

自然语言处理 (2) 主题模型 LDA (2 实现原理篇)



随风

大数据、人工智能

关注他

18 人赞同了该文章

1 前言

在我们阅读一篇文章的时候，需要明确文章的类别（体育类、新闻类）、内容以及中心思想。通常情况下，一篇文章可能包含多个主题，比如在介绍一座城市的时候，可能会从历史、经济、政治、教育、交通等多个方面做介绍。LDA 正是一种自动分析每篇文档，统计文档中的词语，根据统计的信息判断文档包含哪些主题以及各个主题所占比例的模型。

由于 LDA 的实现原理牵扯到一些数学知识，因此想彻底搞懂 LDA，请先参看 LDA 的数学基础篇：

随风：自然语言处理 (2) 主题模型
LDA (1 数学基础篇)

zhuanlan.zhihu.com



语言：python3

数据集：eventregistry.org/（实时新闻数据）

2 什么是主题模型？

主题模型 (Topic Model) 是通过学习一系列的文档，发现**抽象主题**的一种统计模型。从词频的角度来讲，如果一篇文章包含某个主题，那么一定存在一些特定的词语会频繁出现。通常情况下，一篇文章中包含多个主题，而且每个主题所占的比例各不相同。

比如：“美国对伊朗再度实施单边制裁，这将对伊朗的出口贸易造成严重影响”。这里可以归为政治主题，因为描述国家的词语频繁出现。也可以归为经济主题，因为出现了制裁、出口贸易等词。我们可以预估一下，政治主题的比例为 0.7，经济主题的比例为 0.3。

主题模型是对文本中**隐含主题**的一种建模方法，每个主题其实是词表上词语的概率分布。主题模型是一种**生成模型**，一篇文章中每个词都是通过“以一定概率选择某个主题，并从这个主题中以一定概率选择某个词语”这样一个过程得到的。

比如，我们写文章一般先选定一个主题，然后用和这个主题有关的词语进行组合形成文章。

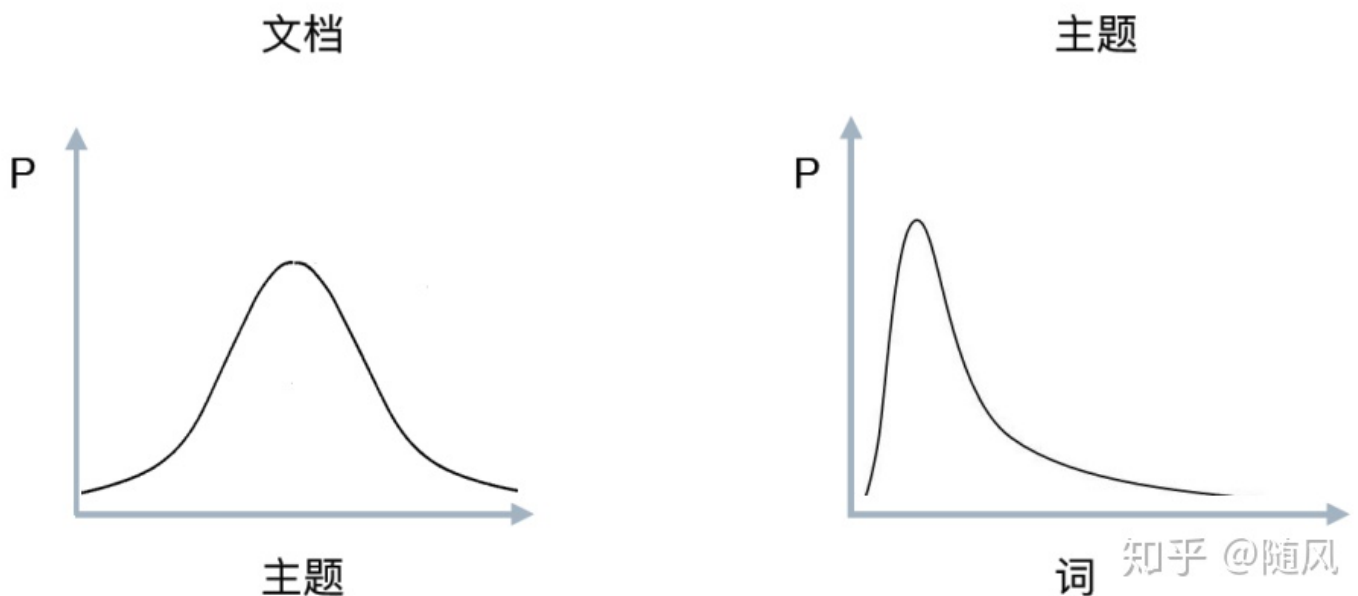
3 LDA 主题模型直观认识

隐含狄利克雷分布 (Latent Dirichlet Allocation, 简称 LDA), 是一种概率主题模型。LDA 可以将文档集中每篇文档的主题以概率分布的形式给出, 通过分析一批文档集, 抽取出它们的主题分布, 就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关系。此外, 一篇文档可以包含多个主题, 文档中每个词都由其中的一个主题生成。

LDA 是一种无监督学习方法, 在训练时不需要手工标注的训练集, 需要的仅仅是文档集以及指定主题的数量 k 即可。此外, LDA 的另一个优点是, 对于每一个主题均可找出一些词语来描述它。

一篇文章的每个词都是以一定概率选择某个主题, 并从这个主题中以一定概率选择某个词语而组成的。用公式表示为: $P(\text{word}|\text{doc}) = P(\text{word}|\text{topic}) * P(\text{topic}|\text{doc})$

从公式来看, $P(\text{word}|\text{doc})$ 可以通过文档中该词语出现的次数除以文档中词语总数计算出来。这里需要获得两个分布: 文档-主题分布、主题-词分布。



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

令 w 表示词语, d 表示文档, t 表示主题; 小写代表个体, 大写代表集合。D 中每篇文档 d 看作一个词语序列 (w_1, w_2, \dots, w_n) , w_i 表示第 i 个词语。D 中所有不同词语组成一个词汇集合 V 。假设有 k 个主题, V 中的词语数量为 m , LDA 以文档集合 D 作为输入, 得到两个输出:

对 D 中的每一篇文档 d , 对应到不同主题的概率 $\theta_d(P_{t1}, \dots, P_{tk})$ 。其中, P_{ti} 表示对应 T 中第 i 个主题的概率。计算方法是 $P_{ti} = n_{ti}/n$, 其中 n_{ti} 表示 d 中对应第 i 个主题的词语的数量, n 表示 d 中所有词语的总数。

对 T 中的每一个主题 t , 生成不同词语的概率 $\phi_t(P_{w1}, \dots, P_{wm})$ 。其中, P_{wi} 表示 t 生成 V 中第 i 个词语的概率。计算方法是 $P_{wi} = n_{wi}/n$, 其中 n_{wi} 表示对应到 t 的 V 中第 i 个词语

在语料库（所有文档）中的数量， n 表示语料库中所有对应到 t 的词语总数。

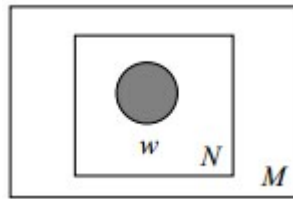
LDA 的核心公式为： $P(w|d) = P(w|t) * P(t|d)$ 。公式左侧的 $P(w|d)$ 可以用文档 d 中词语 w 的数量除以文档 d 中词语的总数得到，这是真实的 $P(w|d)$ 。公式右侧就是通过主题作为中间层，利用 θ_d 和 ϕ_t 中的元素计算 $P(w|d)$ ，这是通过两个分布拟合得到的 $P(w|d)$ 。LDA 学习的目的就是改变 θ_d 和 ϕ_t ，去拟合文档 d 中真实的 $P(w|d)$ 。

