法律声明

- 本课件包括演示文稿、示例、代码、题库、视频和声音等内容，小象学院和主讲老师拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意及内容，我们保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象
  - 新浪微博：ChinaHadoop

主题模型

邹博

主要内容

- LDA
  - 隐Dirichlet分布
  - Latent Dirichlet Allocation
- 先验分布 – 共轭分布
- Beta分布 – Dirichlet分布
- 三层贝叶斯网络模型LDA
- Gibbs采样和更新规则

LDA的应用方向

- 信息提取和搜索
  - 语义分析
- 文档分类/聚类、文章摘要、社区挖掘
- 基于内容的图像聚类、目标识别
  - 以及其他计算机视觉应用
- 生物信息数据的应用

朴素贝叶斯的分析

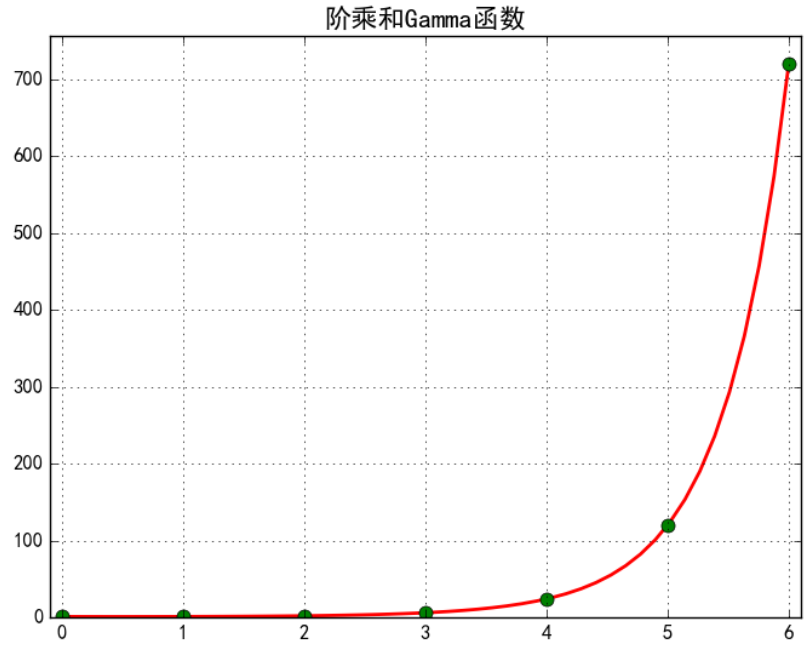
- 可以胜任许多文本分类问题。
- 无法解决语料中一词多义和多词一义的问题——它更像是词法分析，而非语义分析。
- 如果使用词向量作为文档的特征，一词多义和多词一义会造成计算文档间相似度的不准确性。
- 可以通过增加“主题”的方式，一定程度的解决上述问题：
  - 一个词可能被映射到多个主题中
    - ——一词多义
  - 多个词可能被映射到某个主题的概率很高
    - ——多词一义

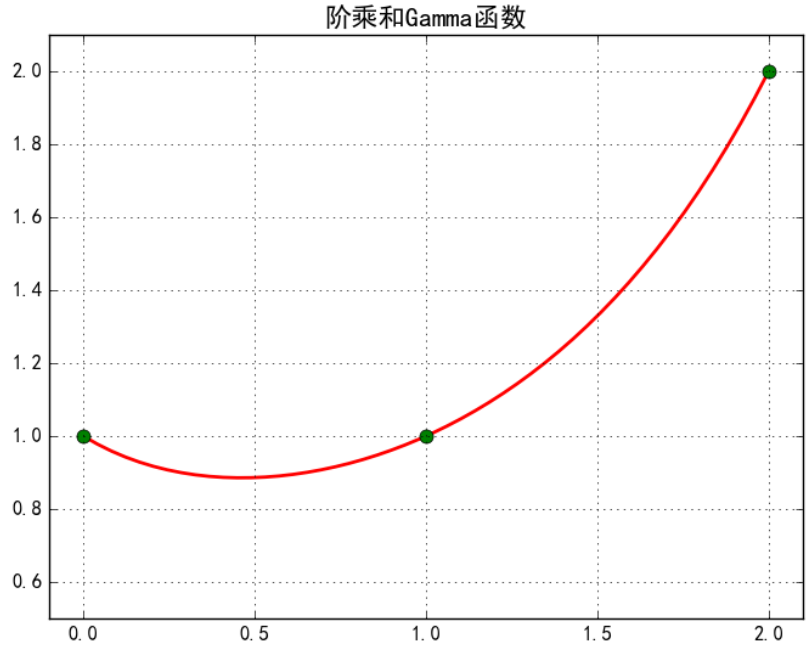
引：Γ函数

$$\Gamma(x)=(x-1)\cdot\Gamma(x-1)\Rightarrow\frac{\Gamma(x)}{\Gamma(x-1)}=x-1$$

- Γ函数是阶乘在实数上的推广

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





Beta分布

- Beta分布的概率密度：

$$f(x)=\begin{cases}\dfrac{1}{B(\alpha,\beta)}x^{\alpha-1}(1-x)^{\beta-1}, & x\in[0,1]\\ 0, & \text{其他}\end{cases}$$

- 其中系数B为：

$$B(\alpha,\beta)=\int_0^1 x^{\alpha-1}(1-x)^{\beta-1}dx=\frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

- Gamma函数可以看成阶乘的实数域推广：

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

$$\Rightarrow\Gamma(n)=(n-1)!\Rightarrow B(\alpha,\beta)=\frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

Beta分布的期望

$$f(x)=\frac{1}{B(\alpha,\beta)}x^{\alpha-1}(1-x)^{\beta-1},\ x\in[0,1]$$

$$B(\alpha,\beta)=\int_0^1 x^{\alpha-1}(1-x)^{\beta-1}dx=\frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

- 根据定义：

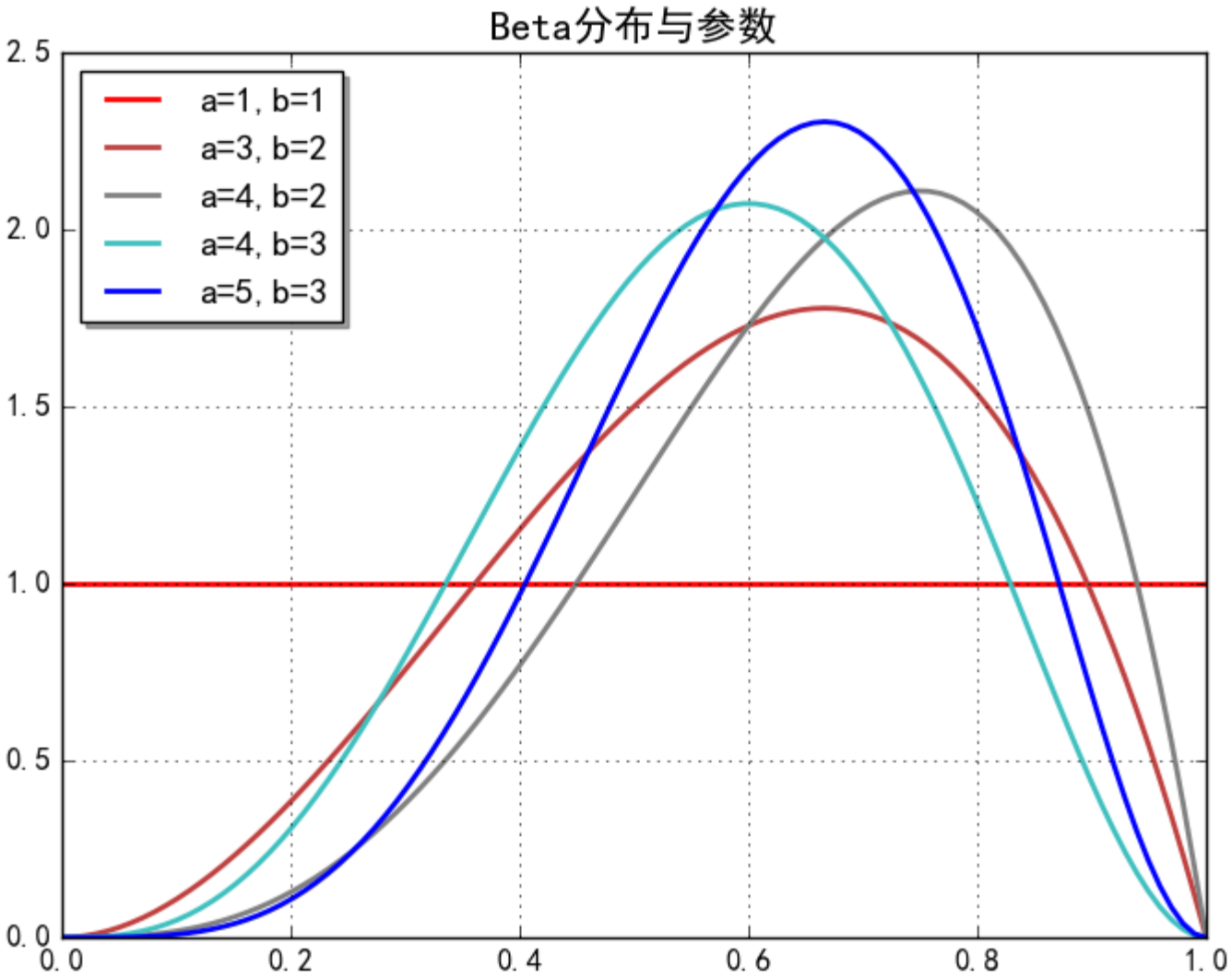
$$E(X)=\int_0^1 x\cdot\frac{1}{B(\alpha,\beta)}x^{\alpha-1}(1-x)^{\beta-1}dx$$

$$=\frac{1}{B(\alpha,\beta)}\int_0^1 x^{(\alpha+1)-1}(1-x)^{\beta-1}dx$$

$$=\frac{B(\alpha+1,\beta)}{B(\alpha,\beta)}=\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\Big/\frac{\Gamma(\alpha+\beta+1)}{\Gamma(\alpha+1)\Gamma(\beta)}$$

