潜在语义分析

1 潜在语义分析

1.1 单词向量空间

给定一个含有 n 个文本的集合 $D = \{d_1, d_2, ..., d_n\}$ 以及在所有文本中出现的 m 个单词的集合 $W = \{w_1, w_2, ..., w_m\}$。则单词在文本中出现的数据可以用一个**单词-文本矩阵**表示，记作 $X = [x_{ij}]_{m \times n}$。元素 x_{ij} 表示单词 w_i 在文本 d_j 中出现的频数或权值。单词-文本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，其定义为：

$$TFIDF_{ij} = TF_{ij}IDF_{ij} = \frac{tf_{ij}}{tf_{\cdot j}} \log \frac{df}{df_i}$$

其中 tf_{ij} 是单词 w_i 在文本 d_j 中的频数，$tf_{\cdot j}$ 是文本 d_j 中所有单词的频数之和，df_i 是含有单词 w_i 的文本数，df 是文本 D 的全部文本数。因此，TF-IDF 是两种重要度的积，表示综合重要度。

在判断文本相似度时，对于文本 d_i 和 d_j 可以直接使用单词-文本矩阵的第 i 列向量 x_i 和第 j 列向量 x_j 的内积 $x_i \cdot x_j$ 或余弦夹角 $\frac{x_i \cdot x_j}{||x_i|| ||x_j||}$ 进行判断。该模型简单实用，但不能很好地处理一词多义性和多词一义性。

1.2 话题向量空间

为处理一词多义性和多词一义性，可以引入话题向量空间。话题可以由若干个语义相关的单词表示。假设所有文本共有 k 个话题，假设每个话题由一个定义在单词集合 W 上的 m 维向量表示，称为话题向量，即 $t_l = (t_{1l}, t_{2l}, ..., t_{ml})^T, l = 1, 2, ..., k$。其中，$t_{il}$ 是单词 w_i 在话题 t_l 上的权值。k 个话题向量构成一个话题向量空间 T。话题向量空间也可以由**单词-话题矩阵**表示，记作 $T = [t_{ij}]_{m \times k}$。

对于文本集合 D 中的文本 d_j，其在单词向量空间中的表示为 x_j，将其投影到话题向量空间 T 中，可以得到在话题向量空间中的表示 $y_j = (y_{1j}, y_{2j}, ..., y_{kj})^T$。其中 y_{ij} 是文本 d_j 在话题 t_l 上的权值。记 $Y = [y_1, y_2, ..., y_n]$ 为**话题-文本矩阵**。

考虑使用线性变换将在单词向量空间中的文本向量 x_j 通过它在话题空间中的向量 y_j 近似表示，即将 k 个话题向量以 y_j 为系数进行线性组合近似表示。

$$x_j \approx y_{1j}t_1 + y_{2j}t_2 + ... + y_{kj}t_k, \quad j = 1, 2, ..., n$$

所以，**单词-文本矩阵 X 可以近似地表示为单词-话题矩阵 T 和话题-文本矩阵 Y 的乘积**，即 $X \approx TY$。其中仅有单词-文本矩阵 X 是可以观测的。潜在语义分析就是通过可观测的 X，根据数据内在的关联信息，同时决定话题空间 T 和文本在话题空间的表示 Y。

1.3 矩阵奇异值分解法

矩阵奇异值分解法使用截断奇异值分解对单词-文本矩阵 X 进行分解。

$$X \approx U_k \Sigma_k V_k^T$$

式中 X 是 $m \times n$ 的单词-文本矩阵。$k \leq n \leq m$，U_k 是 $m \times k$ 矩阵，它的列由 X 的前 k 个互相正交的左奇异向量组成。Σ_k 是 k 阶对角方阵，对角元素是 X 的前 k 个最大的奇异值。V_k 是 $n \times k$ 矩阵，它的列由 X 的前 k 个互相正交的右奇异向量组成。

矩阵奇异值分解法令

$$X \approx U_k \Sigma_k V_k^T = U_k(\Sigma_k V_k^T) := TY$$

即将 U_k 作为话题空间，其每一列表示一个话题；$\Sigma_k V_k^T$ 作为文本在话题向量空间中的表示，其第 j 列 $(\Sigma_k V_k^T)_j$ 表示文本 d_j 在话题向量空间中的表示。矩阵奇异值方法分解法步骤如下：

输入：单词-文本矩阵 X 和话题个数 $k \leq n \leq m$。

1. 对 X 进行截断奇异值分解 $X \approx U_k \Sigma_k V_k^T$
2. 计算话题空间 $T = U_k$ 和文本在话题向量空间的表示 $Y = \Sigma_k V_k^T$

输出：话题空间 T 和文本在话题向量空间的表示 Y。

使用矩阵奇异值分解法计算本文相似度时，可直接用 $(\Sigma_k V_k^T)_j$ 进行内积或余弦夹角的计算。

1.4 非负矩阵分解法

对于一个非负矩阵（矩阵中的所有元素非负）$X \geq 0$，可以找到两个非负矩阵 $W \geq 0$ 和 $H \geq 0$，使得 $X \approx WH$。X 是一个 $m \times n$，假设 $k < \min(m, n)$，非负矩阵 W, H 分别为 $m \times k, k \times n$ 的矩阵。

