潜在语义分析（LSA）的原理讲解以及python实现

在传统的文本信息处理中，以单词向量表示文本的语义内容，以单词向量空间的度量来表示文本之间的语义近似度。这种方法不能准确表示语义。
潜在语义分析试图从大量的文本数据中发现潜在的话题，以话题向量来表示文本的语义内容，以话题向量的空间度量更准确地表示文本之间的语义相似度。
潜在语义分析使用的是非概率的话题分析模型，具体来说，就是将文本集合表示为单词-文本矩阵，对单词-文本矩阵进行奇异值分解，从而得到话题向量空间，以及文本在话题向量空间的表示。可采用的矩阵分解方法有：**奇异值分解、非负矩阵分解。**
给定一个含有nnn个文本的集合D={d1,d2,⋯ ,dn}D=\{d_1,d_2,\cdots,d_n\}D={d1,d2,⋯,dn}，以及在所有文本中出现的mmm个单词W={w1,w2,⋯ ,wm}W=\{w_1,w_2,\cdots,w_m\}W={w1,w2,⋯,wm}，则将单词在文本中出现的数据用一个单词-文本表示，记作XXX。
X=[xij]m×nX = [x_{ij}]_{m\times n}X=[xij]m×n
其中，元素xijx_{ij}xij表示单词wiw_iwi在文本djd_jdj中出现的频数或权值。该矩阵是一个稀疏矩阵。
权值通常用单词频率-逆文本频率（TF-IDF）表示，其定义是：
TFIDFij=tfijtf·jlogdfdfi,i=1,2,⋯ ,m;j=1,2,⋯ ,nTFIDF_{ij}=\frac{tf_{ij}}{tf_{\cdot j}}\log\frac{df}{df_i},i=1,2,\cdots,m;j=1,2,\cdots,nTFIDFij=tf·jtfijlogdfidf,i=1,2,⋯,m;j=1,2,⋯,n
式中tfijtf_{ij}tfij是单词wiw_iwi出现在文本djd_jdj中的频数，tf·jtf_{\cdot j}tf·j是文本djd_jdj中出现的所有单词的频数之和，dfidf_idfi是含有单词mim_imi的文本数，dfdfdf是文本集合DDD的全部文本数。直观的，一个单词在一个文本中出现的次数越高，这个单词在这个文本中的重要度就越高；一个单词在整个文本集合中出现的文本越少，这个单词就越能表示其所在文本的特点，重要程度就越高。
单词向量空间模型直接使用单词-文本矩阵信息。单词文本矩阵的第jjj列向量xjx_jxj表示文本djd_jdj:
xj=[x1j,x2j,⋯ ,xmj]T,j=1,2,⋯ ,nx_j=[x_{1j},x_{2j},\cdots,x_{mj}]^T,j=1,2,\cdots,nxj=[x1j,x2j,⋯,xmj]T,j=1,2,⋯,n
其中xijx_{ij}xij是单词wiw_iwi在文本djd_jdj的权值，i=1,2,⋯ ,mi=1,2,\cdots,mi=1,2,⋯,m，权值越大，该单词在该文本中的重要度就越高。
之后就可以采用两个单词向量的内积或标准化内积来表示对应的文本之间的语义相似度。did_idi与djd_jdj之间的相似度为：
xi·xj, xi·xj||xi||xj|x_i\cdot x_j,\,\,\,\,\frac{x_i\cdot x_j}{||x_i||\,||x_j||}xi·xj,||xi|||xj|xi·xj
这种简单的单词向量空间模型不能处理一词多义的问题，也无法处理同义的问题，需要更加精确的方法。

话题向量空间

假设所有文本共含有kkk个话题，假设每个话题由一个定义在单词集合WWW上的mmm为向量表示，称为话题向量，即：
tl=[t1l,t2l,⋯ ,tml]T,l=1,2,⋯ ,kt_l=[t_{1l},t_{2l},\cdots,t_{ml}]^T,l=1,2,\cdots,ktl=[t1l,t2l,⋯,tml]T,l=1,2,⋯,k
其中tilt_{il}til是单词wiw_iwi在话题tlt_ltl的权值，i=1,2,⋯ ,mi=1,2,\cdots,mi=1,2,⋯,m，权值越大，该单词在该话题中的重要度就越高。这kkk个话题向量t1,t2,⋯ ,tkt_1,t_2,\cdots,t_kt1,t2,⋯,tk张成一个话题向量空间，维数为kkk。

现在考虑将文本集合DDD中的文本djd_jdj，在单词向量空间中由一个向量xjx_jxj表示，将xjx_jxj投影到话题向量空间TTT，得到在话题向量空间的一个向量yjy_jyj，yjy_jyj是一个kkk维向量，其表达式为：
yj=[y1j,y2j,⋯ ,ykj],j=1,2,⋯ ,ny_j=[y_{1j},y_{2j},\cdots,y_{kj}],j=1,2,\cdots,nyj=[y1j,y2j,⋯,ykj],j=1,2,⋯,n
其中yljy_{lj}ylj是话题tlt_ltl在文本djd_jdj的权值，权值越大，该话题在该文本中的重要度就越高。
矩阵YYY表示话题在文本中出现的情况，称为话题-文本矩阵。
Y=[yij]k×nY=[y_{ij}]_{k\times n}Y=[yij]k×n