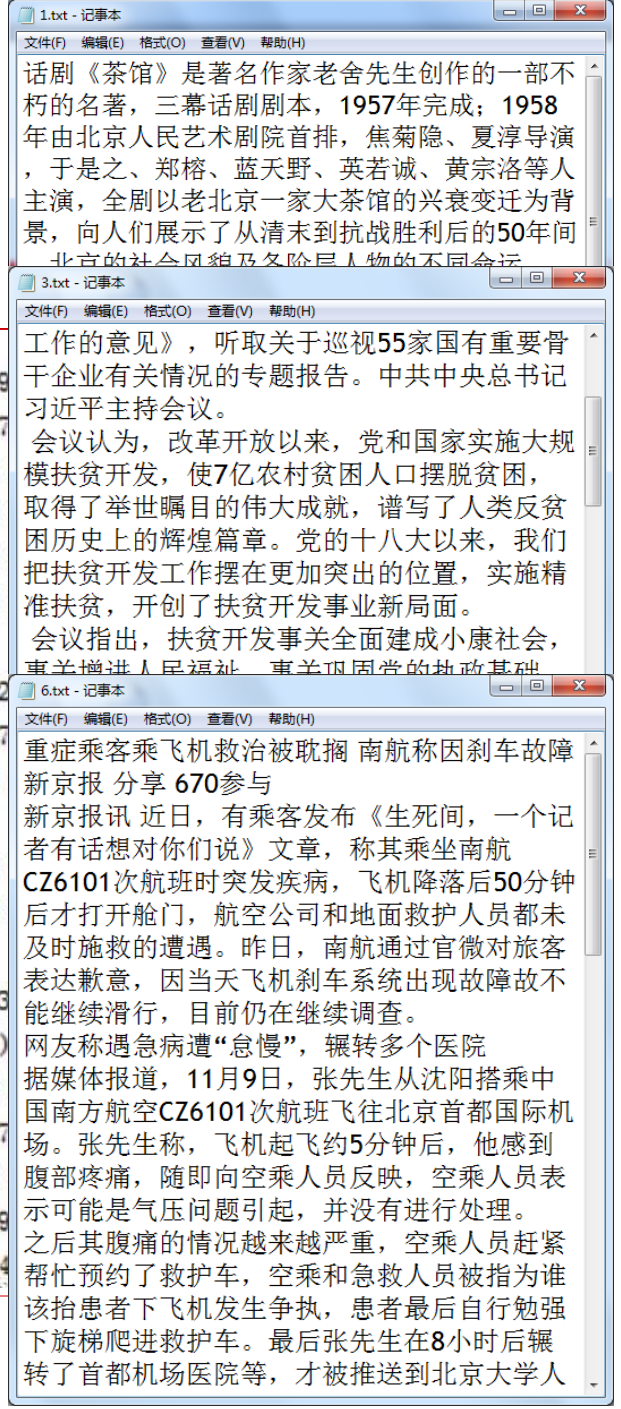
$$=\frac{\alpha}{\alpha+\beta}$$

Beta分布


Beta分布与参数
a=1, b=1
a=3, b=2
a=4, b=2
a=4, b=3
a=5, b=3
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

文档和主题

```
文档 1 : 茶馆 ( 0.0163591635916 ) 社会 ( 0.00528905289053 ) 王利发 ( 0.0052890528
文档 2 : 决议 ( 0.013898305084 ) 打击 ( 0.00824858757062 ) 安理会 ( 0.0082485875
文档 3 : 会议 ( 0.012449145646 ) 脱贫 ( 0.012449145646 ) 党校 ( 0.0108218063466
文档 4 : 美团 ( 0.0306066176471 ) 阿里 ( 0.0103860294118 ) 业务 ( 0.0103860294118
文档 5 : 户口 ( 0.0221347331584 ) 登记 ( 0.0195100612423 ) 人口 ( 0.0142607174103
文档 6 : 人员 ( 0.0111471861472 ) 飞机 ( 0.00898268398268 ) 称 ( 0.00681818181818
文档 7 : 号线 ( 0.0328544061303 ) 站 ( 0.0194444444444 ) 14 ( 0.0184865900383 )
文档 8 : 支付 ( 0.0198394495413 ) 腾讯 ( 0.0072247706422 ) 支付宝 ( 0.007224770642
文档 9 : 决议 ( 0.013898305084 ) 打击 ( 0.00824858757062 ) 安理会 ( 0.0082485875
文档 10 : 足协 ( 0.0186473429952 ) 足球 ( 0.0138164251208 ) 佩兰 ( 0.011884057971
==========
主题 1 : 美团 ( 0.0306066176471 ) 阿里 ( 0.0103860294118 ) 业务 ( 0.0103860294118
主题 2 : 会议 ( 0.012449145646 ) 脱贫 ( 0.012449145646 ) 党校 ( 0.0108218063466
主题 3 : 号线 ( 0.0328544061303 ) 站 ( 0.0194444444444 ) 14 ( 0.0184865900383 )
主题 4 : 人物 ( 0.00214876033058 ) 民族 ( 0.00214876033058 ) 资本家 ( 0.0021487603
主题 5 : 足协 ( 0.0186473429952 ) 足球 ( 0.0138164251208 ) 佩兰 ( 0.011884057971 )
主题 6 : 户口 ( 0.0221347331584 ) 登记 ( 0.0195100612423 ) 人口 ( 0.0142607174103
主题 7 : 决议 ( 0.013898305084 ) 打击 ( 0.00824858757062 ) 安理会 ( 0.0082485875
主题 8 : 人员 ( 0.0111471861472 ) 飞机 ( 0.00898268398268 ) 称 ( 0.00681818181818
主题 9 : 茶馆 ( 0.0163591635916 ) 社会 ( 0.00528905289053 ) 王利发 ( 0.0052890528
主题 10 : 支付 ( 0.0198394495413 ) 腾讯 ( 0.0072247706422 ) 支付宝 ( 0.0072247706
```



LDA涉及的主要问题

- 共轭先验分布
- Dirichlet分布
- LDA模型
  - Gibbs采样算法学习参数

共轭先验分布

- 由于x为给定样本，P(x)有时被称为“证据”，仅仅是归一化因子，如果不关心P(θ|x)的具体值，只考察θ取何值时后验概率P(θ|x)最大，则可将分母省去。

$$P(\theta \mid x) = \frac{P(x \mid \theta)P(\theta)}{P(x)} \propto P(x \mid \theta)P(\theta)$$

- 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。
- 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.

复习：二项分布的最大似然估计

- 投硬币试验中，进行N次独立试验，n次朝上，N-n次朝下。
- 假定朝上的概率为p，使用对数似然函数作为目标函数：

$$f(n \mid p) = \log\left(p^n (1-p)^{N-n}\right) \xrightarrow{\Delta} h(p)$$

$$\frac{\partial h(p)}{\partial p} = \frac{n}{p} - \frac{N-n}{1-p} \xrightarrow{\Delta} 0 \Rightarrow p = \frac{n}{N}$$

二项分布与先验举例

- 在校门口统计一定时间段内出入的男女生数目分别为N_B和N_G，估算该校男女生比例。

$$\begin{cases} P_B = \dfrac{N_B}{N_B + N_G} \\ P_G = \dfrac{N_G}{N_B + N_G} \end{cases}$$

- 若观察到4个女生和1个男生，可以得出该校女生比例是80%吗？
- 修正公式：

$$\begin{cases} P_B = \dfrac{N_B + 5}{N_B + N_G + 10} \\ P_G = \dfrac{N_G + 5}{N_B + N_G + 10} \end{cases} \Rightarrow \begin{cases} P_B = \dfrac{1+5}{1+4+10} = 40\% \\ P_G = \dfrac{4+5}{1+4+10} = 60\% \end{cases}$$

上述过程的理论解释

- 投掷一个非均匀硬币，可以使用参数为θ的伯努利模型，θ为硬币为正面的概率，那么结果x的分布形式为：$P(x|\theta)=C_n^k\cdot\theta^k\cdot(1-\theta)^{n-k}$
- 两点分布/二项分布的共轭先验是Beta分布，它具有两个参数α和β，Beta分布形式为

$$P(\theta\,|\,\alpha,\beta)=\begin{cases}\dfrac{1}{B(\alpha,\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}, & \theta\in[0,1]\\ 0, & \text{其他}\end{cases}$$

先验概率和后验概率的关系

- 根据似然和先验：

$$P(x|\theta)=C_n^k\cdot\theta^k\cdot(1-\theta)^{n-k}$$

$$P(\theta|\alpha,\beta)=\frac{1}{B(\alpha,\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}$$

计算后验概率：

$$P(\theta|x)=\frac{P(x|\theta)\cdot P(\theta)}{P(x)}\propto P(x|\theta)\cdot P(\theta)$$

$$=\left(C_n^k\theta^k(1-\theta)^{n-k}\right)\cdot\left(\frac{1}{B(\alpha,\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}\right)$$

$$=\frac{C_n^k}{B(\alpha,\beta)}\theta^{(k+\alpha)-1}(1-\theta)^{(n-k+\beta)-1}$$

$$\propto\frac{1}{B(k+\alpha,n-k+\beta)}\theta^{(k+\alpha)-1}(1-\theta)^{(n-k+\beta)-1}$$

- 后验概率是参数为(k+α,n-k+β)的Beta分布，即：伯努利分布/二项分布的共轭先验是Beta分布。

$$P(\theta)=\frac{1}{B(\alpha,\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}$$

伪计数

$$P(\theta|x)=\frac{1}{B(k+\alpha,n-k+\beta)}\theta^{(k+\alpha)-1}(1-\theta)^{(n-k+\beta)-1}$$

- 参数α、β是决定参数θ的参数，即超参数。
- 在后验概率的最终表达式中，参数α、β和k、n-k一起作为参数θ的指数——后验概率的参数为(k+α,n-k+β)。
- 根据这个指数的实践意义：投币过程中，正面朝上的次数，α和β先验性的给出了在没有任何实验的前提下，硬币朝上的概率分配；因此，α和β可被称作“伪计数”。

共轭先验的直接推广

- 从2到K：
  - 二项分布→多项分布
  - Beta分布→Dirichlet分布

Dirichlet分布

- Beta分布：$$f(x)=\begin{cases}\dfrac{1}{B(\alpha,\beta)}x^{\alpha-1}(1-x)^{\beta-1}, & x\in[0,1]\\ 0, & \text{其他}\end{cases}$$
  - 其中：$B(\alpha,\beta)=\dfrac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$
- Dirichlet分布：$$f(\vec{p}\,|\,\vec{\alpha})=\begin{cases}\dfrac{1}{\Delta(\vec{\alpha})}\prod_{k=1}^{K}p_k^{\alpha_k-1}, & p_k\in[0,1]\\ 0, & \text{其他}\end{cases}$$
  - 简记：$Dir(\vec{p}\,|\,\vec{\alpha})=\dfrac{1}{\Delta(\vec{\alpha})}\prod_{k=1}^{K}p_k^{\alpha_k-1}$ 其中：$\Delta(\vec{\alpha})=\dfrac{\prod_{k=1}^{K}\Gamma(\alpha_k)}{\Gamma\left(\sum_{k=1}^{K}\alpha_k\right)}$

Dirichlet分布的期望

- 根据Beta分布的期望公式：

$$f(x)=\frac{1}{B(\alpha,\beta)}x^{\alpha-1}(1-x)^{\beta-1},\ x\in[0,1]\Rightarrow E(X)=\frac{\alpha}{\alpha+\beta}$$

- 推广得到：

$$f(\vec{p}\,|\,\vec{\alpha})=\frac{1}{\Delta(\vec{\alpha})}\prod_{k=1}^{K}p_k^{\alpha_k-1},\ p\in[0,1]\Rightarrow E(p_i)=\frac{\alpha_i}{\sum_{k=1}^{K}\alpha_k}$$

Dirichlet分布分析 $Dir(\vec{p}\,|\,\vec{\alpha})=\frac{1}{\Delta(\vec{\alpha})}\prod_{k=1}^{K}p_k^{\alpha_k-1}$

- α是参数向量，共K个
- 定义在$x_1,x_2\ldots x_{K-1}$维上
  - $x_1+x_2+\ldots+x_{K-1}+x_K=1$
  - $x_1,x_2\ldots x_{K-1}>0$
  - 定义在(K-1)维的单纯形上，其他区域的概率密度为0
- α的取值对Dir(p|α)有什么影响？