4 LDA 主题模型

4.1 Unigram model

对于文档 $w = (w_1, w_2, \dots, w_N)$ ，用 $p(w_i)$ 表示词 w_i 的先验概率（即一篇文档中该词的词频除以词的总数），则生成文档 w 的概率为：
$$p(w) = \prod_{i=1}^N p(w_i)。$$

其图模型为（图中被涂色的 w 表示可观测变量， N 表示一篇文档中总共 N 个词语， M 表示 M 篇文档）：

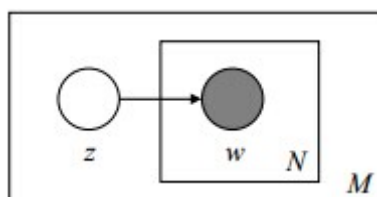


4.2 Mixture of unigrams model

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, z_2, \dots, z_k ，生成文档 w 的概率为：

$$p(w) = p(z_1) \prod_{i=1}^N p(w_i|z_1) + \dots + p(z_k) \prod_{i=1}^N p(w_i|z_k) = \sum_z p(z) \prod_{i=1}^N p(w_i|z)$$

其图模型为（图中被涂色的 w 表示可观测变量，未被涂色的 z 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）：



4.3 PLSA model

PLSA 模型是最接近 LDA 模型的，所以理解 PLSA 模型有助于我们理解 LDA 模型。

(1) PLSA 模型下生成文档过程

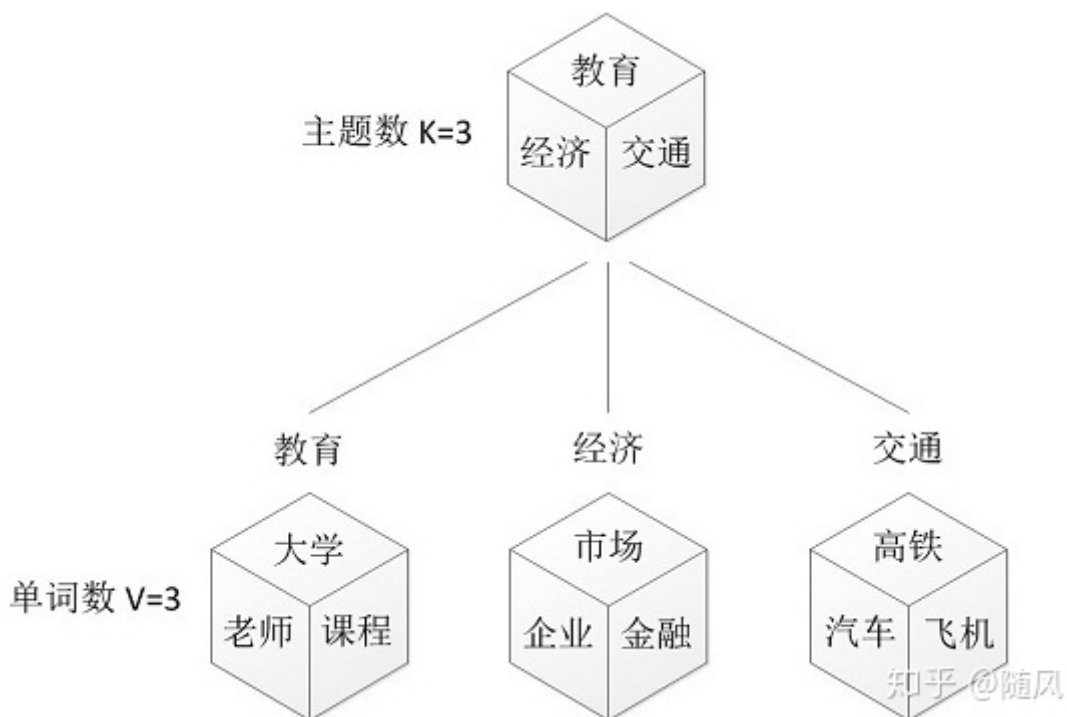
在上面的 Mixture of unigrams model 中，我们假定一篇文档只有一个主题生成，可实际中，一篇文章往往有多个主题，只是这多个主题各自在文档中出现的概率大小不一样。比如介绍一个国家的文档中，往往会分别从教育、经济、交通等多个主题进行介绍。那么在 PLSA 中，文档是怎样被生成的呢？

假设你要写 M 篇文档，由于一篇文档由各个不同的词组成，所以你需要确定每篇文档里每个位置上的词。

再假定你一共有 K 个可选的主题，有 V 个可选的词，咱们来玩一个扔骰子的游戏。

第一步：假设你每写一篇文档会制作一颗 K 面的“文档-主题”骰子（扔此骰子能得到 K 个主题中的任意一个），和 K 个 V 面的“主题-词项”骰子（每个骰子对应一个主题， K 个骰子对应之前的 K 个主题，且骰子的每一面对应要选择的词， V 个面对应着 V 个可选的词）。

比如可令 $K=3$ ，即制作 1 个含有 3 个主题的“文档-主题”骰子，这 3 个主题可以是：教育、经济、交通。然后令 $V=3$ ，制作 3 个有着 3 面的“主题-词项”骰子，其中，教育主题骰子的 3 个面上的词可以是：大学、老师、课程，经济主题骰子的 3 个面上的词可以是：市场、企业、金融，交通主题骰子的 3 个面上的词可以是：高铁、汽车、飞机。



第二步：每写一个词，先扔该“文档-主题”骰子选择主题，得到主题的结果后，使用和主题结果对应的那颗“主题-词项”骰子，扔该骰子选择要写的词。

先扔“文档-主题”的骰子，假设（以一定的概率）得到的主题是：教育，所以下一步便是扔教育主题筛子，（以一定的概率）得到教育主题筛子对应的某个词：大学。

上面这个投骰子产生词的过程简化一下便是：“**先以一定的概率选取主题，再以一定的概率选取词**”。事实上，一开始可供选择的主题有 3 个：教育、经济、交通，那为何偏偏选取教育这个主题呢？其实是随机选取的，只是这个随机遵循一定的概率分布。比如可能选取教育主题的概率是 0.5，选取经济主题的概率是 0.3，选取交通主题的概率是 0.2，那么这 3 个主题的概率分布便是 {教育：0.5，经济：0.3，交通：0.2}，**我们把各个主题 z 在文档 d 中出现的概率分布称之为主题分布，且是一个多项分布（因为主题分布是 n 次掷筛子产生的）。**

同样的，从主题分布中随机抽取出教育主题后，依然面对着 3 个词：大学、老师、课程，这 3 个词都可能被选中，但它们被选中的概率也是不一样的。比如大学这个词被选中的概率是 0.5，老师这个词被选中的概率是 0.3，课程被选中的概率是 0.2，那么这 3 个词的概率分布便是 {大学：0.5，老师：0.3，课程：0.2}，**我们把各个词语 w 在主题 z 下出现的概率分布称之为词分布，这个词分布也是一个多项分布（因为词分布是 n 次掷筛子产生的）。**

所以，选主题和选词都是两个随机的过程，先从主题分布 {教育：0.5，经济：0.3，交通：0.2} 中取出主题：教育，然后从该教育主题对应的词分布 {大学：0.5，老师：0.3，课程：0.2} 中取出词：大学。

第三步：最后，你不停的重复扔“文档-主题”骰子和“主题-词项”骰子，重复 N 次（产生 N 个词），完成一篇文档，重复这产生一篇文档的方法 M 次，则完成 M 篇文档。

上述过程抽象出来即是 PLSA 的文档生成模型。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。定义：

