。（2）在将切好的P份列，切成P份行，如果不相等，多次**随机交换行**，最后挑选出各个块之间**差距的最大值最小**的那一次。

### 选择

切分成P \* P块完成后，就是挑选出哪些无冲突的P个块可以同时被并发执行（组内并行，组间串行）。通用的策略有2种：

1. **八皇后法**

八皇后问题是一个经典的问题，在8X8格的国际象棋上摆放八个皇后，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上，问有多少种摆法。所以这里的挑选无冲突块类似于八皇后摆棋子，我们这里的情况是弱化的八皇后问题，即挑选的块可以出现在同一个斜线上。八皇后的经典解法是回溯法**[[32]](#footnote-32)**。

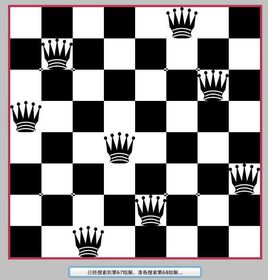


图 5‑2八皇后问题

1. **对角线法**

腾讯的peacock系统[17]采用的是更简单的对角线法，由于同一行或同一列的块不能被同时选择，因此选择对角线肯定是没有问题的。沿着一条条对角线进行采样计算，一旦一条对角线计算完成后，就并行计算下一条对角线，同一对角线内并行计算，不同对角线间串行计算。

### 计算和合并

由于上述算法过程中的方法是：组内并行，组间串行，所以小组并行执行的一次迭代完毕后，小组的nd、nw等统计量需要同步(sync)到下一个小组内。而组内各个块内sampling计算过程与**单机版gibbs sampling完全一致**，只是将统计量切换：nd、ndsum和nw切分成P份，比如第j份是nd\_j和nw\_j，由于在公式(3.16)中，nwsum(topic k被指定的单词的总数)无论如何都会产生冲突(依赖3)，所以整个算法的误差也就在这个统计量了。因此在第p台机器上的nwsum被修改为nwsum\_p后（并行计算完结后）都需要用global update方法merge back。



(.)

除了nwsum需要merge back之外，其余几个统计量，nd\_j和nw\_j都需要合并到下个小组的文档d和单词w。（这种合并过程可以用spark的broadcast）整个过程如下所示：

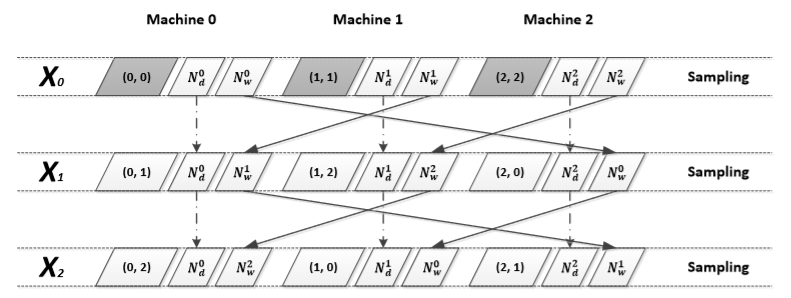


图 5‑3计算后合并，同一个单词的nw和同一个文档的nd统计量被合并

### 总结

**实验证明：**这种算法产生的混淆度(perlexity)与单机版Gibbs Sampling一致，因此误差非常小，而且分布式计算中的耗费内存也可以接受。但是实现复杂度稍高，需要比较好的spark编程功底。

(1)整个训练文章集(doc set)不仅以行切分(行指文章)，也以列切分。

(2)各个小块被切分尽可能数据量均等。

(3)统计量nw和nd因此也被切分到各个机器进程上。

(4)执行流程为：每轮迭代组内并行，组间串行。

(5)同一行或同一列的数据不能在一个小组内的不同块上同时并行执行。

(6)各个独立进程运行完毕后，误差仅在nwsum，通过global update合并nwsum。

# 附录

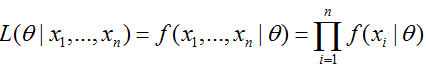
**极大似然估计(MLE)[13]**

最大似然估计提供了一种给定观察数据来评估模型参数的方法，即：“模型已定，参数未知”。简单而言，假设我们要统计全国人口的身高，首先假设这个身高服从正态分布，但是该分布的均值与方差未知。我们没有人力与物力去统计全国每个人的身高，但是可以通过采样，获取部分人的身高，然后通过最大似然估计来获取上述假设中的正态分布的均值与方差。