Symmetric Dirichlet distribution

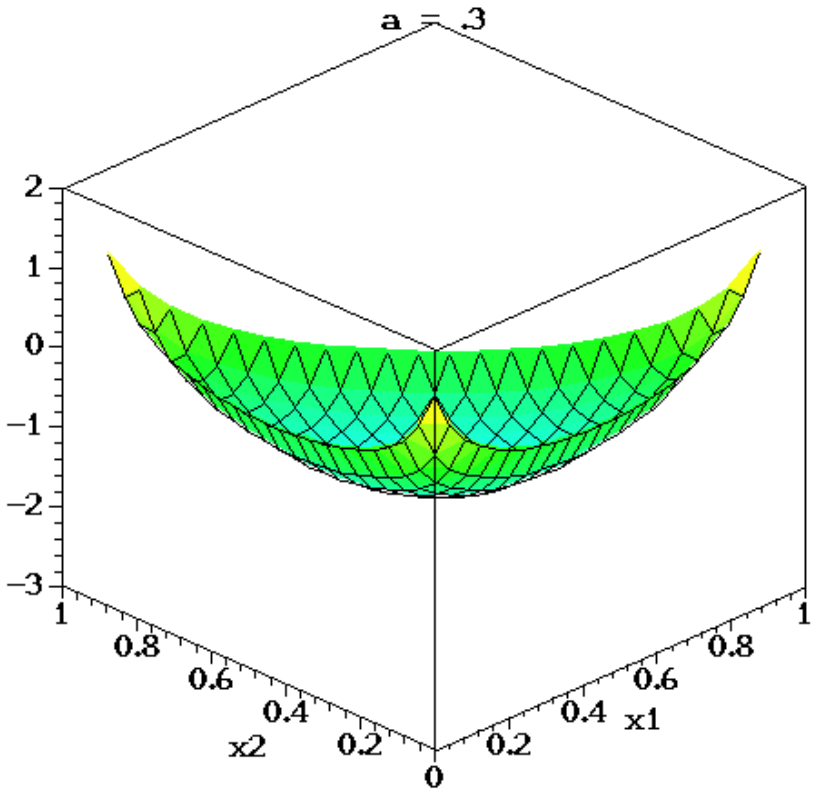
- A very common special case is the symmetric Dirichlet distribution, where all of the elements making up the parameter vector have the same value. Symmetric Dirichlet distributions are often used when a Dirichlet prior is called for, since there typically is *no prior knowledge* favoring one component over another. Since all elements of the parameter vector have the same value, the distribution alternatively can be parametrized by a single scalar value α, called the concentration parameter(聚集参数).

对称Dirichlet分布

- Dirichlet分布：$Dir(\vec{p}\,|\,\vec{\alpha}) = \frac{1}{\Delta(\vec{\alpha})}\prod_{k=1}^{K} p_k^{\alpha_k - 1}$
  - 其中：$\Delta(\vec{\alpha}) = \frac{\prod_{k=1}^{K}\Gamma(\alpha_k)}{\Gamma\left(\sum_{k=1}^{K}\alpha_k\right)}$
- 对称Dirichlet分布：$Dir(\vec{p}\,|\,\alpha, K) = \frac{1}{\Delta_K(\alpha)}\prod_{k=1}^{K} p_k^{\alpha - 1}$
  - 其中：$\Delta_K(\vec{\alpha}) = \frac{\Gamma^K(\alpha_k)}{\Gamma(K\cdot\alpha)}$

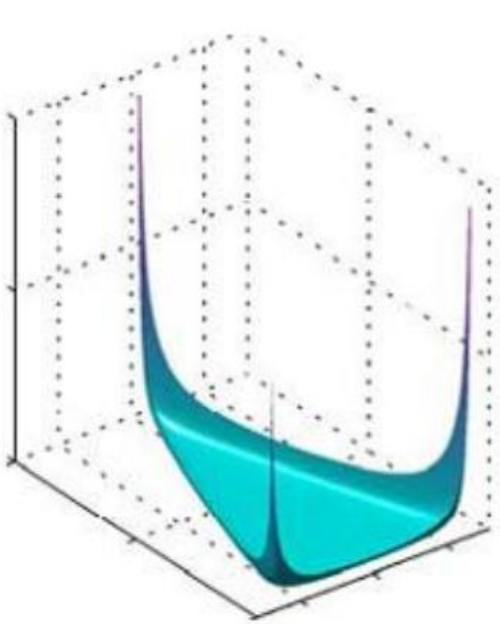
对称Dirichlet分布的参数分析

- α=1时
  - 退化为均匀分布
- 当α>1时
  - $p_1=p_2=\ldots=p_k$的概率增大
- 当α<1时
  - $p_i=1$，$p_{非i}=0$的概率增大

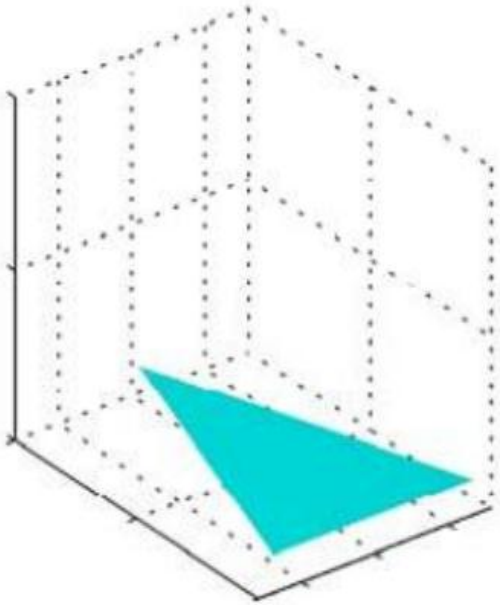


图像说明：将Dirichlet分布的概率密度函数取对数,绘制对称Dirichlet分布的图像，取K=3，也就是有两个独立参数x1,x2，分别对应图中的两个坐标轴，第三个参数始终满足x3=1−x1−x2且α1=α2=α3=α，图中反映的是α从0.3变化到2.0的概率对数值的变化情况。

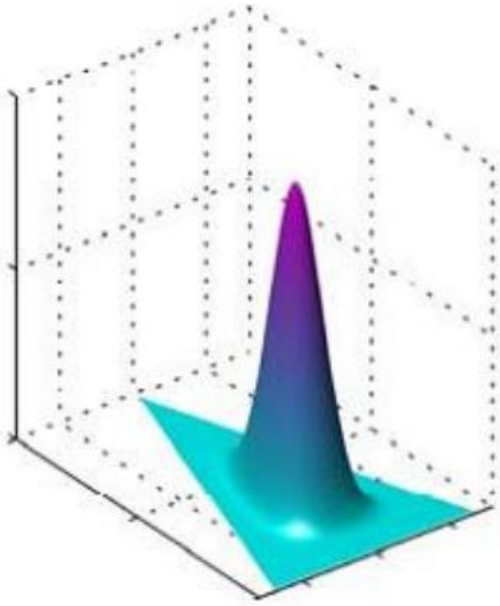
参数α对Dirichlet分布的影响



$\{\alpha_k\} = 0.1$



$\{\alpha_k\} = 1$



$\{\alpha_k\} = 10$

参数选择对对称Dirichlet分布的影响

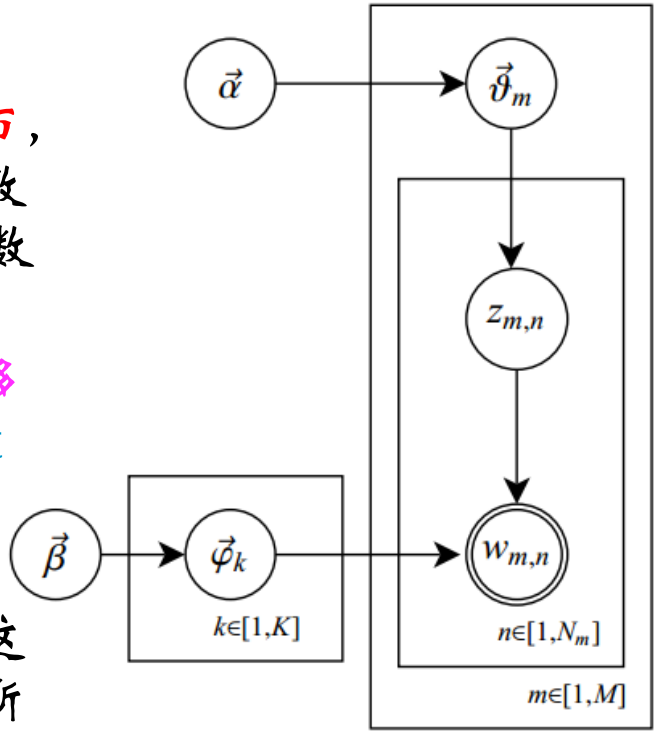
- When α=1, the symmetric Dirichlet distribution is equivalent to a uniform distribution over the open standard (K−1)-simplex, i.e. it is uniform over all points in its support. Values of the concentration parameter above 1 prefer variants that are dense, evenly distributed distributions, i.e. all the values within a single sample are similar to each other. Values of the concentration parameter below 1 prefer sparse distributions, i.e. most of the values within a single sample will be close to 0, and the vast majority of the mass will be concentrated in a few of the values.

多项分布的共轭分布是Dirichlet分布

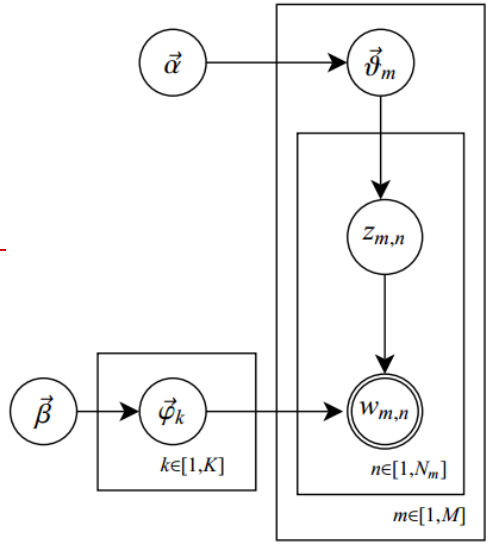
$$
\begin{aligned}
\boldsymbol{\alpha} &= (\alpha_1, \cdots, \alpha_K) &&= \text{concentration hyperparameter} \\
\mathbf{p} \mid \boldsymbol{\alpha} &= (p_1, \cdots, p_K) &&\sim \operatorname{Dir}(K, \boldsymbol{\alpha}) \\
\mathbb{X} \mid \mathbf{p} &= (\mathbf{x}_1, \cdots, \mathbf{x}_K) &&\sim \operatorname{Cat}(K, \mathbf{p}) \\
\mathbf{c} &= (c_1, \cdots, c_K) &&= \text{number of occurrences of category } i \\
\mathbf{p} \mid \mathbb{X}, \boldsymbol{\alpha} &\sim \operatorname{Dir}(K, \mathbf{c} + \boldsymbol{\alpha}) &&= \operatorname{Dir}(K, c_1 + \alpha_1, \cdots, c_K + \alpha_K)
\end{aligned}
$$

LDA的解释

- 共有m篇文章，一共涉及了K个主题；
- 每篇文章(长度为N_m)都有各自的主题分布，主题分布是多项分布，该多项分布的参数服从Dirichlet分布，该Dirichlet分布的参数为α；
- 每个主题都有各自的词分布，词分布为多项分布，该多项分布的参数服从Dirichlet分布，该Dirichlet分布的参数为β；
- 对于某篇文章中的第n个词，首先从该文章的主题分布中采样一个主题，然后在这个主题对应的词分布中采样一个词。不断重复这个随机生成过程，直到m篇文章全部完成上述过程。