$p(d_j)$ 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 w_j 在给定文档 d_i 中出现的概率（文档中该词出现次数/总次数）。

$p(z_k|d_i)$ 表示具体某个主题 z_k 在给定文档 d_i 下出现的概率。

$p(w_j|z_k)$ 表示具体某个词 w_j 在给定主题 z_k 下出现的概率，与主题关系越密切的词，其条件概率 $p(w_j|z_k)$ 越大。

利用上述定义，我们便可以按照如下的步骤得到“文档-词项”的生成模型：

按照概率 $p(d_i)$ 选择一篇文档 d_i

选定文档 d_i 后，从主题分布中按照概率 $p(z_k|d_i)$ 选择一个隐含的主题类别 z_k

选定主题 z_k 后，从词分布中按照概率 $p(w_j|z_k)$ 选择一个词 w_j

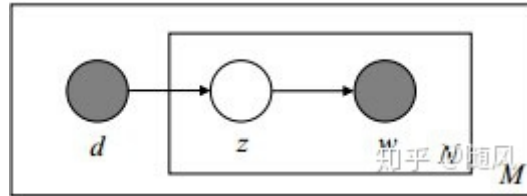
(2) 根据文档反推其主题分布

反过来，既然文档已经产生，那么如何根据已经产生好的文档反推其主题呢？这个利用看到的文档推断其隐藏的主题（分布）的过程（其实也就是产生文档的逆过程），便是主题建模的目的：自动

地发现文档集中的主题（分布）。

换言之，人类根据文档生成模型写成了各类文章，然后丢给了计算机，相当于计算机看到的是一篇篇已经写好的文章。现在计算机需要根据一篇篇文章中看到的一系列词归纳出每篇文章的主题，进而得出每篇文章中各个主题的出现概率：主题分布。即文档 d 和单词 w 是可被观察到的，但主题 z 却是隐藏的。

如下图所示（图中被涂色的 d 、 w 表示可观测变量，未被涂色的 z 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）：



上图中，文档 d 和词 w 是我们得到的样本（样本随机，参数虽未知但固定，所以 PLSA 属于频率派思想。区别于下文要介绍的 LDA 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 属于贝叶斯派思想），可观测得到，所以对于任意一篇文档，其 $P(w_j|d_i)$ 是已知的。

频率派与贝叶斯派各自不同的思考方式：

频率派把需要推断的参数 θ 看做是固定的未知常数，即概率虽然是未知的，但最起码是确定的一个值，同时，样本 X 是随机的，所以频率派重点研究样本空间，大部分的概率计算都是针对样本 X 的分布；

而贝叶斯派的观点则截然相反，他们认为待估计的参数是随机变量，服从一定的分布，而样本 X 是固定的，由于样本是固定的，所以他们重点研究的是参数的分布。

从而可以根据大量已知的文档-词项信息 $P(w_j|d_i)$ ，训练出文档-主题 $p(z_k|d_i)$ 和主题-词项

$p(w_j|z_k)$ ，公式：
$$p(w_j|d_i) = \sum_{k=1}^K p(w_j|z_k)p(z_k|d_i)$$
。故得到文档中每个词的生成概率

为：
$$p(d_i, w_j) = p(d_i)p(w_j|d_i) = p(d_i) \sum_{k=1}^K p(w_j|z_k)p(z_k|d_i)$$
。由于 $p(d_i)$ 可事先

计算求出，而 $P(w_j|z_k), P(z_k|d_i)$ 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，通俗地讲，就是要最大化这个 θ 。（优化的方法是采用 EM 算法，主要思想是极大似然估计）

4.4 LDA model

事实上，理解了 PLSA 模型，也就差不多快理解了 LDA 模型了，因为 LDA 就是在 PLSA 的基础上加了层贝叶斯框架，即 LDA 就是 PLSA 的贝叶斯版本。

(1) LDA 模型生成一篇文档的方式:

按照先验概率 $p(d_i)$ 选择一篇文档 d_i

从 Dirichlet 分布 α 中取样生成文档 d_i 的主题分布 θ_i (θ_i 为多项分布), 换言之, 主题分布 θ_i 由超参数 α 的 Dirichlet 分布生成

从主题的多项式分布 θ_i 中取样生成文档 d_i 的第 j 个词的主题 $z_{i,j}$

从 Dirichlet 分布 β 中取样生成主题 $z_{i,j}$ 对应的词语分布 $\phi_{z_{i,j}}$ ($\phi_{z_{i,j}}$ 为多项分布), 换言之, 词语分布 $\phi_{z_{i,j}}$ 由参数为 β 的 Dirichlet 分布生成

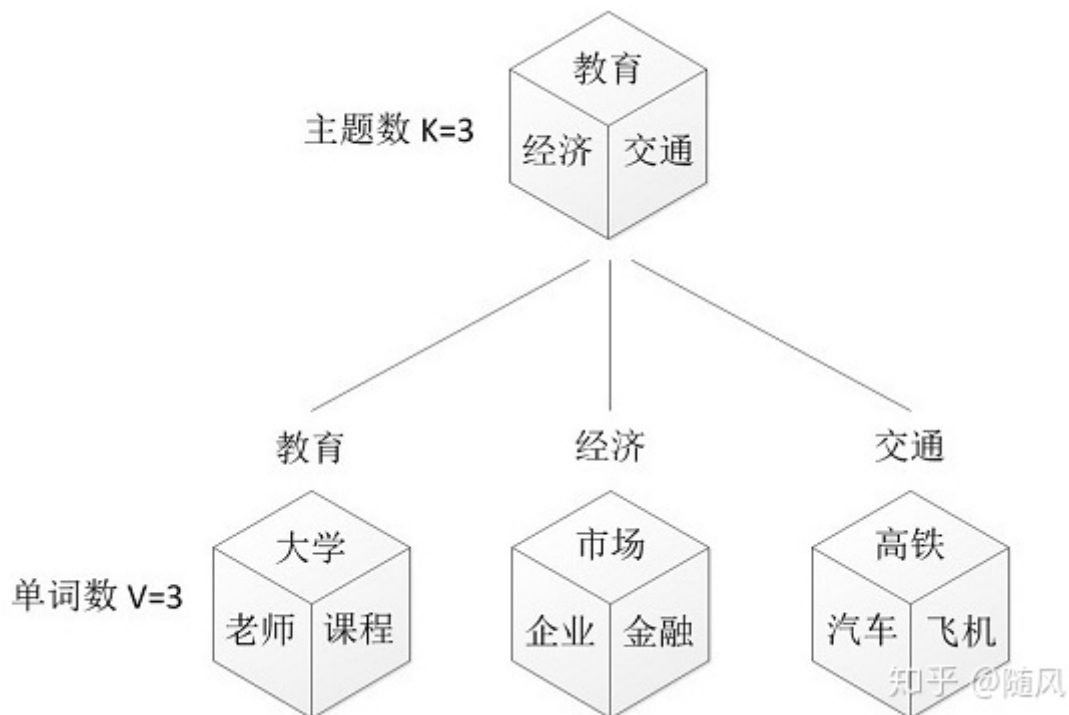
从词语的多项式分布 $\phi_{z_{i,j}}$ 中采样最终生成词语 $w_{i,j}$

从上面的过程可以看出, 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

(2) PLSA 与 LDA 的区别

如前所述, 在 PLSA 中, 选主题和选词都是两个随机的过程。主题分布和词分布是唯一确定的, 能明确的指出主题分布可能就是{教育: 0.5, 经济: 0.3, 交通: 0.2}, 词分布可能就是{大学: 0.5, 老师: 0.3, 课程: 0.2}。

