https://blog.csdn.net/eastmount/article/details/50824215

import numpy as np

导入numpy并取别名为np

Numpy是一个科学计数库，提供矩阵计算功能

X = lda.datasets.load\_reuters() 加载语料库，每一行为一篇文章 ，统计每一篇文章的单词出现得次数（词频），x是一个(a\*b)的矩阵，a代表文章数目，b代表a篇文章中不同单词的个数

.format

>>>"{} {}".format("hello", "world") # 不设置指定位置，按默认顺序

'hello world'

>>> "{0} {1}".format("hello", "world") # 设置指定位置

' hello world'

>>> "{1} {0} {1}".format("hello", "world") # 设置指定位置

'world hello world'

type()

一个参数：返回元素的类型

该函数认为子类和父类不是同一个类型，与instance有区别

X[:5,:5]

假设x是一个二维矩阵，那么这段代码的意思是取前5行和前5列

vocab=lda.datasets.load\_reuters\_vocab();

返回单词列表

len

返回列表元素的个数

range 返回一个整数列表

range(5):[0,1,2,3,4,5]

sum求和函数，sum(1,2,3) 得到6

enumerate()

>>>seasons = ['Spring', 'Summer', 'Fall', 'Winter']

>>> list(enumerate(seasons))

[(0, 'Spring'), (1, 'Summer'), (2, 'Fall'), (3, 'Winter')]

tuple 不可变元组，使用时可以把它转换成其他类型

[:-5:-1] 从倒数第一到倒数第5（不包括），每格（n-1）个元素取一个，此时n为1

[:-5:1] 从第一到倒数第5（不包括），每格（n-1）个元素取一个，此时n为1

np.argsort

如果参数是一维数组，从小到大返回元素的坐标

Join

将序列的元素加上指定字符后连接成一个字符窜

**import** numpy **as** np  
**import** lda  
**import** lda.datasets  
  
x=lda.datasets.load\_reuters(); x是一个395\*4258的矩阵，395代表文章的数目，4258代表不同单词的个数，每一行代表一篇文章，x[i][j]表示第i篇文章中第j个单词的数目  
print(**"type(x):{}"**.format(type(x)))  
print(**"shape:{}"**.format(x.shape))  
print(x[:5,:5]);  
  
vocab=lda.datasets.load\_reuters\_vocab();vocab存储着不同单词，可以理解为词料库  
print(**"type(vocab):{}"**.format(type(vocab)))  
print(**"len(vocab):{}"**.format(len(vocab)))  
print(vocab[:5])  
  
titles=lda.datasets.load\_reuters\_titles();title存储着不同文章的题目  
print(**"type(titles):{}"**.format(type(titles)))  
print(**"len(titles):{}"**.format(len(titles)))  
print(titles[:5])  
  
doc\_id=0  
world\_id=2017  
print(**"doc\_id:{} world\_id:{}"**.format(doc\_id,world\_id))  
print(**"-----count:{}"**.format(x[doc\_id,world\_id]))  
print(**"-----world:{}"**.format(vocab[world\_id]))  
print(**"-----title:{}"**.format(titles[doc\_id]))  
  
model=lda.LDA(n\_topics=20,n\_iter=500,random\_state=1)设置LDA模型要生成的主题数，迭代次数  
model.fit\_transform(x)生成模型  
  
topic\_word=model.topic\_word\_获得主题-词分布，是一个20\*4258的矩阵，a[i][j]代表第i个主题下第j个词的概率  
print(**"type(topic\_word):{}"**.format(type(topic\_word)))  
print(**"topic\_word.shape:{}"**.format(topic\_word.shape))  
print(vocab[:3])  
print(topic\_word[:,:3])  
  
**for** n **in** range(5):  
 sum\_pr=sum(topic\_word[n,:])  
 print(**"topic:{} sum\_pr:{}"**.format(n,sum\_pr))  
  
n=5  
**for** i,topic\_list **in** enumerate(topic\_word):  
 topic\_words=np.array(vocab)[np.argsort(topic\_list)][:-(n+1):-1]  
 print(**"topic {}:\n -{}"**.format(i,topic\_words))  
  
  
doc\_topic=model.doc\_topic\_获得文章-主题分布，是一个395\*20的矩阵，不同文章中主题的概率是不一样的  
**for** i **in** range(10):  
 topic\_pr=doc\_topic[i].argmax();获得概率最大的主题  
 print(**"len(doc\_topic[i]):{}"**.format(len(doc\_topic[i])))  
 print(**"doc:{} topic:{}"**.format(i,topic\_pr))

LDA算法：不同文章有不同的主题分布，不同主题有不同的词语分布。先从主题分布中选取一个主题分布，确定词语分布，按照词语分布的概率从词料库中选取词语，再把这些组成一篇文章