详细解释

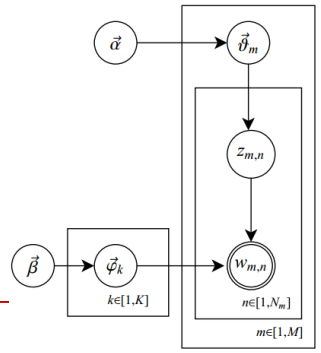


- 字典中共有V个term(不可重复)，这些term出现在具体的文章中，就是word——在具体某文章中的word当然是有可能重复的。
- 语料库中共有m篇文档$d_1,d_2 \ldots d_m$；
- 对于文档d_i，由N_i个word组成，可重复；
- 语料库中共有K个主题T_1，$T_2 \ldots T_k$；
- α和β为先验分布的参数，一般事先给定：如取0.1的对称Dirichlet分布——表示在参数学习结束后，期望每个文档的主题不会十分集中。
- θ是每篇文档的主题分布
  - 对于第i篇文档d_i的主题分布是$\boldsymbol{\theta}_i=(\theta_{i1}, \theta_{i2} \ldots, \theta_{iK})$，是长度为K的向量；
- 对于第i篇文档d_i，在主题分布$\boldsymbol{\theta}_i$下，可以确定一个具体的主题$z_{ij}=k$，$k \in [1,K]$，
- φ_k表示第k个主题的词分布，$k \in [1,K]$
  - 对于第k个主题T_k的词分布$\varphi_k=(\varphi_{k1}, \varphi_{k2} \ldots \varphi_{kv})$，是长度为v的向量
- 由z_{ij}选择φ_{zij}，表示由词分布φ_{zij}确定term，即得到观测值w_{ij}。

详细解释

- 图中K为主题个数，M为文档总数，Nm是第m个文档的单词总数。β是每个Topic下词的多项分布的Dirichlet先验参数，α是每个文档下Topic的多项分布的Dirichlet先验参数。zmn是第m个文档中第n个词的主题，wmn是m个文档中的第n个词。两个隐含变量θ和φ分别表示第m个文档下的Topic分布和第k个Topic下词的分布，前者是k维(k为Topic总数)向量，后者是v维向量(v为词典中term总数)

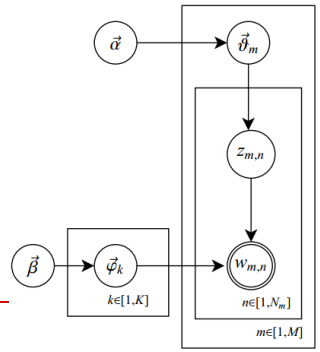
参数的学习



- 给定一个文档集合，w_{mn}是可以观察到的已知变量，α和β是根据经验给定的先验参数，其他的变量z_{mn}、θ、φ都是未知的隐含变量，需要根据观察到的变量来学习估计的。根据LDA的图模型，可以写出所有变量的联合分布：

$$p(\vec{w}_m, \vec{z}_m, \vec{\vartheta}_m, \underline{\Phi}|\vec{\alpha}, \vec{\beta}) = \prod_{n=1}^{N_m} p(w_{m,n}|\vec{\varphi}_{z_{m,n}}) p(z_{m,n}|\vec{\vartheta}_m) \cdot p(\vec{\vartheta}_m|\vec{\alpha}) \cdot p(\underline{\Phi}|\vec{\beta})$$

似然概率



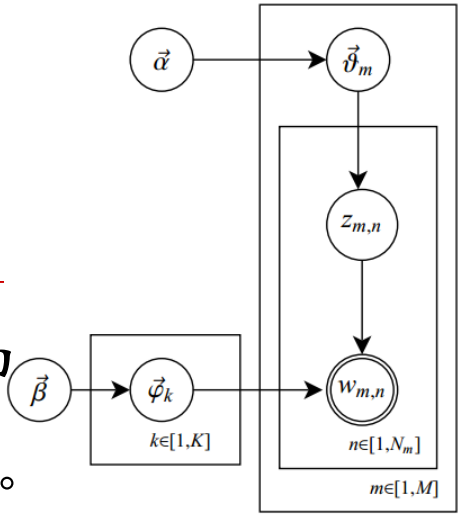
- 一个词w_{mn}初始化为一个词t的概率是

$$p(w_{m,n}=t|\vec{\vartheta}_m, \underline{\Phi}) = \sum_{k=1}^{K} p(w_{m,n}=t|\vec{\varphi}_k)p(z_{m,n}=k|\vec{\vartheta}_m)$$

- 每个文档中出现主题k的概率乘以主题k下出现词t的概率，然后枚举所有主题求和得到。整个文档集合的似然函数为：

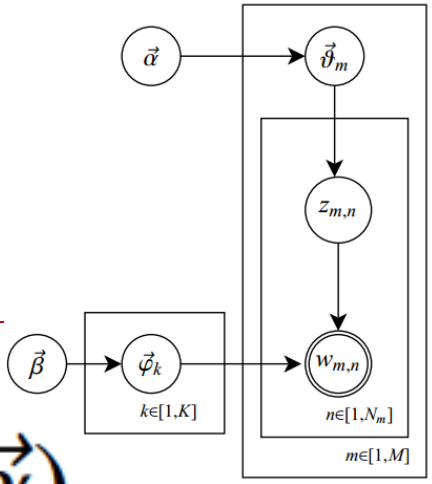
$$p(\mathcal{W}|\underline{\Theta}, \underline{\Phi}) = \prod_{m=1}^{M} p(\vec{w}_m|\vec{\vartheta}_m, \underline{\Phi}) = \prod_{m=1}^{M}\prod_{n=1}^{N_m} p(w_{m,n}|\vec{\vartheta}_m, \underline{\Phi})$$

Gibbs Sampling



- Gibbs Sampling算法的运行方式是每次选取概率向量的一个维度，给定其他维度的变量值采样当前维度的值。不断迭代直到收敛输出待估计的参数。
- 初始时随机给文本中的每个词分配主题$z^{(0)}$，然后统计每个主题z下出现词t的数量以及每个文档m下出现主题z的数量，每一轮计算$p(z_i|\mathbf{z}_{-i},\mathbf{d},\mathbf{w})$，即排除当前词的主题分布：
  - 根据其他所有词的主题分布估计当前词分配各个主题的概率。
- 当得到当前词属于所有主题z的概率分布后，根据这个概率分布为该词采样一个新的主题。
- 用同样的方法更新下一个词的主题，直到发现每个文档的主题分布θ_i和每个主题的词分布φ_j收敛，算法停止，输出待估计的参数θ和φ，同时每个单词的主题z_{mn}也可同时得出。
- 实际应用中会设置最大迭代次数。每一次计算$p(zi|\mathbf{z}_{-i},\mathbf{d},\mathbf{w})$的公式称为Gibbs updating rule。

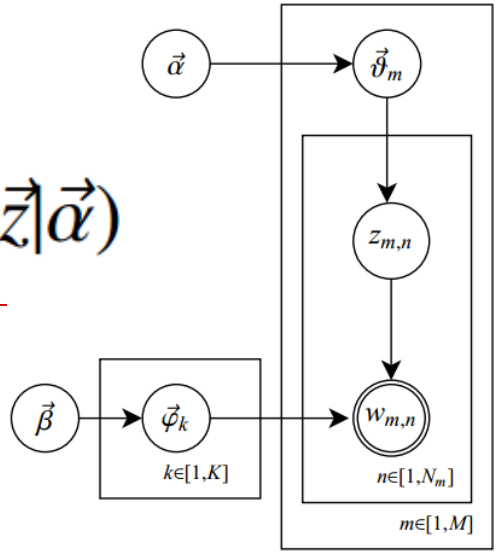
联合分布



$$p(\vec{w},\vec{z}|\vec{\alpha},\vec{\beta}) = p(\vec{w}|\vec{z},\vec{\beta})p(\vec{z}|\vec{\alpha})$$

- 第一项因子是给定主题采样词的过程
- 后面的因子计算，$n_z^{(t)}$表示词t被观察到分配给主题z的次数，$n_m^{(k)}$表示主题k分配给文档m的次数。

计算因子 $p(\vec{w},\vec{z}|\vec{\alpha},\vec{\beta}) = p(\vec{w}|\vec{z},\vec{\beta})p(\vec{z}|\vec{\alpha})$



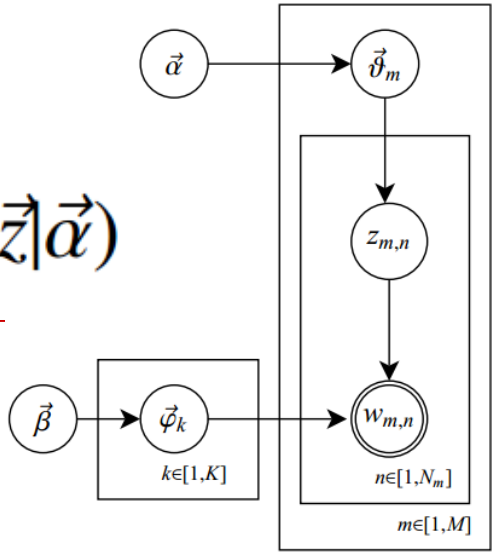
$$p(\vec{w}|\vec{z},\vec{\beta}) = \int p(\vec{w}|\vec{z},\underline{\Phi})\ p(\underline{\Phi}|\vec{\beta})\ \mathrm{d}\underline{\Phi}$$

$$= \int \prod_{z=1}^{K} \frac{1}{\Delta(\vec{\beta})} \prod_{t=1}^{V} \varphi_{z,t}^{n_z^{(t)}+\beta_t-1}\ \mathrm{d}\vec{\varphi}_z$$

$$= \prod_{z=1}^{K} \frac{\Delta(\vec{n}_z+\vec{\beta})}{\Delta(\vec{\beta})}, \quad \vec{n}_z = \{n_z^{(t)}\}_{t=1}^{V}$$

$$\int_{\vec{p}} \prod_{k=1}^{K} p_k^{\alpha_k-1} d\vec{p} = \Delta(\vec{\alpha})$$

计算因子 $p(\vec{w},\vec{z}|\vec{\alpha},\vec{\beta}) = p(\vec{w}|\vec{z},\vec{\beta})p(\vec{z}|\vec{\alpha})$



$$p(\vec{z}|\vec{\alpha}) = \int p(\vec{z}|\underline{\Theta})\ p(\underline{\Theta}|\vec{\alpha})\ \mathrm{d}\underline{\Theta}$$

$$= \int \prod_{m=1}^{M} \frac{1}{\Delta(\vec{\alpha})} \prod_{k=1}^{K} \vartheta_{m,k}^{n_m^{(k)}+\alpha_k-1}\ \mathrm{d}\vec{\vartheta}_m$$

$$= \prod_{m=1}^{M} \frac{\Delta(\vec{n}_m+\vec{\alpha})}{\Delta(\vec{\alpha})}, \qquad \vec{n}_m = \{n_m^{(k)}\}_{k=1}^{K}$$

$$\int_{\vec{p}} \prod_{k=1}^{K} p_k^{\alpha_k-1} d\vec{p} = \Delta(\vec{\alpha})$$