但在 LDA 中, 主题分布和词分布不再唯一确定不变, 即无法确切给出。例如主题分布可能是{教育: 0.5, 经济: 0.3, 交通: 0.2}, 也可能是{教育: 0.6, 经济: 0.2, 交通: 0.2}, 到底是哪个我们不再确定, 因为它是随机的。但再怎么变化, 也依然服从一定的分布, 即主题分布跟词分布由 Dirichlet 先验随机确定。



前面我们提到的是, 有很多个主题或词, 各个主题或词被抽中的概率不一样, 所以抽取主题或词是随机抽取了。

但是现在主题分布和词分布也不确定了，也就是说，在我们抽取主题的时候，之前是从一个固定的主题分布中，现在是这个主题分布也是不确定的，可能是{教育：0.5，经济：0.3，交通：0.2}，也可能是{教育：0.6，经济：0.2，交通：0.2}。那这个主题分布到底是什么取值？需要根据一个先验 Dirichlet 分布来随机取，所以可以从无穷多个主题分布中按照 Dirichlet 分布随机抽取出某个主题分布来。词分布也是根据 Dirichlet 分布随机抽取出来的，可以从无穷多个词分布中按照 dirichlet 分布随机抽取出某个词分布来。

换言之，LDA 在 PLSA 的基础上给主题分布和词分布加了两个先验分布的参数（贝叶斯化）：一个主题分布的先验分布 Dirichlet 分布 α ，和一个词语分布的先验分布 Dirichlet 分布 β 。

所以，PLSA 跟 LDA 的本质区别就在于它们去估计未知参数所采用的思想不同，前者用的是频率派思想，后者用的是贝叶斯派思想。因为 LDA 是 PLSA 的贝叶斯版本，所以主题分布跟词分布本身由先验知识随机给定。

好比，我去一朋友家：

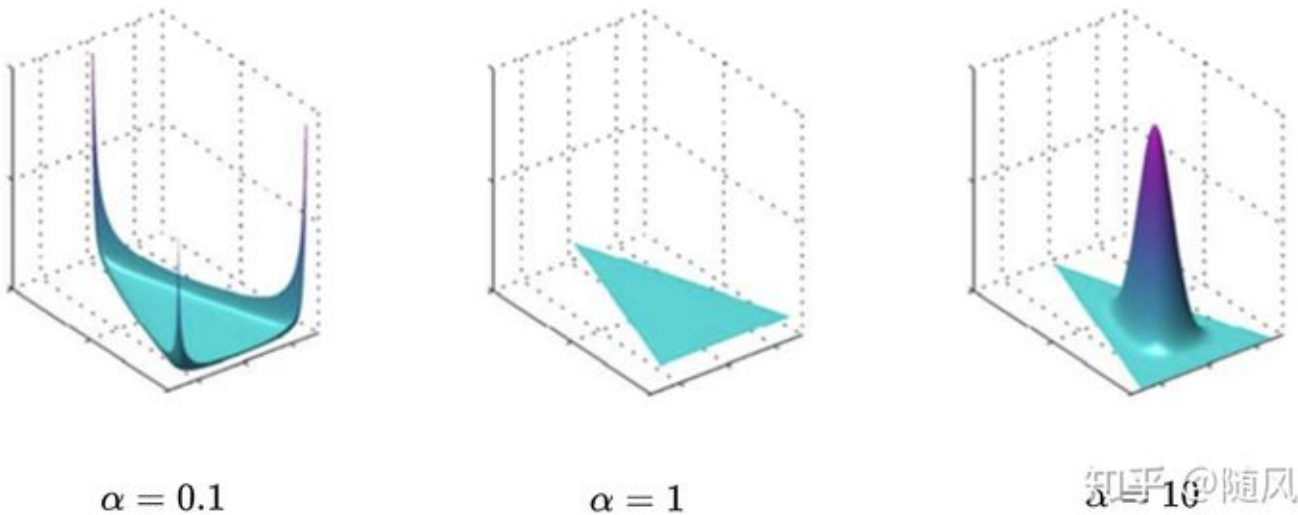
按照频率派的思想，我估计他在家的概率是 1/2，不在家的概率也是 1/2，是个定值。

按照贝叶斯派的思想，他在家不在家的概率不再认为是个定值 1/2，而是随机变量。比如按照我们的经验（比如当天周末），猜测他在家的概率是 0.6，但这个 0.6 不是说就是完全确定的，也有可能是 0.7。如此，贝叶斯派没法确切给出参数的确定值（0.3,0.4,0.6,0.7,0.8,0.9都有可能），但至少明白在哪个范围或哪些取值（0.6,0.7,0.8,0.9）更有可能，哪个范围或哪些取值（0.3,0.4）不太可能。进一步，贝叶斯估计中，参数的多个估计值服从一定的先验分布，而后根据实践获得的数据（例如周末不断跑他家），不断修正之前的参数估计，从先验分布慢慢过渡到后验分布。

(3) LDA 生成文档过程的进一步理解

上面说，LDA 中，主题分布是由 Dirichlet 先验给定的，不是根据文档产生的。所以，LDA 生成文档的过程中，先从 Dirichlet 先验中“随机”抽取出主题分布，然后从主题分布中“随机”抽取出主题，最后从确定后的主题对应的词分布中“随机”抽取出词。那么，Dirichlet 先验到底是如何“随机”抽取主题分布的呢？

事实上，从 Dirichlet 分布中随机抽取主题分布，这个过程不是完全随机的。假设我们的主题分布是一个二维分布 (p_1, p_2) ，如下图所示，三维空间中的底面表示主题分布的一个点 (p_1, p_2) ，高度代表某个主题分布（某个点）被 Dirichlet 分布选中的概率，且选不同的 Dirichlet 分布会偏向不同的主题分布。显然从图中可以看出，当选取不同的 Dirichlet 分布时，各个主题分布被取到的概率也各不相同。

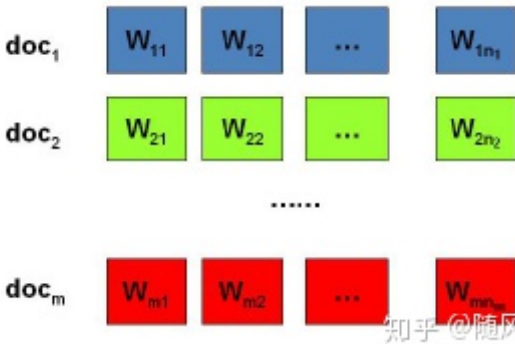


此外，得到主题分布以后，“随机”选主题也是根据主题分布来“随机”选取，这里的随机不是完全随机的意思，而是根据各个主题出现的概率值大小来抽取。比如当 Dirichlet 先验为文档 d 生成的主题分布是 $(0.3, 0.7)$ ，那么主题 z_2 在文档 d 中出现的概率便是 0.7。所以，从主题分布中抽取主题，这个过程也不是完全随机的，而是按照各个主题出现的概率值大小进行抽取。