使用非负矩阵分解法可以找到单词-文本矩阵的另一种分解方式。即 W 为话题向量空间，H 是文本在话题向量空间中的表示。

非负矩阵分解法等价于寻找 W, H，而这又可以转化为一个优化问题，可以分别使用平方损失和散度损失来刻画这一问题。平方损失的优化问题如下：

$$\min_{W,H} \quad ||X-WH||^2 = \sum_{i,j}(X_{ij}-(WH)_{ij})^2$$
$$s.t. \quad W,H \geq 0$$

散度损失的优化问题如下：

$$\min_{W,H} \quad D(X||WH) = \sum_{i,j}(X_{ij}\log\frac{X_{ij}}{(WH)_{ij}} - X_{ij} + (WH)_{ij})$$
$$s.t. \quad W,H \geq 0$$

两个问题可分别通过以下参数更新方法求得参数。对于平方损失 $||X-WH||^2$，更新方法为

$$H_{lj} := H_{lj}\frac{(W^TX)_{lj}}{(W^TWH)_{lj}}$$

$$W_{il} := W_{il}\frac{(XH^T)_{il}}{(WHH^T)_{il}}$$

对于散度损失，更新方法为：

$$H_{lj} := H_{lj}\frac{\sum_i[W_{il}X_{ij}/(WH)_{ij}]}{\sum_i W_{il}}$$
$$W_{il} := W_{il}\frac{\sum_j H_{lj}X_{ij}/(WH)_{ij}}{\sum_j H_{lj}}$$

以上两个更新方法都是梯度下降法的特殊形式，且可以证明，当且仅当 W,H 是损失函数的稳定点时，函数的更新不变。非负矩阵分解法步骤如下：

输入：单词-文本矩阵 $X \geq 0$ 和话题个数 k。

1. 初始化：$W,H \geq 0$，并对 W 的每一列数据归一化
2. 迭代直至收敛：

- 更新 W 的元素，从 $l = 1,2,...,k; i = 1,2,...,m$ 依次更新 W_{il}
- 更新 H 的元素，从 $l = 1,2,...,k; i = 1,2,...,n$ 依次更新 H_{lj}

输出：话题空间 W 和文本在话题向量空间的表示 H。

2 代码实现

此处的数据为：

$$X = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 3 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 2 & 1 \end{bmatrix}$$

2.1 sklearn 实现

准备数据：

```
import numpy as np
X = np.array([[2,0,0,0],[0,2,0,0],[0,0,1,0],[0,0,2,3],[0,0,0,1],[1,2,2,1]])
```

潜在语义分析的 sklearn 接口位于 sklearn.decomposition.TruncatedSVD, 其文档可见**此处**。注意该接口实际就是截断奇异值分解的接口，官方文档中也有显示 (Dimensionality reduction using truncated SVD (aka LSA).)。其中较为重要的参数有：

- n_components：话题个数

```
from sklearn.decomposition import TruncatedSVD
clf = TruncatedSVD(n_components=2)
clf.fit(X)
```

其话题向量空间为：

```
print(clf.fit_transform(X))
```

```
## [[ 0.3511592   0.78272254]
##  [ 0.7023184   1.56544509]
##  [ 0.63851545 -0.03795798]
##  [ 3.26283385 -1.52321299]
##  [ 0.66193431 -0.48243234]
##  [ 2.81686322  1.39845805]]
```

文本在话题空间的表示为：

```
print(clf.components_)
```

```
## [[ 0.1755796   0.3511592   0.63851545  0.66193431]
##  [ 0.39136127  0.78272254 -0.03795798 -0.48243234]]
```

2.2 矩阵奇异值分解算法

```
class LSA:

    def __init__(self,topic_num=2):
        self.topic_num = topic_num

    def fit(self,X):
        U,Sigma,V = np.linalg.svd(X)
        T,Y = U[:,:self.topic_num],(np.diag(Sigma).dot(V))[:self.topic_num,:]
        self.components = Y
        self.topic_vector = T

    def fit_transfrom(self,X):
        return self.topic_vector
```

其结果为：

```
clf = LSA(topic_num=2)
clf.fit(X)
```

```
print(clf.fit_transfrom(X))
```

```
## [[-0.07843687 -0.28442303]
##  [-0.15687373 -0.56884607]
##  [-0.14262235  0.01379304]
##  [-0.72880467  0.55349991]
##  [-0.14785332  0.17530461]
##  [-0.6291902  -0.50816689]]
```

```
print(clf.components)
```

```
## [[-0.78606393 -1.57212786 -2.85861209 -2.96345752]
##  [-1.07701296 -2.15402591  0.10445908  1.32763745]]
```

注意到这和 sklearn 输出的结果不完全一样，原因是：numpy 中的 svd 分解未对奇异向量组做正交化且矩阵的奇异值分解不唯一。