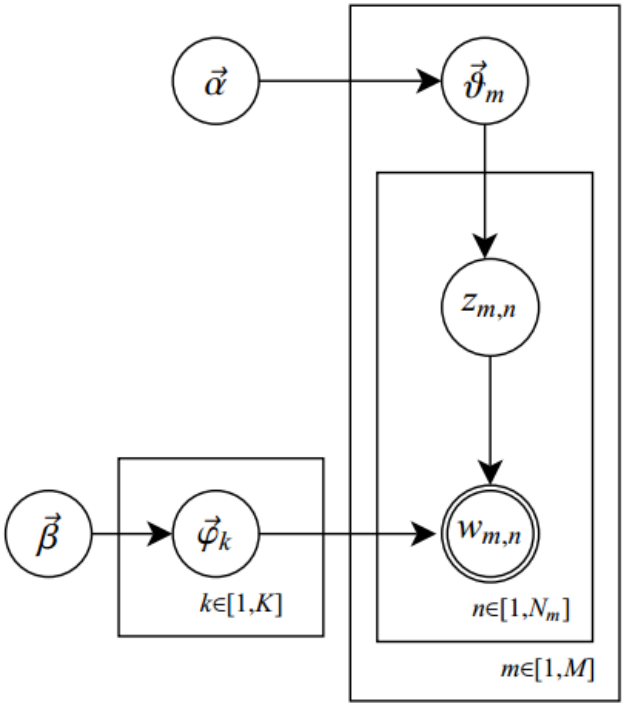
Gibbs updating rule

$$
\begin{aligned}
p(z_i{=}k|\vec{z}_{\neg i}, \vec{w}) &= \frac{p(\vec{w}, \vec{z})}{p(\vec{w}, \vec{z}_{\neg i})} = \frac{p(\vec{w}|\vec{z})}{p(\vec{w}_{\neg i}|\vec{z}_{\neg i})p(w_i)} \cdot \frac{p(\vec{z})}{p(\vec{z}_{\neg i})} \\
&\propto \frac{\Delta(\vec{n}_z + \vec{\beta})}{\Delta(\vec{n}_{z,\neg i} + \vec{\beta})} \cdot \frac{\Delta(\vec{n}_m + \vec{\alpha})}{\Delta(\vec{n}_{m,\neg i} + \vec{\alpha})} \\
&= \frac{\Gamma(n_k^{(t)} + \beta_t)\,\Gamma(\sum_{t=1}^{V} n_{k,\neg i}^{(t)} + \beta_t)}{\Gamma(n_{k,\neg i}^{(t)} + \beta_t)\,\Gamma(\sum_{t=1}^{V} n_k^{(t)} + \beta_t)} \cdot \frac{\Gamma(n_m^{(k)} + \alpha_k)\,\Gamma(\sum_{k=1}^{K} n_{m,\neg i}^{(k)} + \alpha_k)}{\Gamma(n_{m,\neg i}^{(k)} + \alpha_k)\,\Gamma(\sum_{k=1}^{K} n_m^{(k)} + \alpha_k)} \\
&= \frac{n_{k,\neg i}^{(t)} + \beta_t}{\sum_{t=1}^{V} n_{k,\neg i}^{(t)} + \beta_t} \cdot \frac{n_{m,\neg i}^{(k)} + \alpha_k}{[\sum_{k=1}^{K} n_m^{(k)} + \alpha_k] - 1} \\
&\propto \frac{n_{k,\neg i}^{(t)} + \beta_t}{\sum_{t=1}^{V} n_{k,\neg i}^{(t)} + \beta_t} (n_{m,\neg i}^{(k)} + \alpha_k)
\end{aligned}
$$

词分布和主题分布

$$\varphi_{k,t} = \frac{n_k^{(t)} + \beta_t}{\sum_{t=1}^{V} n_k^{(t)} + \beta_t}$$

$$\vartheta_{m,k} = \frac{n_m^{(k)} + \alpha_k}{\sum_{k=1}^{K} n_m^{(k)} + \alpha_k}$$



$$p(\vec{\vartheta}_m|\vec{z}_m,\vec{\alpha}) = \frac{1}{Z_{\vartheta_m}} \prod_{n=1}^{N_m} p(z_{m,n}|\vec{\vartheta}_m) \cdot p(\vec{\vartheta}_m|\vec{\alpha}) = \mathrm{Dir}(\vec{\vartheta}_m|\vec{n}_m + \vec{\alpha})$$

$$p(\vec{\varphi}_k|\vec{z},\vec{w},\vec{\beta}) = \frac{1}{Z_{\varphi_k}} \prod_{\{i:z_i=k\}} p(w_i|\vec{\varphi}_k) \cdot p(\vec{\varphi}_k|\vec{\beta}) = \mathrm{Dir}(\vec{\varphi}_k|\vec{n}_k + \vec{\beta})$$

Gibbs采样算法

Algorithm `LdaGibbs`$(\{\vec{w}\}, \alpha, \beta, K)$
Input: word vectors $\{\vec{w}\}$, hyperparameters α, β, topic number K
Global data: count statistics $\{n_m^{(k)}\}$, $\{n_k^{(t)}\}$ and their sums $\{n_m\}$, $\{n_k\}$, memory for full conditional array $p(z_i|\cdot)$
Output: topic associations $\{\vec{z}\}$, multinomial parameters $\underline{\Phi}$ and $\underline{\Theta}$, hyperparameter estimates α, β

```
// initialisation
```
zero all count variables, $n_m^{(k)}, n_m, n_k^{(t)}, n_k$
for all documents $m \in [1, M]$ **do**
    for all words $n \in [1, N_m]$ in document m **do**
        sample topic index $z_{m,n}{=}k \sim \text{Mult}(1/K)$
        increment document–topic count: $n_m^{(k)}$ += 1
        increment document–topic sum: n_m += 1
        increment topic–term count: $n_k^{(t)}$ += 1
        increment topic–term sum: n_k += 1

```
// Gibbs sampling over burn-in period and sampling period
```
while not finished **do**
    for all documents $m \in [1, M]$ **do**
        for all words $n \in [1, N_m]$ in document m **do**
            `// for the current assignment of` k `to a term` t `for word` $w_{m,n}$`:`
            decrement counts and sums: $n_m^{(k)} -= 1; n_m -= 1; n_k^{(t)} -= 1; n_k -= 1$
            `// multinomial sampling acc. to Eq. 78 (decrements from previous step):`
            sample topic index $\tilde{k} \sim p(z_i|\vec{z}_{\neg i}, \vec{w})$
            `// for the new assignment of` $z_{m,n}$ `to the term` t `for word` $w_{m,n}$`:`
            increment counts and sums: $n_m^{(\tilde{k})}$ += 1; n_m += 1; $n_{\tilde{k}}^{(t)}$ += 1; $n_{\tilde{k}}$ += 1

    `// check convergence and read out parameters`
    if converged and L sampling iterations since last read out **then**
        `// the different parameters read outs are averaged.`
        read out parameter set $\underline{\Phi}$ according to Eq. 81
        read out parameter set $\underline{\Theta}$ according to Eq. 82

代码实现

- 数目：
  - 文档数目：M
  - 词数目：V(非重复的，“term”)
  - 主题数目：K
- 记号：
  - 用d表述第几个文档，k表示主题，w表示词汇(term)，n表示词(word)

三个矩阵和三个向量

- z[d][w]：第d篇文档的第w个词来自哪个主题。M行，X列，X为相应文档长度：即词(可重复)的数目。
- nw[w][t]：第w个词是第t个主题的次数。word-topic矩阵，列向量nw[][t]表示主题t的词频数分布；V行K列
- nd[d][t]：第d篇文档中第t个主题出现的次数，doc-topic矩阵，行向量nd[d]表示文档d的主题频数分布。M行，K列。
- 辅助向量：
  - ntSum[t]：第t个主题在所有语料出现的次数，K维
  - ndSum[d]：第d篇文档中词的数目(可重复)，M维；
  - P[t]：对于当前计算的某词属于主题t的概率，K维。

Code

```
if __name__ == "__main__":
    doc_num = 10  # 文档数目
    # 载入停止词库
    stop_words = load_stopwords()
    dic = {}
    doc = read_document(doc_num, stop_words, dic)

    # LDA
    term_num = len(dic) # 词汇的数目
    # nt[w][t]: 第term个词属于第t个主题的次数
    nt = [[0 for t in range(topic_number)] for term in range(term_num)]
    # nd[d][t]: 第d个文档中出现第t个主题的次数
    nd = [[0 for t in range(topic_number)] for d in range(doc_num)]
    # nt_sum[t]: 第t个主题出现的次数(nt矩阵的第t列)
    nt_sum = [0 for t in range(topic_number)]
    # nd_sum[d]: 第d个文档的长度(nd矩阵的第d行)
    nd_sum = [0 for d in range(doc_num)]
    z = init_topic(doc, nt, nd, nt_sum, nd_sum, dic)
    theta, phi = lda(z, nt, nd, nt_sum, nd_sum, dic, doc)
    show_result(theta, phi, dic)    # 输出每个文档的主题和每个主题的关键字
```

Code

```
def lda(z, nt, nd, nt_sum, nd_sum, dic, doc):
    doc_num = len(z)
    for time in range(50):
        for m in range(doc_num):
            doc_length = len(z[m])
            for i in range(doc_length):
                term = dic[doc[m][i]]    # 词语 -> 词汇
                gibbs_sampling(z, m, i, nt, nd, nt_sum, nd_sum, term)
    theta = calc_theta(nd, nd_sum)     # 计算每个文档的主题分布
    phi = calc_phi(nt, nt_sum)       # 计算每个主题的词分布
    return theta, phi
```