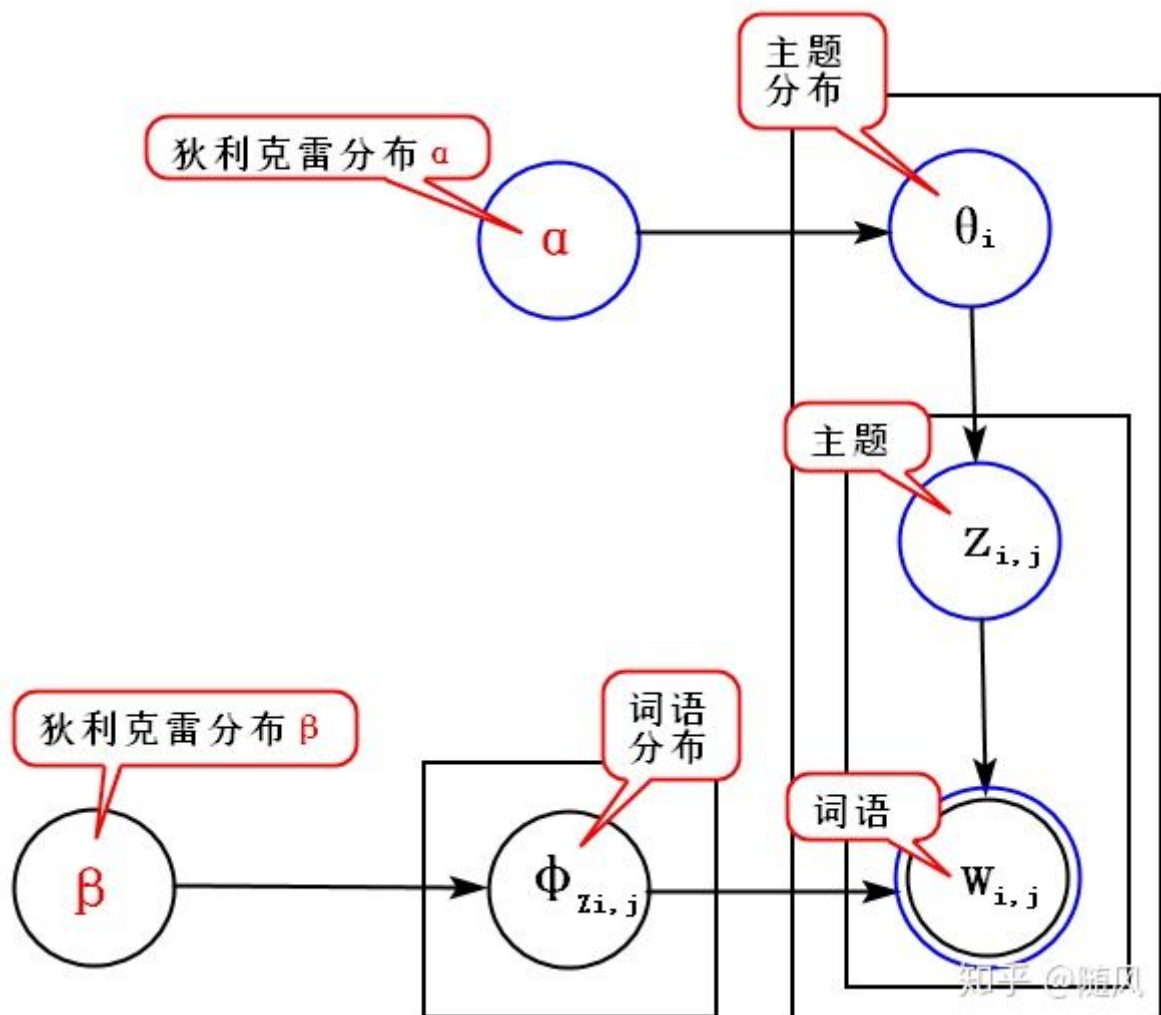
5 LDA 实现原理

5.1 建立模型

我们的问题是这样的，有 M 篇文档，对应第 m 篇文档中有 n_m 个词。即输入为如下图：



我们的目标是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 ，这样所有的分布就都基于 K 个主题展开。具体 LDA 模型如下图：



LDA 假设文档-主题的先验分布是 Dirichlet 分布，即对于任一篇文档 m ，其主题分布 θ_m 为：

从 $Dirichlet(\bar{\alpha})$ 中随机抽取一次，产生一个多项分布 θ_m 。

其中 $\bar{\alpha}$ 是分布的超参数，是一个 K 维向量， K 表示主题的数量。

LDA 假设主题-词的先验分布是 Dirichlet 分布，即对于任一主题 k ，其词分布 ϕ_k 为：

从 $Dirichlet(\bar{\beta})$ 中随机抽取一次，产生一个多项分布 ϕ_k 。

其中 $\bar{\beta}$ 是分布的超参数，是一个 V 维向量， V 表示词汇表里所有词的个数（注意词汇表里的词是文档集合 M 中所有不重复的词语组成的集合）。

对于文档集 M 中任意一篇文档 m 中的第 i 个词 w_{mi} 属于某个主题 z_k 的概率分布即为： θ_m ，而对于该主题，出现词 w_{mi} 的概率分布为： ϕ_k 。

理解 LDA 主题模型的主要任务就是理解上面这个模型。这个模型里，我们有 M 个关于文档主题的 Dirichlet 先验分布，而对应的文档集就有 M 个文档主题的多项分布，这样 $\alpha \rightarrow \theta_m \rightarrow z_m \rightarrow w_m$ 就组成了 Dirichlet-multi 共轭，可以使用前面提到的贝叶斯推断的方法得到基于 Dirichlet 分布的关于文档主题的后验分布。

如果在第 m 篇文档中, 属于第 k 个主题的词个数为: $n_m^{(k)}$, 则对应的多项分布的计数可以表示为: $\bar{n}_m = (n_m^{(1)}, n_m^{(2)}, \dots, n_m^{(K)})$ 。

利用 Dirichlet-multi 共轭, 得到 θ_m 的后验分布为: $Dirichlet(\theta_m | \bar{\alpha} + \bar{n}_m)$ 。

同理, 对于主题-词分布, 我们有 K 个关于主题词的 Dirichlet 分布, 而对应的文档集就有 K 个主题词的多项分布, 这样 $\beta \rightarrow \phi_k \rightarrow \bar{w}_k$ 就组成了 Dirichlet-multi 共轭, 可以使用前面提到的贝叶斯推断的方法得到基于 Dirichlet 分布的关于主题词的后验分布。

如果在第 k 个主题中, 第 v 个词的个数为: $n_k^{(v)}$, 则对应的多项分布的计数可以表示为: $\bar{n}_k = (n_k^{(1)}, n_k^{(2)}, \dots, n_k^{(V)})$ 。

利用 Dirichlet-multi 共轭, 得到 ϕ_k 的后验分布为: $Dirichlet(\phi_k | \bar{\beta} + \bar{n}_k)$ 。

由于主题产生词不依赖于具体的某一篇文档, 因此文档-主题分布和主题-词分布是独立的。理解了上面这 $M+K$ 组 Dirichlet-multi 共轭, 就理解了 LDA 的基本原理了。

现在的问题是, 基于这个 LDA 模型如何求解我们想要的每一篇文档的主题分布和每一个主题中词的分布呢?