最大似然估计中采样需满足一个很重要的假设，就是所有的采样都是独立同分布(IID)的。下面我们具体描述一下最大似然估计：

首先，假设C:\Users\chen\AppData\Local\Temp\enhtmlclip\Image.png为独立同分布的采样，θ为模型参数,f为我们所使用的模型，遵循我们上述的独立同分布假设。参数为θ的模型f产生上述采样可表示为C:\Users\chen\AppData\Local\Temp\enhtmlclip\Image(1).png

回到上面的“模型已定，参数未知”的说法，此时，我们已知的为，未知为θ，故似然定义为:



在实际应用中常用的是两边取对数，得到公式如下：

C:\Users\chen\AppData\Local\Temp\enhtmlclip\Image(3).png

其中C:\Users\chen\AppData\Local\Temp\enhtmlclip\Image(4).png称为对数似然，而C:\Users\chen\AppData\Local\Temp\enhtmlclip\Image(5).png称为平均对数似然。而我们平时所称的最大似然为最大的对数平均似然，即：

C:\Users\chen\AppData\Local\Temp\enhtmlclip\Image(6).png

**例子1：**假如有一个罐子，里面有黑白两种颜色的球，数目多少不知，两种颜色的比例也不知。我 们想知道罐中白球和黑球的比例，但我们不能把罐中的球全部拿出来数。现在我们可以每次任意从已经摇匀的罐中拿一个球出来，记录球的颜色，然后把拿出来的球 再放回罐中。这个过程可以重复，我们可以用记录的球的颜色来估计罐中黑白球的比例。假如在前面的一百次重复记录中，有七十次是白球，请问罐中白球所占的比例最有可能是多少？很多人马上就有答案了：70%。而其后的理论支撑是什么呢？

我们假设罐中白球的比例是p，那么黑球的比例就是1-p。因为每抽一个球出来，在记录颜色之后，我们把抽出的球放回了罐中并摇匀，所以每次抽出来的球的颜 色服从同一独立分布。这里我们把一次抽出来球的颜色称为一次抽样。题目中在一百次抽样中，七十次是白球的概率是P(Data | M)，这里Data是所有的数据，M是所给出的模型，表示每次抽出来的球是白色的概率为p。如果第一抽样的结果记为x1，第二抽样的结果记为x2... 那么Data = (x1,x2,…,x100)。

P(Data | M)

= P(x1,x2,…,x100|M)

= P(x1|M)P(x2|M)…P(x100|M)

= p^70(1-p)^30.

那么p在取什么值的时候，P(Data |M)的值最大呢？将p^70(1-p)^30对p求导，并其等于零。

70p^69(1-p)^30-p^70\*30(1-p)^29=0。

解方程可以得到p=0.7。

在边界点p=0,1，P(Data|M)=0。所以当p=0.7时，P(Data|M)的值最大。这和我们常识中按抽样中的比例来计算的结果是一样的。

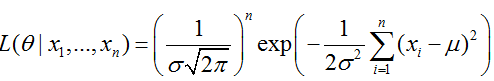
**例子2：**假如我们有一组连续变量的采样值（x1,x2,…,xn），我们知道这组数据服从正态分布，标准差已知。请问这个正态分布的期望值为多少时，产生这个已有数据的概率最大？

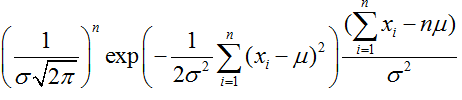
P(Data | M) = ?