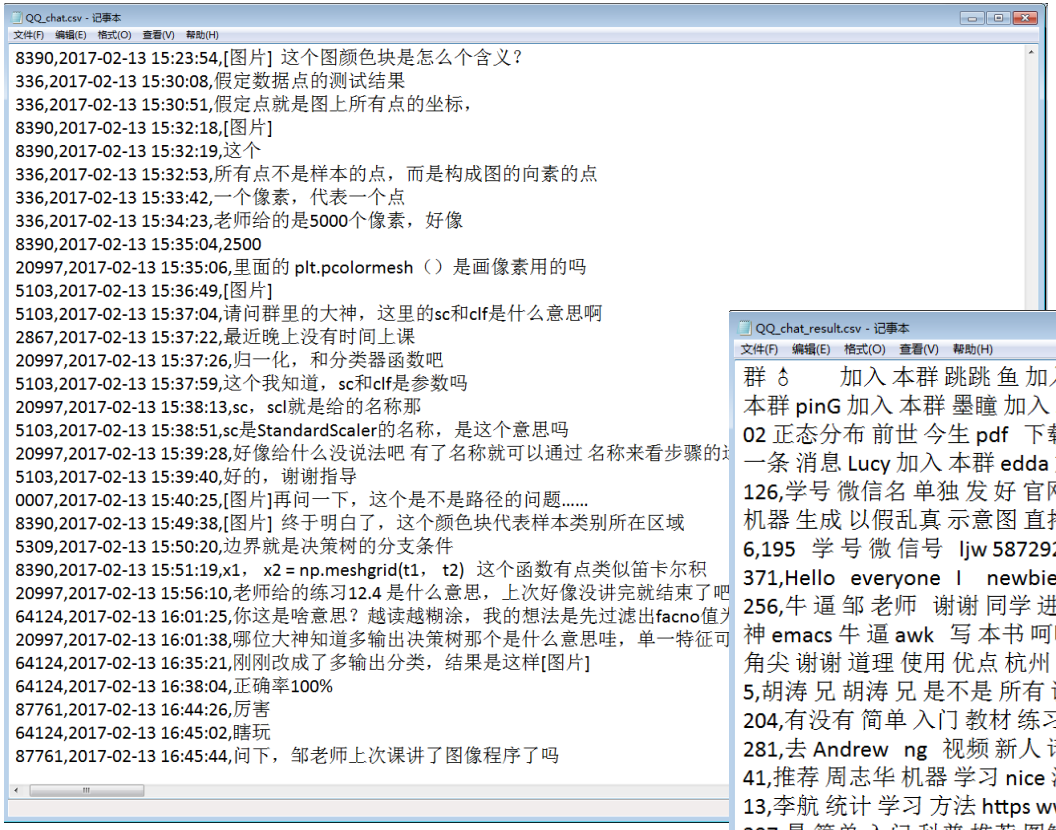
Code

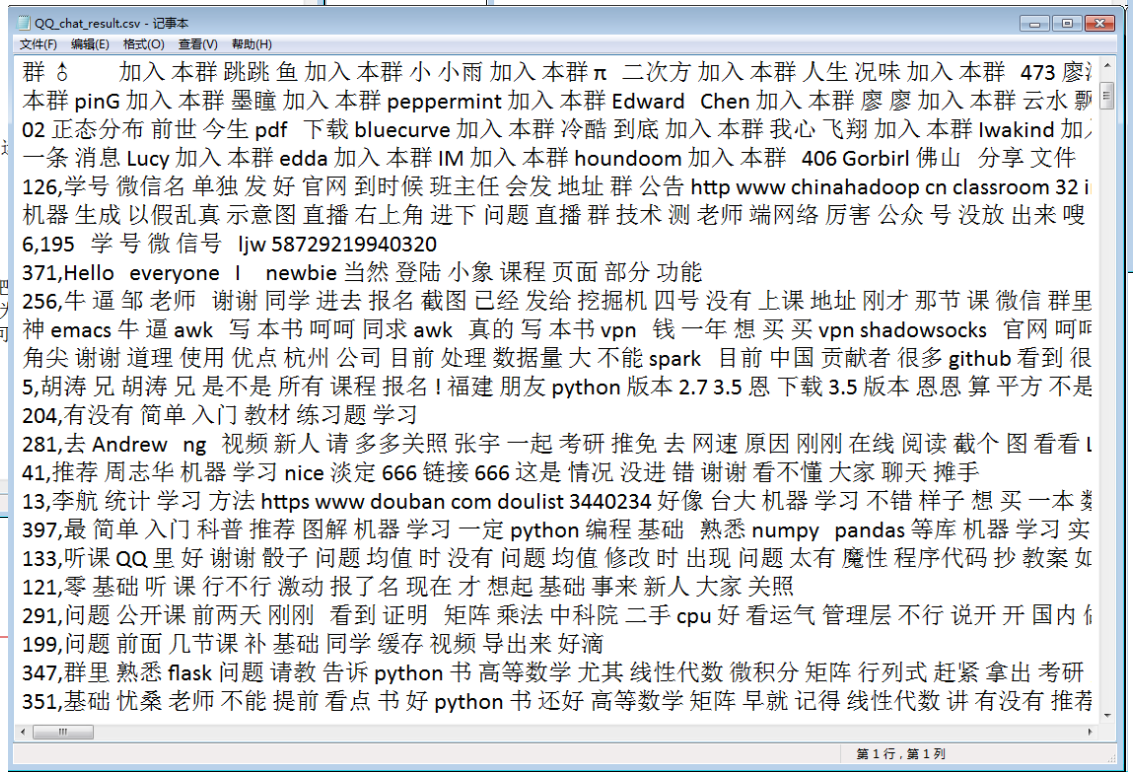
```
def calc_theta(nd, nd_sum):    # 每个文档的主题分布
    doc_num = len(nd)
    topic_alpha = topic_number * alpha
    theta = [[0 for t in range(topic_number)] for d in range(doc_num)]
    for m in range(doc_num):
        for k in range(topic_number):
            theta[m][k] = (nd[m][k] + alpha) / (nd_sum[m] + topic_alpha)
    return theta


def calc_phi(nt, nt_sum):    # 每个主题的词分布
    term_num = len(nt)
    term_beta = term_num * beta
    phi = [[0 for w in range(term_num)] for t in range(topic_number)]
    for k in range(topic_number):
        for term in range(term_num):
            phi[k][term] = (nt[term][k] + beta) / (nt_sum[k] + term_beta)
    return phi
```

文档和主题

文档 1 ： 茶馆 （ 0.0163591635916 ） 社会 （ 0.00528905289053 ） 王利发 （ 0.00528905289053 ）
文档 2 ： 决议 （ 0.0138983050847 ） 打击 （ 0.00824858757062 ） 安理会 （ 0.00824858757062 ）
文档 3 ： 会议 （ 0.0124491456469 ） 脱贫 （ 0.0124491456469 ） 党校 （ 0.0108218063466 ）
文档 4 ： 美团 （ 0.0306066176471 ） 阿里 （ 0.0103860294118 ） 业务 （ 0.0103860294118 ）
文档 5 ： 户口 （ 0.0221347331584 ） 登记 （ 0.0195100612423 ） 人口 （ 0.0142607174103 ）
文档 6 ： 人员 （ 0.0111471861472 ） 飞机 （ 0.00898268398268 ） 称 （ 0.00681818181818 ）
文档 7 ： 号线 （ 0.0328544061303 ） 站 （ 0.0194444444444 ） 14 （ 0.0184865900383 ）
文档 8 ： 支付 （ 0.0198394495413 ） 腾讯 （ 0.0072247706422 ） 支付宝 （ 0.0072247706422 ）
文档 9 ： 决议 （ 0.0138983050847 ） 打击 （ 0.00824858757062 ） 安理会 （ 0.00824858757062 ）
文档 10 ： 足协 （ 0.0186473429952 ） 足球 （ 0.0138164251208 ） 佩兰 （ 0.011884057971 ）
==========
主题 1 ： 美团 （ 0.0306066176471 ） 阿里 （ 0.0103860294118 ） 业务 （ 0.0103860294118 ）
主题 2 ： 会议 （ 0.0124491456469 ） 脱贫 （ 0.0124491456469 ） 党校 （ 0.0108218063466 ）
主题 3 ： 号线 （ 0.0328544061303 ） 站 （ 0.0194444444444 ） 14 （ 0.0184865900383 ）
主题 4 ： 人物 （ 0.00214876033058 ） 民族 （ 0.00214876033058 ） 资本家 （ 0.00214876033058 ）
主题 5 ： 足协 （ 0.0186473429952 ） 足球 （ 0.0138164251208 ） 佩兰 （ 0.011884057971 ）
主题 6 ： 户口 （ 0.0221347331584 ） 登记 （ 0.0195100612423 ） 人口 （ 0.0142607174103 ）
主题 7 ： 决议 （ 0.0138983050847 ） 打击 （ 0.00824858757062 ） 安理会 （ 0.00824858757062 ）
主题 8 ： 人员 （ 0.0111471861472 ） 飞机 （ 0.00898268398268 ） 称 （ 0.00681818181818 ）
主题 9 ： 茶馆 （ 0.0163591635916 ） 社会 （ 0.00528905289053 ） 王利发 （ 0.00528905289053 ）
主题 10 ： 支付 （ 0.0198394495413 ） 腾讯 （ 0.0072247706422 ） 支付宝 （ 0.0072247706422 ）

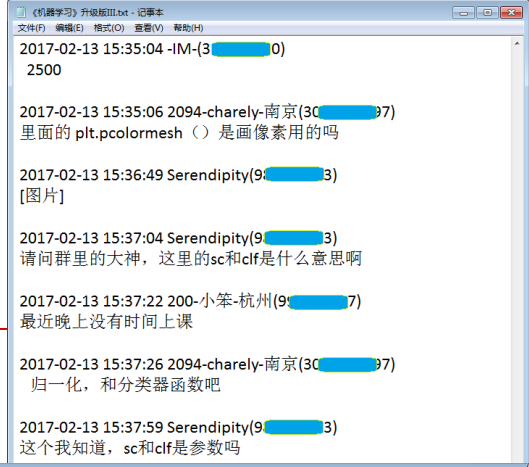
聊天记录分析感兴趣话题

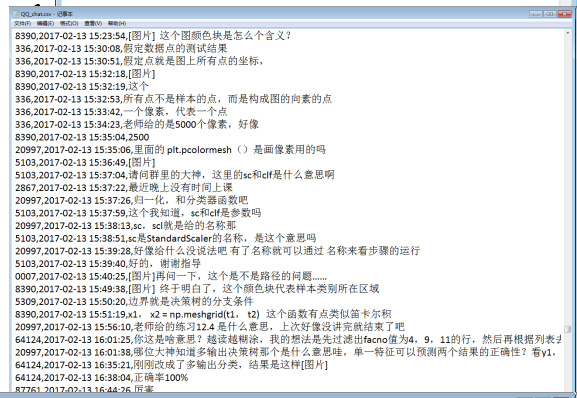


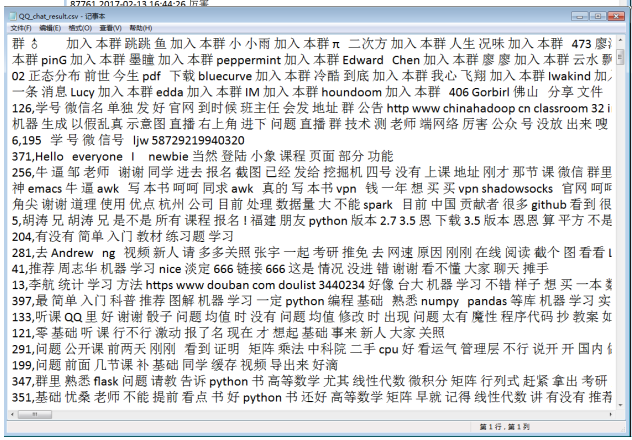


数据处理流程

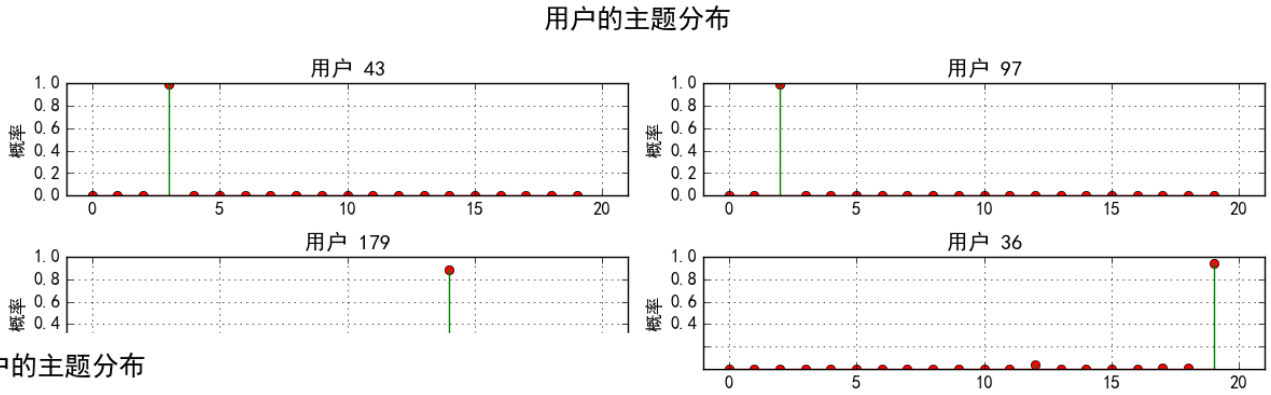
- 获取QQ群聊天记录：txt文本格式(图1)
- 整理成“QQ号/时间/留言”的规则形
  - 正则表达式
  - 清洗特定词：表情、@XX
  - 使用停止词库
  - 获得csv表格数据(图2)
- 合并相同QQ号的留言
  - 长文档利于计算每人感兴趣话题(图3)
- LDA模型计算主题
  - 调参与可视化
- 计算每个QQ号及众人感兴趣话题