从单词向量空间到话题向量空间的线性变换
这样一来，在单词向量空间的文本向量xjx_jxj可以通过它在话题空间的向量yjy_jyj近似表示，具体的由kkk个话题向量yjy_jyj为系数的线性组合近似表示。
xj≈y1jt1+y2jt2+⋯+ykjtk,j=1,2,⋯ ,nx_j\approx y_{1j}t_1+y_{2j}t_2+\cdots+y_{kj}t_k, j=1,2,\cdots,nxj≈y1jt1+y2jt2+⋯+ykjtk,j=1,2,⋯,n
所以，单词文本矩阵XXX可以近似的表示为单词-话题矩阵与话题-文本矩阵YYY的乘积形式。这就是潜在语义分析。
X≈TYX\approx TYX≈TY
要尽心潜在语义分析，需要同时决定两部分内容：话题向量T与文本在话题空间的表示Y，使两者的乘积近似等于原始矩阵。这些完全从话题-文本矩阵的信息中获得。

潜在语义分析算法

潜在语义分析利用矩阵奇异值分解，具体的，对单词-文本矩阵进行奇异值分解，将其左矩阵作为话题向量空间，将其对角矩阵与右矩阵的乘积作为文本在话题向量空间的表示。
算法：
1、给定文本集合和单词集合，首先构造一个文本-单词矩阵，矩阵中的每个元素表示单词在文本中出现的频数或权值。
2、截断奇异值分解
根据给定的话题数目k进行阶段奇异值分解：
X≈UkΣkVkX\approx U_k\Sigma_kV_kX≈UkΣkVk
3、话题向量空间
矩阵UkU_kUk的每一列向量表示一个话题，称为话题向量。由这k话题张成一个子空间：
Uk=[u1 u2 ⋯ uk]U_k=[u_1\,\,\,u_2\,\,\,\cdots\,\,\,u_k]Uk=[u1u2⋯uk] 4、文本的话题向量表示
矩阵(ΣkVkT)(\Sigma_kV_k^T)(ΣkVkT)的每一个列向量是一个文本在话题向量空间的表示。
用代码表示：

```
import numpy as np
import pandas as pd
import jieba
```

```
df = pd.read_excel(r'D:\BaiduNetdiskDownload\最终data.xlsx')
data = np.array(df['abstract'])
dic = [] for i in data:
cutdata = jieba.cut(i)
for j in cutdata:
if j not in dic and len(j)>=2:
dic.append(j)
a = len(dic)
X = np.zeros((a, len(data)))

for i in range(len(data)):
cutdata = jieba.cut(data[i])
p = [] for j in cutdata:
p.append(j)
for k in range(len(p)):
if p[k] in dic:
index = dic.index(p[k])
X[index][i] += 1

U, O, V = np.linalg.svd(X)

print(U[:, :3])
a = np.diag(O[:3])
b = V[:3, :] c = np.dot(a, b)
print(c.shape)
```

最终输出结果就是话题向量空间与文本在话题空间的线性表示。

非负矩阵分解算法

另外一种对单词文本矩阵分解的方法是非负矩阵分解。
若XXX是非负的，记作X⩾0X\geqslant0X⩾0，则分别找到两个非负矩阵W⩾0W\geqslant0W⩾0和H⩾0H\geqslant0H⩾0，使得X≈WHX\approx WHX≈WH
非负矩阵分解是为了用较少的基向量、系数向量来表示较大的数据矩阵。
我们采用最优化问题求解的方式来求W,HW,HW,H
首先定义损失函数
1、平方损失
||A−B||=∑i,j(aij−bij)2||A-B||=\sum_{i,j}(a_{ij}-b_{ij})^2||A−B||=i,j∑(aij−bij)2
2、散度
D(A||B)=∑i,j(aijlogaijbij−aij+bij)D(A||B)=\sum_{i,j}(a_{ij}\log\frac{a_{ij}}{b_{ij}}-a_{ij}+b_{ij})D(A||B)=i,j∑(aijlogbijaij−aij+bij)
因此，最优化问题转变为：
minW,H ||X−WH||2\min_{W,H}\,\,\,\,\,\,||X-WH||^2W,Hmin||X−WH||2
s.t.W,H⩾0W,H\geqslant0W,H⩾0
或者为：
minW,H D(X||WH)min_{W,H}\,\,\,\,\,\,D(X||WH)minW,HD(X||WH)
s.t.W,H⩾0W,H\geqslant0W,H⩾0

算法实现

定理：
平方损失||X−WH||2||X-WH||^2||X−WH||2对下列乘法更新规则
Hlj←Hlj(WTX)lj(WHHT)ilH_{lj}\leftarrow H_{lj}\frac{(W^TX)_{lj}}{(WHH^T)_{il}}Hlj←Hlj(WHHT)il(WTX)lj
Wil←Wil(XHT)il(WHHT)ilW_{il}\leftarrow W_{il}\frac{(XH^T)_{il}}{(WHH^T)_{il}}Wil←Wil(WHHT)il(XHT)il
是**非增**的，当且仅当WWW和HHH是平方损失函数的稳定点时函数的更新不变。