一般有两种方法, 第一种是基于 Gibbs 采样算法求解, 第二种是基于变分推断 EM 算法求解 (原论文采用变分推断 EM 算法)。下面介绍 Gibbs 采样方法来求解。

5.2 模型求解

文档集 M 联合起来形成的词向量为 \bar{w} (注意向量中有重复词, 而且向量是已知的), 文档集 M 联合起来形成的主题分布为 \bar{z} (注意这是所有文档联合形成的主题分布, 不是某一篇文档的主题分布, 是未知的)。假如我们可以先求出 \bar{w}, \bar{z} 的联合概率分布 $p(\bar{w}, \bar{z})$, 就可以求出某个词 w_i 对应主题 z_k 的条件概率分布 $p(z_i = z_k | \bar{w}, \bar{z}_{-i})$, 其中 z_i 表示第 i 个词 w_i 对应的主题, \bar{z}_{-i} 表示去掉下标为 i 的词后的主题分布。

有了条件概率分布 $p(z_i = z_k | \bar{w}, \bar{z}_{-i})$, 我们就可以进行 Gibbs 采样了, 最终在 Gibbs 采样收敛后得到第 i 个词的主题。

如果我们通过采样得到了所有词的主题, 那么通过统计所有词的主题计数, 就可以得到各个主题的词分布。接着统计各个文档对应词的主题计数, 就可以得到各个文档的主题分布。

以上就是 Gibbs 采样算法求解 LDA 的思路。

因为 $\bar{\alpha}$ 产生主题分布 θ_m , 主题分布 θ_m 产生主题 z_k , $\bar{\beta}$ 产生主题 z_k 的词分布 ϕ_k , 词分布 ϕ_k 确定具体的词 w_i 。所以联合概率分布可表示为:

$$p(\bar{w}, \bar{z}) \propto p(\bar{w}, \bar{z} | \bar{\alpha}, \bar{\beta}) = p(\bar{w} | \bar{z}, \bar{\beta}) p(\bar{z} | \bar{\alpha}) .$$

第一项 $p(\bar{z} | \bar{\alpha})$ 表示根据主题分布的先验分布参数 $\bar{\alpha}$ 采样主题的过程

第二项 $p(\bar{w} | \bar{z}, \bar{\beta})$ 表示根据确定的主题 \bar{z} 和词分布的先验分布参数 $\bar{\beta}$ 采样词的过程

这两项因子是需要计算的未知参数。由于这两个过程是独立的，所以下面可以分别处理。

根据主题分布的先验分布参数 $\bar{\alpha}$ 采样得到第 m 篇文档的主题分布 θ_m 的概率为：

$$Dirichlet(\theta_m | \bar{\alpha}) = \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \theta_{m,z_k}^{\alpha_k-1} = \frac{1}{\Delta(\bar{\alpha})} \prod_{k=1}^K \theta_{m,z_k}^{\alpha_k-1} , \text{ 其中}$$

$$\frac{1}{\Delta \bar{\alpha}} = \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} , \text{ 作为归一化因子。}$$

根据第 m 篇文档的主题分布 θ_m 产生主题向量 \bar{z}_m 的条件概率为：

$$p(\bar{z}_m | \theta_m) = \prod_{k=1}^K p(z_{mi} = z_k) = \prod_{k=1}^K \theta_{m,z_k}^{n_m^{(z_k)}}$$

其中： z_{mi} 表示第 m 篇文档中第 i 个词对应的主题， θ_{m,z_k} 表示第 m 篇文档中主题 z_k 的概率， $n_m^{(z_k)}$ 表示主题 z_k 在第 m 篇文档中出现的次数。则条件概率 $p(\bar{z}_m | \bar{\alpha})$ 表现为对主题分布 θ_m 的积分：

$$\begin{aligned} p(\bar{z}_m | \bar{\alpha}) &= \int p(\bar{z}_m | \theta_m) p(\theta_m | \bar{\alpha}) d\theta_m = \int \prod_{k=1}^K \theta_{m,z_k}^{n_m^{(z_k)}} \frac{1}{\Delta(\bar{\alpha})} \prod_{k=1}^K \theta_{m,z_k}^{\alpha_k-1} d\theta_m \\ &= \frac{1}{\Delta(\bar{\alpha})} \int \prod_{k=1}^K \theta_{m,z_k}^{n_m^{(z_k)} + \alpha_k - 1} d\theta_m = \frac{\Delta(\bar{n}_m + \bar{\alpha})}{\Delta(\bar{\alpha})} \end{aligned}$$

有了第 m 篇文档的主题条件分布 $p(\bar{z}_m | \bar{\alpha})$ ，就可以得到文档集合 M 的主题条件分布：

$$p(\bar{z} | \bar{\alpha}) = \prod_{m=1}^M p(\bar{z}_m | \bar{\alpha}) = \prod_{m=1}^M \frac{\Delta(\bar{n}_m + \bar{\alpha})}{\Delta(\bar{\alpha})}$$

$$\text{同理，可以得到： } p(\bar{w} | \bar{z}, \bar{\beta}) = \prod_{k=1}^K p(\bar{w}_k | \bar{z}, \bar{\beta}) = \prod_{k=1}^K \frac{\Delta(\bar{n}_k + \bar{\beta})}{\Delta(\bar{\beta})}$$

最终我们得到主题和词的联合分布：

$$p(\bar{w}, \bar{z} | \bar{\alpha}, \bar{\beta}) = p(\bar{z} | \bar{\alpha}) p(\bar{w} | \bar{z}, \bar{\beta}) = \prod_{m=1}^M \frac{\Delta(\bar{n}_m + \bar{\alpha})}{\Delta(\bar{\alpha})} \prod_{k=1}^K \frac{\Delta(\bar{n}_k + \bar{\beta})}{\Delta(\bar{\beta})}$$

有了联合分布，现在我们就可以求 Gibbs 采样需要的条件概率分布 $p(z_i = z_k | \bar{w}, \bar{z}_{-i})$ 了。需要注意的是这里的 i 是一个二维下标，对应第 m 篇文档的第 k 个主题 z_k 中的第 j 个词。

对于下标 i ，由于它对应的词 w_i 是可以观察到的，因此有：

$$p(z_i = z_k | \bar{w}, \bar{z}_{-i}) \propto p(z_i = z_k, w_i = z_{kj} | \bar{w}_{-i}, \bar{z}_{-i})$$

其中 z_{kj} 表示主题 z_k 中的第 j 个词。对于 $z_i = z_k, w_i = z_{kj}$ ，它只涉及到第 m 篇文档和第 k 个主题两个 Dirichlet-multi 共轭，即： $\alpha \rightarrow \theta_m \rightarrow \bar{z}_m, \beta \rightarrow \phi_k \rightarrow \bar{w}_k$

其余的 $M+K-2$ 个 Dirichlet-multi 共轭和它们这两个共轭是独立的。如果我们在语料库中去掉 z_i, w_i ，并不会改变之前的 $M+K$ 个 Dirichlet-multi 共轭结构，只是向量的某些位置的计数会减少，因此对于 θ_m, ϕ_k 对应的后验分布为：

$$p(\theta_m | \bar{w}_{-i}, \bar{z}_{-i}) = \text{Dirichlet}(\theta_m | n_{m-} + \bar{\alpha})$$

$$p(\phi_k | \bar{w}_{-i}, \bar{z}_{-i}) = \text{Dirichlet}(\phi_k | n_{k-} + \bar{\beta})$$

现在开始计算 Gibbs 采样需要的条件概率：

$$\begin{aligned} p(z_i = z_k | \bar{w}, \bar{z}_{-i}) &\propto p(z_i = z_k, w_i = z_{kj} | \bar{w}_{-i}, \bar{z}_{-i}) \\ &= \int p(z_i = z_k, w_i = z_{kj}, \theta_m, \phi_k | \bar{w}_{-i}, \bar{z}_{-i}) d\theta_m d\phi_k \\ &= \int p(z_i = z_k, \theta_m | \bar{w}_{-i}, \bar{z}_{-i}) p(w_i = z_{kj}, \phi_k | \bar{w}_{-i}, \bar{z}_{-i}) d\theta_m d\phi_k \\ &= \int p(z_i = z_k | \theta_m) p(\theta_m | \bar{w}_{-i}, \bar{z}_{-i}) p(w_i = z_{kj} | \phi_k) p(\phi_k | \bar{w}_{-i}, \bar{z}_{-i}) d\theta_m d\phi_k \\ &= \int p(z_i = z_k | \theta_m) \text{Dirichlet}(\theta_m | n_{m-} + \bar{\alpha}) d\theta_m * \int p(w_i = z_{kj} | \phi_k) \text{Dirichlet}(\phi_k | n_{k-} + \bar{\beta}) d\phi_k \\ &= \int \theta_{m,z_k} \text{Dirichlet}(\theta_m | n_{m-} + \bar{\alpha}) d\theta_m * \int \phi_{k,j} \text{Dirichlet}(\phi_k | n_{k-} + \bar{\beta}) d\phi_k \\ &= E_{\text{Dirichlet}(\theta_m)}(\theta_{m,z_k}) E_{\text{Dirichlet}(\phi_k)}(\phi_{k,j}) \end{aligned}$$