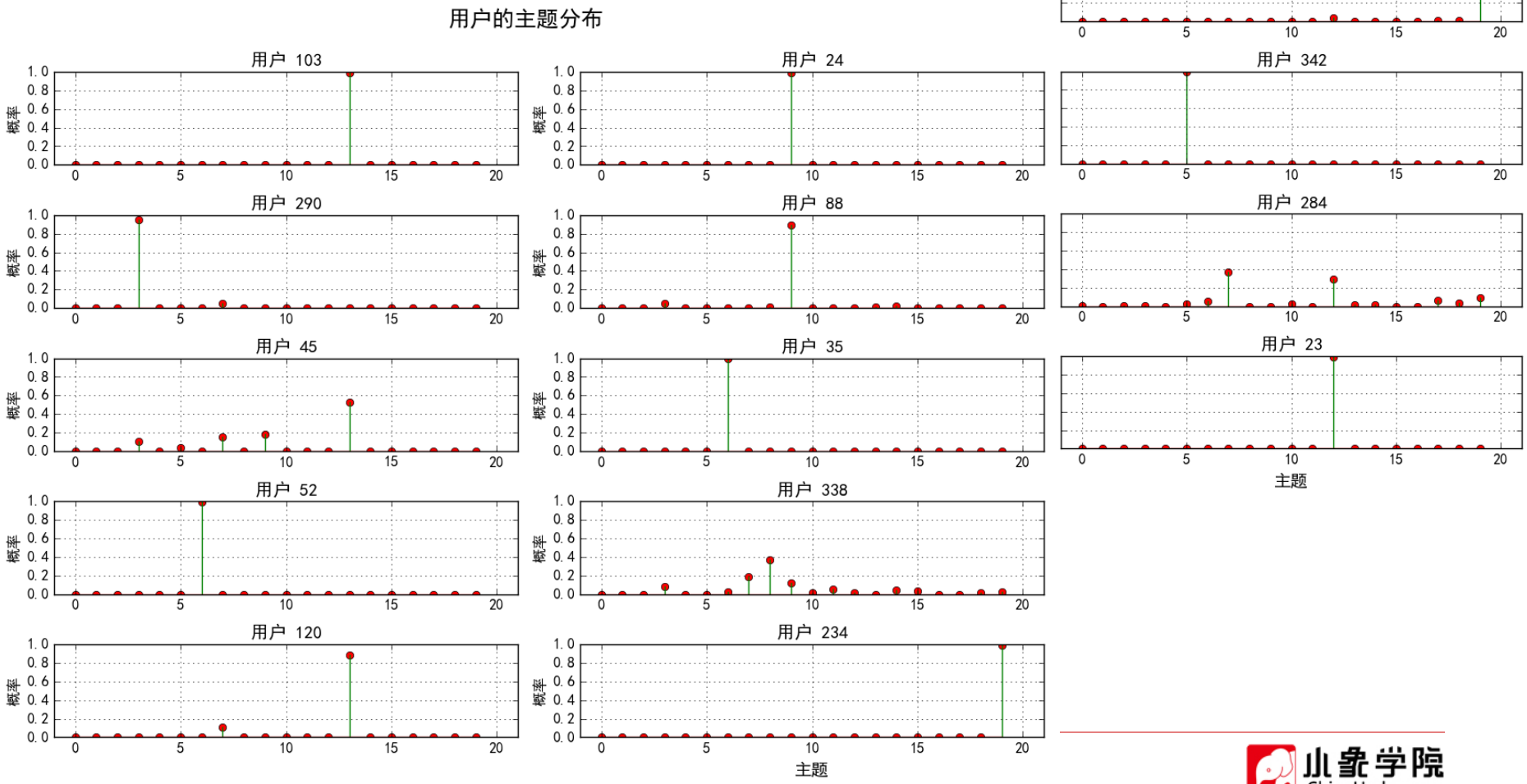






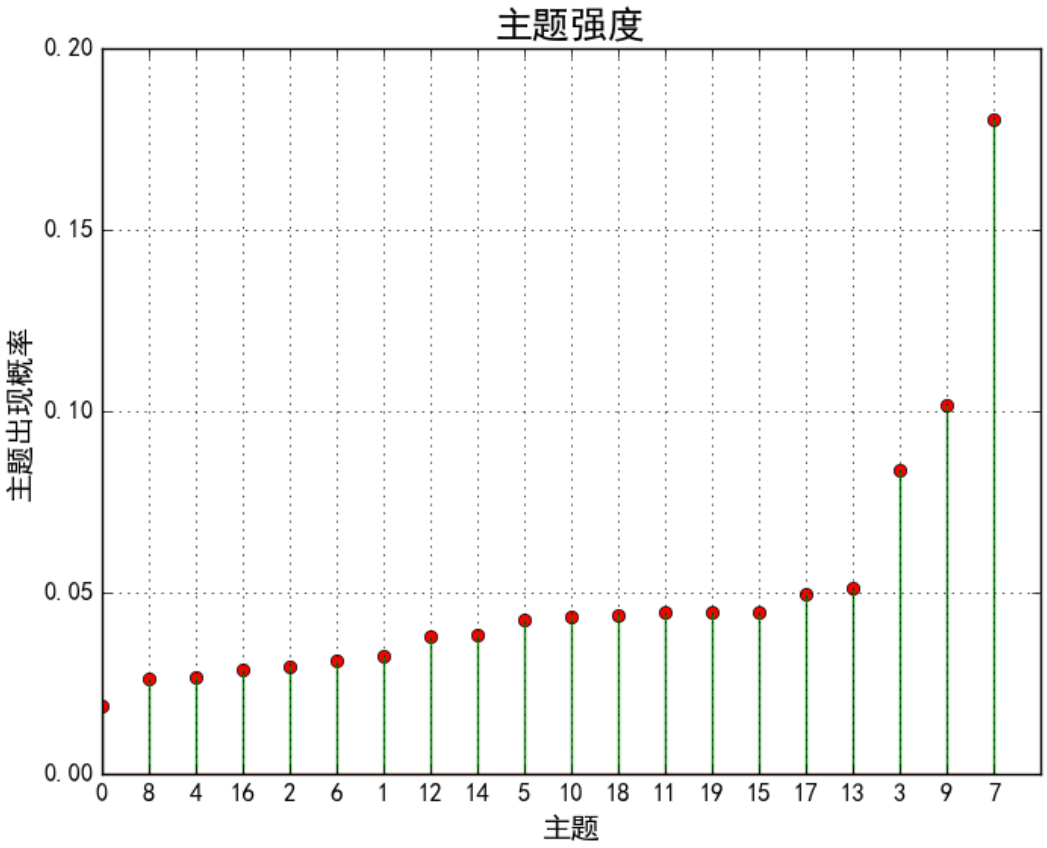
主题分布





感兴趣话题

主题#1: 下周 用到 绑定 matlab 手册 详细描述 666
概率: [0.00386318 0.00379716 0.00275161 0.00267257 0.0025779 0.002
主题#2: 决策树 下载 代码 新入 记得 无法 if
概率: [0.00459593 0.00368178 0.00332368 0.00322059 0.00282207 0.002
主题#3: 下载 视频 app 中 cn 电脑
概率: [0.01650318 0.00643555 0.00534376 0.00493769 0.00477131 0.004
主题#4: 互扫 上装 停留 一份 层次 缺 找点
概率: [0.00457096 0.00322019 0.00314435 0.00283468 0.0027911 0.002
主题#5: 同问 变量 极限 训练 毕竟 问
概率: [0.00799138 0.00796053 0.00353821 0.00350646 0.00323556 0.003
主题#6: 上网 元素 等待 白屏 中国 在线看
概率: [0.00678803 0.00387909 0.00340132 0.00335809 0.00303858 0.002
主题#7: 好 大家 老师 学习 没有 谢谢
概率: [0.02435813 0.01137977 0.01116381 0.01048067 0.01000842 0.008
主题#8: 参考书 需不需要 离散 型 基础知识 组成
概率: [0.00909812 0.00337894 0.00242403 0.00229328 0.00220396 0.00215262 0.00209999]
主题#9: com 91pintuan wxe9c061d2aed9ae09 group https 请
概率: [0.01010167 0.00642439 0.00517232 0.00517232 0.00495921 0.00485321 0.00467808]
主题#10: 画面 端 找回 QQ 群 账号
概率: [0.00641685 0.00384681 0.00359311 0.00355134 0.00343251 0.00288641 0.00272948]
主题#11: 直播 今天 一片空白 没有 进不去 问题
概率: [0.01035459 0.00352562 0.00291557 0.00276692 0.00275323 0.00259631 0.00257422]
主题#12: 太过分 编程 讲完 闪照 配置 这部分
概率: [0.00510845 0.00290508 0.00289446 0.00273205 0.00265899 0.00251825 0.00249154]
主题#13: pydotplus 学号 积分 浏览器 弄 请问
概率: [0.01348479 0.00478899 0.00342937 0.00260794 0.00236339 0.00226667 0.00213521]
主题#14: 分享 两个 满足 话 京东 需求
概率: [0.0143761 0.00353596 0.00270139 0.00264514 0.00255197 0.00249422 0.00244258]
主题#15: 相等 听不见 openCV 版 数据库 设置
概率: [0.01212347 0.00449718 0.0030489 0.0030018 0.00285451 0.00281434 0.00263035]
主题#16: 回放 课程 OpenStack 牛云 投放 点哇
概率: [0.01010629 0.00425028 0.0026812 0.00238546 0.00221883 0.00221883 0.00221883]
主题#17: 福 敬业 早 断 厉害 歌



Code

```
def segment():
    stopwords = load_stopwords()
    data = pd.read_csv('QQ_chat.csv', header=0)
    for i, info in enumerate(data['Info']):
        info_words = []
        words = jieba.cut(info)
        for word in words:
            if word not in stopwords:
                info_words.append(word.encode('utf-8'))
        if info_words:
            data.iloc[i, 2] = ' '.join(info_words)
        else:
            data.iloc[i, 2] = np.nan
    data.dropna(axis=0, how='any', inplace=True)
    data.to_csv('QQ_chat_segment.csv', sep=',', header=True, index=False)


def combine():
    data = pd.read_csv('QQ_chat_segment.csv', header=0)
    data['QQ'] = pd.Categorical(data['QQ']).codes
    f_output = open('QQ_chat_result.csv', mode='w')
    f_output.write('QQ,Info\n')
    for qq in data['QQ'].unique():
        info = ' '.join(data[data['QQ'] == qq]['Info'])
        str = '%s,%s\n' % (qq, info)
        f_output.write(str)
    f_output.close()
```

```
def clean_info(info):
    replace_str = (('\n', ''), ('\r', ''), (',', ', '), ('表情', ''))
    for rs in replace_str:
        info = info.replace(rs[0], rs[1])

    at_pattern = re.compile(r'(@.* )')
    at = re.findall(pattern=at_pattern, string=info)
    for a in at:
        info = info.replace(a, '')
    idx = info.find('@')
    if idx != -1:
        info = info[:idx]
    return info


def regularize_data():
    time_pattern = re.compile(r'\d{4}-\d{2}-\d{2} \d{1,2}:\d{1,2}:\d{1,2}')
    qq_pattern1 = re.compile(r'([1-9][0-9]{4,})')    # QQ号最小是10000
    qq_pattern2 = re.compile(r'(\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*)')
    f = open(u'《机器学习》升级版III.txt')
    f_output = open(u'QQ_chat.csv', mode='w')
    f_output.write('QQ,Time,Info\n')
    qq = chat_time = info = ''
    for line in f:
        line = line.strip()
        if line:
            t = re.findall(pattern=time_pattern, string=line)
            qq1 = re.findall(pattern=qq_pattern1, string=line)
            qq2 = re.findall(pattern=qq_pattern2, string=line)
            if (len(t) >= 1) and ((len(qq1) >= 1) or (len(qq2) >= 1)):
                if info:
                    info = clean_info(info)
                    if info:
                        info = '%s,%s,%s\n' % (qq, chat_time, info)
                        f_output.write(info)
                        info = ''
                if len(qq1) >= 1:
                    qq = qq1[0]
                else:
                    qq = qq2[0][0]
                chat_time = t[0]
            else:
                info += line
    f.close()
    f_output.close()
```