根据公式

C:\Users\chen\AppData\Local\Temp\enhtmlclip\Image(7).png

可得:



对μ求导可得 ,则最大似然估计的结果为μ=(x1+x2+…+xn)/n

由上可知最大似然估计的一般求解过程：

（1）写出似然函数；

（2）对似然函数取对数，并整理；

（3）求导数 ；

（4）解似然方程

注意：最大似然估计只考虑某个模型能产生某个给定观察序列的概率。而未考虑该模型本身的概率。这点与贝叶斯估计区别。

# 参考文献

* 1. Philip Resnik and Eric Hardisty. Gibbs Sampling for the Uninitiated. CS-TR-4956 April 2010
  2. Gregor Heinrich. Parameter estimation for text analysis.
  3. 靳志辉(Rickjin). LDA数学八卦 2013.2.8
  4. 陈希孺. 概率论与数理统计. 中国科学技术大学出版社. 2009.2.1
  5. David M.Blei & Andrew Y.Ng. Latent Dirichlet Allocation
  6. 持之以恒. Reading Note : Parameter estimation for text analysis 暨LDA学习小结. <http://www.xperseverance.net/blogs/tag/lda/> . 2013.3.5
  7. Christopher M. Bishop. Pattern Recognition And Machine Learning. Springer. 2007.10.1. chapter 2 & chapter 11
  8. Mark Steyvers. Probabilistic Topic Models.
  9. David M.Blei & John D. Lafferty. Topic Models.
  10. Loulwah AlSumait and Daniel Barbara,James Gentle,Carlotta Domeniconi. Topic Signiﬁcance Ranking of LDA Generative Models.
  11. Zhuolin Qiu. Gibbs Collapsed Sampling for Latent Dirichlet Allocation on Spark. JMLR W&CP 36 : 17–28, 2014
  12. David Newman, Arthur Asuncion, Padhraic Smyth, Max Welling. Distributed Inference for Latent Dirichlet Allocation. {newman,asuncion,smyth,welling}@ics.uci.edu
  13. 最大似然估计 (Maximum likelihood estimation) <http://www.cnblogs.com/liliu/archive/2010/11/22/1883702.html>
  14. Xuan-Hieu Phan and Cam-Tu Nguyen. GibbsLDA++ Reference. <http://gibbslda.sourceforge.net/>
  15. Christophe Andrieu, Nando de Freitas, Arnaud Doucet, Michael I.Jordan. An Introduction to MCMC for Machine Learning. Kluwer Academic Publishers. 2001.9.10
  16. Allison J.B.Chaney and David M.Blei. Visualizing Topic Models. Association for the Advancement of Artiﬁcial Intelligence ([www.aaai.org](http://www.aaai.org)). 2012
  17. 赵学敏 王莉峰 王流斌 孙振龙 严浩 靳志辉 王益. Peacock：大规模主题模型及其在腾讯业务中的应用.