由 Dirichlet 分布的期望公式：

$$E_{\text{Dirichlet}(\theta_m)}(\theta_{m,z_k}) = \frac{n_{m-}^k + \alpha_k}{\sum_{k=1}^K (n_{m-}^k + \alpha_k)}, E_{\text{Dirichlet}(\phi_k)}(\phi_{k,j}) = \frac{n_{k-}^j + \beta_j}{\sum_{v=1}^V (n_{k-}^v + \beta_v)}$$

$$\text{得: } p(z_i = z_k | \bar{w}, z_{-i}^-) = \frac{n_{m \rightarrow i}^k + \alpha_k}{\sum_{k=1}^K (n_{m \rightarrow i}^k + \alpha_k)} \frac{n_{k \rightarrow i}^j + \beta_j}{\sum_{v=1}^V (n_{k \rightarrow i}^v + \beta_v)}$$

有了这个公式，我们就可以用 Gibbs 采样去采样所有词的主题，当 Gibbs 采样收敛后，即得到所有词的采样主题。利用所有采样得到的词和主题的对应关系，我们就可以得到每个文档-主题分布 θ_m ，和每个主题-词分布 ϕ_k 。

5.3 Gibbs 采样流程

现在我们总结下 LDA Gibbs 采样算法流程。首先是训练流程：

选择合适的主题数 K ，选择合适的超参数向量 $\bar{\alpha}, \bar{\beta}$

对应语料库中每一篇文档的每一个词，随机的赋予一个主题编号 z

重新扫描语料库，对于每一个词，利用 Gibbs 采样公式更新它的 topic 编号，并更新语料库中该词的编号。

重复第 3 步的基于坐标轴轮换的 Gibbs 采样，直到 Gibbs 采样收敛。

统计语料库中的各个文档各个词的主题，得到文档主题分布 θ_m ，统计语料库中各个主题词的分布，得到 LDA 的主题与词的分布 ϕ_k

下面我们再来看看当新文档出现时，如何统计该文档的主题。此时我们的模型已定，也就是 LDA 的各个主题的词分布 ϕ_k 已经确定，我们需要得到的是该文档的主题分布。因此在 Gibbs 采样时，我们的 $E_{\text{Dirichlet}(\phi_k)}(\phi_{k,j})$ 已经固定，只需要对前半部分 $E_{\text{Dirichlet}(\theta_m)}(\theta_{m,z_k})$ 进行采样计算即可。

现在我们总结下 LDA Gibbs 采样算法的预测流程：

对于当前文档的每一个词，随机的赋予一个主题编号 z

重新扫描当前文档，对于每一个词，利用 Gibbs 采样公式更新它的 topic 编号。

重复第 2 步的基于坐标轴轮换的 Gibbs 采样，直到 Gibbs 采样收敛。

统计文档中各个词的主题，得到该文档主题分布。

6 LDA 实战

这一次我们采用最新的实时新闻数据，地址：eventregistry.org/。这是一个应用 AI 技术实现实时新闻推送的网站，并且提供了 python 的 API，可以实时下载新闻数据。使用方法：

```
from eventregistry import *

filepath = 'D:/data.txt'
```

```


er = EventRegistry(apiKey='*****') # 这里 key 需要注册后获取, 不能采用个人邮箱, 用自己的公
articles = QueryArticlesIter(keywords=QueryItems.OR(['china', 'travel', 'cate', 'movie
                                     lang='eng', dataType=['news']]) # keywords 是从语料库中选出
                                     # lang 选择英语, 这里支持 35
                                     # dataType 选择了新闻, 也可以

# obtain at most 1000000 newest articles or blog posts
# 通过打印可以查看实际获取的新闻数量。观察发现, 其实它是 100 条请求一次服务端
# 每个 key 有请求数量限制, 实际上这里最后获取了 10 万条新闻
num = 1
with open(filepath, 'w', encoding='utf-8') as f:
    for art in articles.execQuery(er, sortBy='date', maxItems=1000000):
        print(num)
        num = num + 1
        title = art['title']
        body = art['body'].replace('\n', ' ')
        f.write(title + '\n')
        f.write(body + '\n')

f.close()

```

最后查看一下获取的语料库的大小:

 data.txt	2019/8/14 6:35	文本文档	447,305 KB
--	----------------	------	------------

通过 lda 的原理分析, 语料库的内容应该是越全面越好。即假如我的所有语料只包含三个主题: 经济、文化、科技, 我们希望围绕这三个主题的语料尽可能丰富。而实际上我们在做这样的语料分析时, 是不知道具体包含了哪些主题, 所以我们希望语料库尽可能的丰富, 涵盖所有主题的语料尽可能的多一点。

训练 lda 需要三类参数, 即主题数量 K (需要预先指定), 文档-主题先验分布

$\bar{\alpha} = (\bar{\alpha}_1, \bar{\alpha}_2, \dots, \bar{\alpha}_m)$ (m 篇文档), 主题-词先验分布 $\bar{\beta} = (\bar{\beta}_1, \bar{\beta}_2, \dots, \bar{\beta}_K)$ (K 个主题)。先验参数往往难以确定, 只能给定一个合适的范围, 通过样本去修正, 以便得到后验分布。

```

from gensim import corpora, models, similarities
import gensim
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

```

#关于数据预处理, 仁者见仁智者见智, 我这里是通过抽取 500 篇语料观察得到的结果

#使用正则的时候尽量不要采用模糊匹配, 比如 `.*`, 观察发现这样做很可能满足了这篇文档的条件, 但是另一篇文

```

def process_articles(text):
    text = text.lower()
    text = text.replace('【', '')
    text = text.replace('】', '')
    text = text.replace('《', '')
    text = text.replace('》', '')
    text = re.sub('[\u4e00-\u9fa5]', ' ', text) #英文文档竟然出现了汉字,也是醉了
    text = re.sub(r'/', ' ', text)
    text = re.sub(r'//', ' ', text)
    text = re.sub(r'\\', ' ', text)
    text = re.sub(r'\\\\', ' ', text)
    text = re.sub(r' - ', ' ', text) #把 "-" 的两个单词分开 (比如: july-edu ==> july edu)
    text = re.sub(r'--', ' ', text)
    text = re.sub(r'—', ' ', text)
    text = re.sub(r'\d+', '', text) #去掉数字
    words = word_tokenize(text)
    english_stopwords = stopwords.words('english')
    addition_stopwords = ['\re', '\s', '\t', '\m', '\ll', '\ve', '\d', 'n\t']
    english_punctuations = [',', '.', ':', ';', '?', '(', ')', '[', ']', '&', '!', '*',
                            '...', '...', '.....', '|', '`', '\'', '\'\'', '<', '>',

    filter_words = [w for w in words if (w not in english_stopwords)]
    filter_words = [w for w in filter_words if (w not in addition_stopwords)]
    filter_words = [w for w in filter_words if (w not in english_punctuations)]

    result = []
    for filter_word in filter_words:
        pattern = re.compile('.*[0-9]+.*') #用正则过滤后还是出现了数字,再过滤一次
        match = pattern.findall(filter_word)

        # 关于网址和邮箱的写法,不同文档也是五花八门,无奈写不出一个好的正则,只能在这里过滤掉
        if ('www' not in filter_word and '.com' not in filter_word and '@' not in filter_word) > 1 and not match):
            result.append(filter_word)

    return result

data_file_path = 'd:/data.txt'
output_file_path = 'd:/output.txt'
data_file = open(data_file_path, encoding='utf-8')

i = 1
texts = []
titles = []