超参数的确定

- 交叉验证
- α表达了不同文档间主题是否鲜明，β度量了有多少近义词能够属于同一个类别。
- 主题数目K，词项数目为W，可以使用：
  - α=50/K
  - β=200/W
  - 注：不一定普遍适用

一种迭代求超参数的方法

- Digamma函数：

$$\Psi(x)=\frac{d\ln\Gamma(x)}{dx}=\frac{\Gamma'(x)}{\Gamma(x)}$$

- 迭代公式：(T. Minka)

$$\alpha_k=\frac{\left(\left(\sum_{m=1}^{M}\Psi\left(n_m^{(k)}+\alpha_k\right)\right)-M\cdot\Psi(\alpha_k)\right)}{\left(\sum_{m=1}^{M}\Psi\left(n_m+\sum_{j=1}^{K}\alpha_j\right)\right)-M\cdot\Psi\left(\sum_{j=1}^{K}\alpha_j\right)}\cdot\alpha_k$$

主题个数的确定

- 相似度最小
- 选取初始的主题个数K，训练LDA模型，计算各主题之间的相似度
- 增加或减少K的值，重新训练LDA模型，再次计算topic之间的相似度
- 选择相似度最小的模型所对应的K作为主题个数。

概率分布的困惑度/复杂度Perplexity

- 某离散概率分布p的困惑度为：

$$Perplexity = 2^{H(p)} = 2^{-\sum_x p(x)\log_2 p(x)}$$

- 样本集x1,x2…xn的估计模型q的困惑度为：

$$Perplexity = a^{-\frac{1}{N}\sum_{i=1}^{n}\log_a q(x_i)}$$，a为任意整数

  - 交叉熵为：$H(p,q) = -\sum_x p(x)\log_2 q(x)$

困惑度Perplexity与主题模型

- 使用训练数据得到无监督模型，在测试数据集中计算所有token似然值几何平均数的倒数
  - 测试数据集中词典大小的期望

$$P(W \mid Model) = \prod_{i=1}^{V} p(w_i \mid Model)^{-\frac{1}{V}} = \exp\left(-\frac{1}{V} \cdot \sum_{i=1}^{V} \log p(w_i \mid Model)\right)$$

- 其中，LDA中词的似然概率为：

$$P(\vec{w}_m \mid Model) = \prod_{n=1}^{N_m} \sum_{k=1}^{K} p(z=k \mid d=m) \cdot p(w=t \mid z=k) = \prod_{t=1}^{V} \left(\sum_{k=1}^{K} \vartheta_{m,k} \cdot \varphi_{k,t}\right)^{n_m^{(t)}}$$

附：PageRank

- 一个网页i的重要度可以使用指向网页i的其他网页j的重要度加权得到。
  - 权值不妨取网页j包含的链接数目。

$$D(P_i)=(1-d)+d\cdot\sum_{j\in In(P_i)}\frac{1}{|Out(P_j)|}\cdot D(P_j)$$

- 参数的意义为：
  - 网页i的中重要性 $D(P_i)$
  - 阻尼系数d，如设置为常系数0.85
  - 指向网页i的网页集合 $In(P_i)$
  - 网页j指向的网页集合 $Out(P_j)$

TextRank

- 将PageRank中的“网页”换成“词”，结论仍成立。
  - 选择合适的窗口大小，认为窗口内的词相互指向。

$$D(w_i)=(1-d)+d\cdot\sum_{j\in In(w_i)}\frac{1}{|Out(w_j)|}\cdot D(w_j)$$

- 句子Si和Sj的相似度：

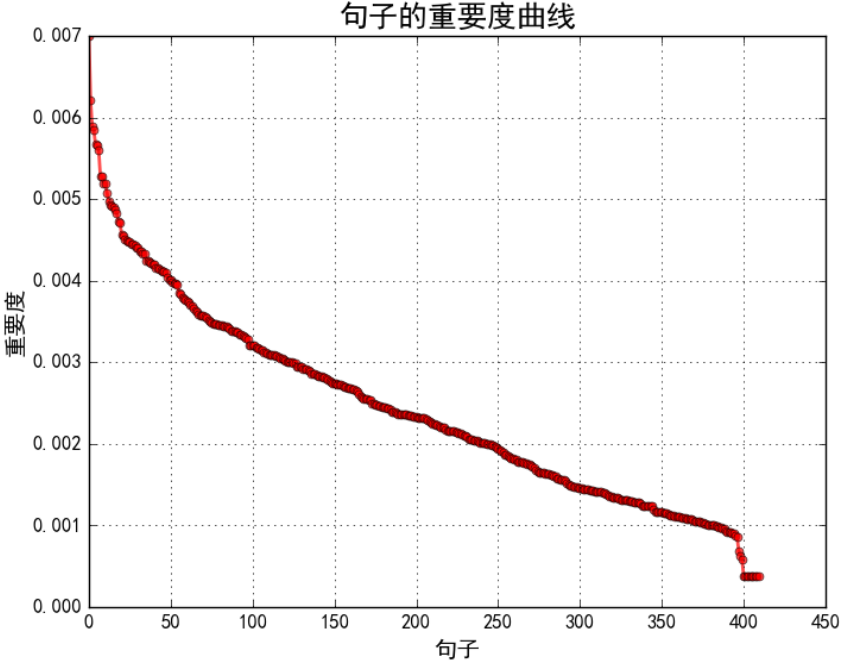
$$Similar(S_i,S_j)=\frac{|\{w_k \mid w_k\in S_i \,\&\, w_k\in S_j\}|}{\log|S_i|+\log|S_j|}$$

- 将PageRank中“网页”换成“句子”，结论仍然基本成立，只需考虑将“链接”加权：

$$D(S_i)=(1-d)+d\cdot\sum_{j\in In(S_i)}\frac{similar(S_j,S_i)}{\sum_{k\in Out(S_j)}similar(S_j,S_k)}\cdot D(S_j)$$

Text Rank

```
tr4s = TextRank4Sentence()
tr4s.analyze(text=text, lower=True, source = 'no_stop_words')
data = pd.DataFrame(data=tr4s.key_sentences)
mpl.rcParams['font.sans-serif'] = [u'SimHei']
mpl.rcParams['axes.unicode_minus'] = False
plt.figure(facecolor='w')
plt.plot(data['weight'], 'ro-', lw=2, ms=5, alpha=0.7)
plt.grid(b=True)
plt.xlabel(u'句子', fontsize=14)
plt.ylabel(u'重要度', fontsize=14)
plt.title(u'句子的重要度曲线', fontsize=18)
plt.show()

key_sentences = tr4s.get_key_sentences(num=20, sentence_min_len=4)
for sentence in key_sentences:
    print sentence['weight'], sentence['sentence']
```



0.00699560759634 她知道我心里的苦闷，知道不该阻止我出去走走，知道我要是老呆在家里结果会更糟，但她又担心我一个人在那荒僻的园子里整天都想些什么

0.00621160375013 这一来你中了魔了，整天都在想哪一件事可以写，哪一个人可以让你写成小说

0.00588860912528 那时她的儿子，还太年轻，还来不及为母亲想，他被命运击昏了头，一心以为自己是世上最不幸的一个，不知道儿子的不幸在母亲那儿总是要加倍的

0.00584459738866 她想，只要儿子能活下去哪怕自己去死呢也行，可她又确信一个人不能仅仅是活着，儿子得有一条路走向自己的幸福

0.00567083997126 我奇怪这么小的孩子怎么一个人跑来这园子里

0.00565208315006 我那时脾气坏到极点，经常是发了疯一样地离开家，从那园子里回来又中了魔似的什么话都不说

0.00559372837107 如今我摇着车在这园子里慢慢走，常常有一种感觉，觉得我一个人跑出来已经玩得太久了

0.00527989619912 而且我想，他的母亲也比我的母亲运气好，他的母亲没有一个双腿残废的儿子，否则事情就不这么简单

0.00527906358787 年年月月我都到这园子里来，年年月月我都要想，母亲盼望我找到的那条路到底是什么

0.00519622569726 那天你又说你不如死了好，你的一个朋友劝你：你不能死，你还得写呢，还有好多好作品等着你去写呢

0.00519145626625 他的衣着过分随便，走路的姿态也不慎重，走上五六十米路便选定一处地方，一只脚踏在石凳上或土埂上或树墩上，解下腰间的酒瓶，解酒瓶的当儿

0.00507970004724 她一个人在园子里走，走过我的身旁，走过我经常呆的一些地方，步履茫然又急迫

0.00497335014554 "在那段日子里--那是好几年长的一段日子，我想我一定使母亲作过了最坏的准备了，但她从来没有对我说过："你为我想想"

0.0049360646412 我便又不能在家里呆了，又整天整天独自跑到地坛去，心里是没头没尾的沉郁和哀怨，走遍整个园子却怎么也想不通：母亲为什么就不能再多活两年

0.00491815078362 是中了魔了，我走到哪儿想到哪儿，在人山人海里只寻找小说，要是有一种小说试剂就好了，见人就滴两滴看他是不是一篇小说，要是有一种小说显

0.00490464531034 我在这园子里坐着，我听见园神告诉我，每一个有激情的演员都难免是一个人质

0.00486932833768 十五年前的一个下午，我摇着轮椅进入园中，它为一个失魂落魄的人把一切都准备好了

0.00483241065578 有一天我在这园子碰见一个老太太，她说："哟，你还在这儿哪

0.0047216969869 我才想到，当年我总是独自跑到地坛去，曾经给母亲出了一个怎样的难题

0.00470900507196 我带着本子和笔，到园中找一个最不为人打扰的角落，偷偷地写

LDA总结

- 由于在词和文档之间加入的主题的概念，可以较好的解决一词多义和多词一义的问题。
- 在实践中发现，LDA用于短文档往往效果不明显——这是可以解释的：因为一个词被分配给某个主题的次数和一个主题包括的词数目尚未敛。往往需要通过其他方案“连接”成长文档。
  - 用户评论/Twitter/微博
- LDA可以和其他算法相结合。首先使用LDA将长度为Ni的文档降维到K维(主题的数目)，同时给出每个主题的概率(主题分布)，从而可以使用if-idf继续分析或者直接作为文档的特征进入聚类或者标签传播算法——用于社区发现等问题。

参考文献

- David M. Blei, Andrew Y. Ng, Michael I. Jordan, *Latent Dirichlet Allocation*, 2003
- Gregor Heinrich, *Parameter estimation for text analysis*. 2008
- Matthew D. Hoffman, David M. Blei, Francis Bach. *Online learning for Latent Dirichlet Allocation*. 2010
- Mihalcea R, Tarau P. TextRank, *Bringing order into texts*. Association for Computational Linguistics, 2004.
- http://en.wikipedia.org/wiki/Dirichlet_distribution
- http://en.wikipedia.org/wiki/Conjugate_prior
- https://en.wikipedia.org/wiki/Perplexity
- https://github.com/letiantian/TextRank4ZH

我们在这里

- http://wenda.ChinaHadoop.cn
  - 视频/课程/社区
- 微博
  - @ChinaHadoop
  - @邹博_机器学习
- 微信公众号
  - 小象学院
  - 大数据分析挖掘

感谢大家！

恳请大家批评指正！