散度损失D(X−WH)D(X-WH)D(X−WH)对下列乘法更新规则
Hlj←Hlj∑i[WilXij/(WH)ij]∑iWilH_{lj}\leftarrow H_{lj}\frac{\sum_i[W_{il}X_{ij}/(WH)_{ij}]}{\sum_iW_{il}}Hlj←Hlj∑iWil∑i[WilXij/(WH)ij]
Wil←Wil∑j[HljXij/(WH)ij]∑jHljW_{il}\leftarrow W_{il}\frac{\sum_j[H_{lj}X_{ij}/(WH)_{ij}]}{\sum_jH_{lj}}Wil←Wil∑jHlj∑j[HljXij/(WH)ij]
是**非增**的，当且仅当WWW和HHH是散度损失函数的稳定点时函数的更新不变。
其实这种乘法更新是将梯度下降法的步长取特殊值，从而使收敛速率加快！
代码如下：

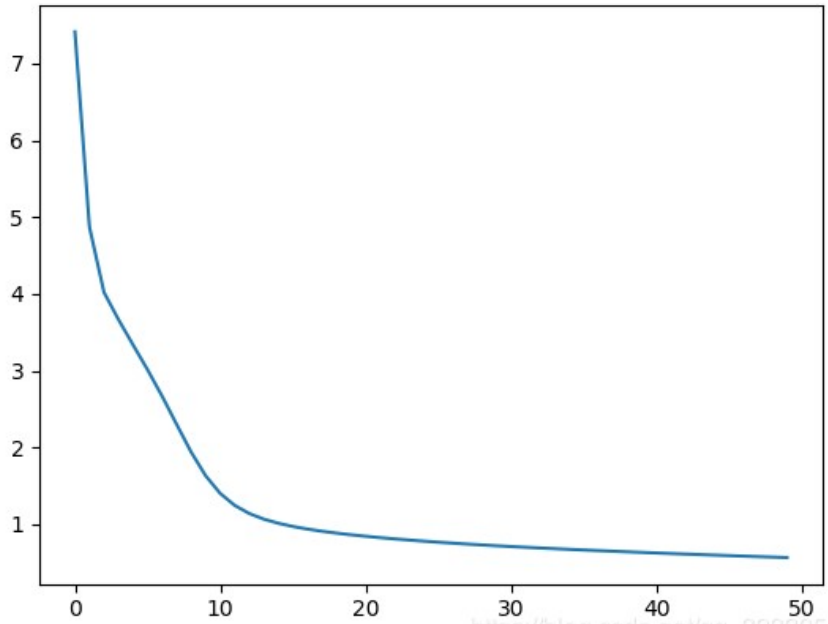
```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed()
X = np.random.randint(0, 3, (5, 4))

class LSA:
def __init__(self, x, k): # k为话题个数
```

```
self.k = k
self.X = x
self.m = x.shape[0] self.n = x.shape[1] self.W = np.random.uniform(0, 1, (self.m, k))
self.H = np.random.uniform(0, 1, (k, self.n))

def standard(self):
t = self.W**2
T = np.sqrt(np.sum(t, axis=0))
for i in range(self.W.shape[0]):
self.W[i] = self.W[i]/T

def update(self):
up1 = np.dot(self.X, self.H.T)
t1 = np.dot(self.W, self.H)
down1 = np.dot(t1, self.H.T)
up2 = np.dot(self.W.T, self.X)
t2 = np.dot(self.W.T, self.W)
down2 = np.dot(t2, self.H)
for i in range(self.m):
for l in range(self.k):
self.W[i][l] = self.W[i][l]*(up1[i][l]/down1[i][l])
for l in range(self.k):
for j in range(self.n):
self.H[l][j] = self.H[l][j]*(up2[l][j]/down2[l][j])

def cost(self):
X = np.dot(self.W, self.H)
S = (self.X - X)**2
score = np.sum(S)
return score

L = LSA(X, 3)
x = [] score = [] for i in range(50):
L.standard()
L.update()
x.append(i)
score.append(L.cost())
print(X)
print(np.dot(L.W, L.H))
plt.scatter(x, score)
plt.show()
```

最终的运行结果为：

```
[[2 0 1 2]
 [2 2 0 0]
 [0 0 2 0]
 [1 2 0 0]
 [1 2 0 1]]
[[1.78520402e+00 2.25009780e-01 1.48614179e+00 1.77822652e+00]
 [2.01112382e+00 1.98756494e+00 2.52594086e-03 1.06065855e-01]
 [5.86539227e-01 5.91590604e-15 5.76135242e-01 6.88493513e-01]
 [1.05609590e+00 1.92703754e+00 3.56676361e-09 2.86944842e-01]
 [9.62883741e-01 2.05593865e+00 3.06548643e-01 7.72345887e-01]]
```


原始矩阵与近似矩阵，可见差异已经相当小。
平方损失的图像：



在迭代到第50步时已经收敛。

作者：xjtu_rzc