```


#在保存语料时,我的做法是一行标题,一行内容。因此在取出时,需要将标题和内容合起来作为一篇文档

```
for line in data_file.readlines():
    if i % 2 == 1:
        titles.append(line)
        title = process_articles(line)
        texts.append(title)
    elif i % 2 == 0:
        body = process_articles(line)
        index = int(i / 2 - 1)
        texts[index] = texts[index] + body

    i = i + 1

data_file.close()

dictionary = corpora.Dictionary(texts)
corpus = [dictionary.doc2bow(text) for text in texts] #词频统计

lda = gensim.models.ldamodel.LdaModel(corpus=corpus, id2word=dictionary, num_topics=20)

print(lda.print_topics(num_topics=20, num_words=10))

#对每篇文档,标记主题分布
with open(output_file_path, 'w', encoding='utf-8') as f:
    for i in range(len(titles)):
        f.write(titles[i])
        topic_pro = lda.get_document_topics(corpus[i])
        f.write(str(topic_pro) + '\n')

f.close()
```

20个主题的打印结果:

```
[(0, '0.021*"food" + 0.015*"climate" + 0.013*"said" + 0.011*"change" + 0.010*"land" +
(1, '0.009*"said" + 0.008*"travel" + 0.007*"city" + 0.005*"people" + 0.005*"road" + 0.
(2, '0.006*"camera" + 0.006*"note" + 0.006*"mobile" + 0.006*"galaxy" + 0.006*"technolo
(3, '0.006*"new" + 0.006*"board" + 0.005*"research" + 0.005*"inc." + 0.005*"science" +
(4, '0.020*"china" + 0.016*"trade" + 0.013*"u.s." + 0.010*"said" + 0.009*"year" + 0.00
(5, '0.018*"india" + 0.013*"said" + 0.010*"kashmir" + 0.008*"pakistan" + 0.008*"govern
(6, '0.096*"market" + 0.043*"report" + 0.023*"global" + 0.022*"analysis" + 0.014*"indu
(7, '0.015*"said" + 0.013*"trump" + 0.012*"president" + 0.006*"state" + 0.006*"people"
(8, '0.060*"military" + 0.037*"wound" + 0.037*"air" + 0.029*"russia" + 0.026*"flights"
```

```
(9, '0.021*"league" + 0.021*"club" + 0.014*"madrid" + 0.013*"season" + 0.012*"united"
(10, '0.037*"said" + 0.016*"people" + 0.015*"would" + 0.014*"like" + 0.011*"school" +
(11, '0.023*"power" + 0.023*"energy" + 0.017*"electric" + 0.015*"car" + 0.014*"vehicle
(12, '0.010*"team" + 0.007*"year" + 0.007*"one" + 0.006*"first" + 0.006*"time" + 0.006
(13, '0.010*"film" + 0.008*"movie" + 0.006*"one" + 0.006*"like" + 0.005*"nuance" + 0.0
(14, '0.007*"also" + 0.007*"one" + 0.006*"new" + 0.006*"live" + 0.006*"tv" + 0.006*"ar
(15, '0.034*"hong" + 0.033*"kong" + 0.029*"china" + 0.020*"chinese" + 0.018*"said" + 0
(16, '0.054*"table" + 0.046*"market" + 0.045*"production" + 0.039*"global" + 0.036*"fi
(17, '0.014*"countries" + 0.012*"africa" + 0.010*"african" + 0.010*"said" + 0.009*"rus
(18, '0.040*"japan" + 0.037*"korea" + 0.032*"us" + 0.030*"south" + 0.027*"nuclear" + 0
(19, '0.018*"million" + 0.014*"company" + 0.013*"year" + 0.012*"quarter" + 0.008*"offi
```

打开主题标记结果文件: d:/output.txt, 奇数行表示文档的 title, 偶数行表示对应文档的主题分布

CNN Host Chris Cuomo Wanted All the Smoke, But C'mon Son, 'Fredo' Ain't the N-Word

[(7, 0.031960193), (10, 0.120341636), (12, 0.09453256), (13, 0.75074166)]

Jamaica extends 'state of emergency' travel warning over high levels of violent crime

[(1, 0.25734723), (5, 0.060726564), (7, 0.30127287), (12, 0.061933476), (15, 0.1399014

More than 500 people sign petition to save GP surgery in Derbyshire villlage

[(0, 0.086525), (1, 0.45324817), (3, 0.27085447), (5, 0.06869615), (12, 0.087278865),

Brews News: Cameron's Brewing pushes boundaries with Skeleton Crew

[(0, 0.027414821), (1, 0.15813638), (9, 0.026252491), (12, 0.06595557), (13, 0.0856866

US delays tariffs on some Chinese goods, drops others

[(2, 0.11514429), (4, 0.5850304), (7, 0.1876555), (11, 0.015779376), (12, 0.029962737)

2019 FIBA Basketball World Cup Preview: Details and Major Storylines

[(12, 0.9007839), (15, 0.0127624385), (17, 0.061190337), (18, 0.017240984)]

Hong Kong's business reputation takes hit with airport chaos

[(1, 0.116996825), (3, 0.026619915), (4, 0.07607849), (8, 0.087246716), (10, 0.0189819

Popular Dublin vegetarian restaurant temporarily closed after cockroaches found

[(0, 0.080911055), (1, 0.42190343), (7, 0.097431436), (13, 0.16738254), (14, 0.1229572

Daredevil Showrunner Has NSFW Reply For Charlie Cox Costume Hater

[(12, 0.2696757), (13, 0.6029142), (14, 0.124671556)]

Raytheon : Kingdom of Bahrain and U.S. sign agreement for Patriot | MarketScreener

[(2, 0.01222461), (3, 0.30114996), (7, 0.12225547), (8, 0.10267337), (16, 0.011453575)

Silver Prices Stare Down False Breakout Attempt amid Tariff News

[(4, 0.84105206), (12, 0.045362327), (19, 0.09444934)]

J&K with PoK should be integral part of India: Digvijay

[(5, 0.9610662), (13, 0.02438791)]



通过结果我们可以看到各个主题的编号以及各个主题中词出现的概率。比如主题 4, 应该是关于中美贸易主题; 主题 15, 应该是关于香港问题的主题(祖国万岁!)。这里我们只能知道主题编号, 通过提取高频词, 人为判定它的主题内容。引用一下陆奇的观点: 重叠的向量且能够解决问题, 它本身就是知识。在 AI 时代, 不应该把知识人化。lda 模型正是学习了计算机可以理解的主

题，编号为 0~20，所以说这本身并不是缺点。比如你要做一个新闻推荐，当用户阅读了一篇文章，且文章中出现主题 0, 1, 2 的概率比较大，那么我们就可以把其他文章中主题 0, 1, 2 概率较大的文章推荐给该用户，尽管我们不知道 0, 1, 2 代表什么意思。

另一点，同样的语料库，跑了两次 lda 结果不一样。读过上一章数学基础篇，了解 Gibbs 采样就会知道，每次的结果不会完全一样，但都是逼近于真实值。即分类出来的主题还是相同的，只是词的权重有一些差别。

lda 模型训练的关键在于主题数量 K 的选取，K 值太大，主题内容分得过细；K 值太小，主体内容分得过粗。最好是根据实际的场景，多次尝试（人为判定）。