1. 陈希孺《概率论与数理统计》 p60, p67-例2.7 [↑](#footnote-ref-1)
2. http://en.wikipedia.org/wiki/Conjugate\_prior [↑](#footnote-ref-2)
3. 后验之意为执果索因，2.6节其实是贝叶斯估计，欲了解更多，参见陈希孺《概率论与数理统计》p155 [↑](#footnote-ref-3)
4. 采用贝叶斯法的出发点在于如果仅仅做了很少几次试验来估计参数（比如均值或概率p等），就会由于小样本数据造成估计不准。但利用之前已经拥有的经验（以前做过的试验数据）就可以令估计更为准确合理 [↑](#footnote-ref-4)
5. 图中被涂色的w表示可观测变量，未被涂色的表示未知的隐变量，N表示一篇文档中总共N个单词，M表示M篇文档 [↑](#footnote-ref-5)
6. 图片来自于《LDA数学八卦》 [↑](#footnote-ref-6)
7. 图片来自于《LDA数学八卦》 [↑](#footnote-ref-7)
8. 这里足见是主题分布的超参数 [↑](#footnote-ref-8)
9. 这里足见是词分布的超参数 [↑](#footnote-ref-9)
10. LDA推断方法有MCMC和变分EM，这个版本讲的MH算法和Gibbs Sampling就是MCMC算法的一种 [↑](#footnote-ref-10)
11. 马尔可夫链是个很简单的概念，百度/google一搜一大堆，我在这里用通俗说法 [↑](#footnote-ref-11)
12. 关于markov chain收敛的更严格详细的证明，参见：<http://users.aims.ac.za/~ioana/notes-ch4-2x1.pdf> 4.4 [↑](#footnote-ref-12)
13. Sample的意思也就是一个随机数生成器：按照某种概率分布来随机生成一系列数字 [↑](#footnote-ref-13)
14. <https://en.wikipedia.org/wiki/Metropolis%E2%80%93Hastings_algorithm>:第(2)个条件的Ergodicity可以进一步用两个条件来表示：(1) be aperiodic—the system does not return to the same state at fixed intervals; and (2) be positive recurrent—the expected number of steps for returning to the same state is finite. [↑](#footnote-ref-14)
15. 一个单词可同时被多个主题生成，φ就是从某个主题产生单词的概率，所以拆分成各主题的乘积 [↑](#footnote-ref-15)
16. 引自《parameter estimation for text analysis》 p21 [↑](#footnote-ref-16)
17. 这就是Collapsed Gibbs Sampling的Collapsed一词来历，这个integrating out出自《Gibbs Sampling for the Uninitiated》p15 这里可以积掉的深层原因是每个词上的topic采样都是独立的，而非《Uninitiated》一文里p18里的选定类标签L后一次性都将生成完一篇文章的所有word。（《Uninitiated中文版》的盘子图θ一对多：一个源头θ箭头指向一篇文章的所有word） [↑](#footnote-ref-17)
18. 上面的φ和θ“可以”被积分掉,原因在于为每个单词是被各topic独立生成，不互相影响φ和θ,因此可被积分掉. [↑](#footnote-ref-18)
19. 引自《parameter estimation for text analysis》p21 [↑](#footnote-ref-19)
20. further reading：

    普林斯顿大学材料：[Appendix D] Derivation of Gibbs sampling equations <https://lists.cs.princeton.edu/pipermail/topic-models/attachments/20110210/89b1646c/attachment-0001.pdf>

    HP实验室Note：Gibbs Sampling Derivation for LDA and TOT <http://home.in.tum.de/~xiaoh/pub/TOTGibbs.pdf> [↑](#footnote-ref-20)
21. 图中被涂色的W表示可观测变量，未被涂色的表示未知的隐变量，M表示M篇文档，K表示K个主题，其中一篇文档中总共N个单词 [↑](#footnote-ref-21)
22. 具体java版本代码请参考JGibbs LDA开源项目 [↑](#footnote-ref-22)
23. θ等价于theta，φ等价于phi [↑](#footnote-ref-23)
24. 第二维每篇文档不一样（变长），第二维=per doc word count [↑](#footnote-ref-24)
25. See：https://lists.cs.princeton.edu/pipermail/topic-models/2012-June/001957.html [↑](#footnote-ref-25)
26. HDP等较为复杂的模型可以自动确定这个参数，但是模型复杂，计算复杂 [↑](#footnote-ref-26)
27. 关于逻辑回归的详细分析我另著一本小册子《LR无痛优化指引》 [↑](#footnote-ref-27)
28. 该方法也可以使用google的word2vec等工具做 [↑](#footnote-ref-28)
29. 我称之为上帝向量或完美向量 [↑](#footnote-ref-29)
30. 注意到公式5.1正确地反映了全局topic assignment z，因为更新后的nw如实地反映了各个机器上topic assignment z，由于各机器p启动Gibbs Sampling时都是从全局nw开始修改: [↑](#footnote-ref-30)
31. 详见《Collapsed Gibbs Sampling for Latent Dirichlet Allocation on Spark》，该方法思想类似腾讯peacock [↑](#footnote-ref-31)
32. http://fdcwqmst.blog.163.com/blog/static/16406145520109294346965/ [↑](#footnote-ref-32)