第五次作业

1 1. 推导概率潜在语义分析的共现模型的EM算法

潜在语义分析的共现模型定义如下:

因为共现模型假设在话题z给定的情况下，单词w与文本d是条件独立的，所以每个单词-文本对 (w,d) 的概率由以下公式决定:

$$P(w,d)=\sum_{z\in Z}P(z)P(w\mid z)P(d\mid z) \tag{7}$$

文本-单词共现数据 T 的生成概率为所有单词-文本对 (w,d) 的生成概率的乘积:

$$\begin{aligned} L=P(T)&=\prod_{(w,d)}P(w,d)^{n(w,d)}\\ &=\prod_{i=1}^{M}\prod_{j=1}^{N}P(w_i,d_j)^{n(w_i,d_j)} \end{aligned} \tag{8}$$

对似然函数取对数后得:

$$\begin{aligned} LL&=\sum_{i=1}^{M}\sum_{j=1}^{N}n(w_i,d_j)\log\left(P(w_i,d_j)\right)\\ &=\sum_{i=1}^{M}\sum_{j=1}^{N}n(w_i,d_j)\log\left(\sum_{k=1}^{K}P(z_k)\frac{P(w_i,d_j,z_k)}{P(z_k)}\right) \end{aligned} \tag{9}$$

其中$n(w_i,d_i)$表示(w_i,d_i)出现的次数。

根据Jesen不等式:

$$\begin{aligned} LL&=\sum_{i=1}^{M}\sum_{j=1}^{N}n(w_i,d_j)\log\left(\sum_{k=1}^{K}P(z_k)\frac{P(w_i,d_j,z_k)}{P(z_k)}\right)\\ &\geq\sum_{i=1}^{M}\sum_{j=1}^{N}n(w_i,d_j)\left(\sum_{k=1}^{K}P(z_k)\log\frac{P(w_i,d_j,z_k)}{P(z_k)}\right)\\ &=\sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=1}^{K}n(w_i,d_j)\left(P(z_k)\log\left(P(w_i,d_j,z_k)\right)-P(z_k)\log\left(P(z_k)\right)\right)\\ &\triangleq J(\theta,P(z)) \end{aligned} \tag{10}$$

E（expectation）步:

$$\begin{aligned} P^{(t)}(z_k)&=\textcolor{red}{\backslash\mathrm{argmax}}_{P(z_k)}J(w^{(t)},d^{(t)},P(z_k))\\ &=P(z_k\mid w_i^{(t)},d_j^{(t)})\\ &=\frac{P(w_i^{(t)},d_j^{(t)},z_k)}{P(w_i^{(t)},d_j^{(t)})}\\ &=\frac{P(z_k)P(w_i^{(t)}|z_k)P(d_j^{(t)}|z_k)}{\sum_{k=1}^{K}P(z_k)P(w_i^{(t)}|z_k)P(d_j^{(t)}|z_k)} \end{aligned} \tag{11}$$

M（maximize）步:

$$\theta^{(t+1)} = \backslash\mathrm{argmax}_{\theta} J(\theta, Q^{(t)}(z)) \tag{12}$$

又因为参数满足如下约束条件：

$$\sum_{k=1}^{K} P(z_k) = 1$$
$$\sum_{i=1}^{M} P(w_i|z_k) = 1, k = 1, 2, \ldots, K$$
$$\sum_{j=1}^{N} P(d_j|z_k) = 1, k = 1, 2, \ldots, K$$

据此构建Lagrange函数，求解带有约束的优化问题，

$$\Lambda = J(\theta, P(z)) + \lambda\left(1 - \sum_{k=1}^{K} P(z_k)\right) + \sum_{k=1}^{K} \tau_k \left(1 - \sum_{i=1}^{M} P(w_i|z_k)\right) + \sum_{k=1}^{K} \rho_k \left(1 - \sum_{j=1}^{N} P(d_j|z_k)\right)$$

解得：

$$\begin{cases} P(z_k) = \dfrac{\sum_{i=1}^{M}\sum_{j=1}^{N} n(w_i, d_j) P(z_k|w_i, d_j)}{\sum_{i=1}^{M}\sum_{j=1}^{N} n(w_i, d_j)} \\ P(w_i|z_k) = \dfrac{\sum_{j=1}^{N} n(w_i, d_j) P(z_k|w_i, d_j)}{\sum_{i=1}^{M}\sum_{j=1}^{N} n(w_i, d_j) P(z_k|w_i, d_j)} \\ P(d_j|z_k) = \dfrac{\sum_{i=1}^{M} n(w_i, d_j) P(z_k|w_i, d_j)}{\sum_{i=1}^{M}\sum_{j=1}^{N} n(w_i, d_j) P(z_k|w_i, d_j)} \end{cases}$$

2 2. 新闻爬取

从交大新闻网主页新闻栏目爬取最新的100条新闻，编程实现概率潜在语义分析的生成模型或共现模型，并输出不同的话题数下各个话题的高频词

2.1 （1）抓取新闻

```python
from bs4 import BeautifulSoup
import requests
import pandas as pd
from urllib import parse


class XJTU_News():
    def __init__(self, url):
        self.current_url = url  # 主url可以和path拼接
        self.cookies = {"_ga": "GA1.3.1733503684.1647506450"}
        self.news_urls = []
        self.content = pd.DataFrame(
            columns=["title", "date", "content", "source", "writer"])

    def get_soup(self, url):
        response = requests.get(url, cookies=self.cookies)
```

```python
        response.encoding = 'UTF-8-SIG'
        soup = BeautifulSoup(response.text, "lxml")
        return soup

    def get_news_list(self, path):
        self.current_url = parse.urljoin(self.current_url, path)
        soup = self.get_soup(self.current_url)
        self.news_urls.extend([parse.urljoin(self.current_url, object["href"])
                               for object in soup.find_all("a", class_="bt")])
        next_page_path = soup.find(
            "span", class_="p_next p_fun").next_element["href"]
        return(next_page_path)

    def get_news_lists(self, number):
        next_page_path = ""
        while(len(self.news_urls) < number):
            next_page_path = self.get_news_list(next_page_path)

    def get_content(self):
        for url in self.news_urls:
            soup = self.get_soup(url)
            title = soup.title.string.split("-西安交通大学")[0]
            try:
                content = soup.find("div", id="vsb_content_2").text.strip()
            except:
                content = None # 有的新闻是视频，所以没有content正文
                print(url)
            writer = soup.find("div", class_="zdf clearfix").text.strip()
            source = None
            date = None
            for temp in soup.find("div", class_="shffffff").contents:
                if "来源" in temp.text:
                    source = temp.text.split("：")[-1].strip()
                elif "日期" in temp.text:
                    date = temp.text.split("：")[-1].strip()
                else:
                    continue
            self.content = self.content.append(
                {"title": title, "date": date, "content": content, "source": source, "writer": writer},ignore_index=True)
```

```python
main = XJTU_News(url="http://news.xjtu.edu.cn/zyxw.htm")
main.get_news_lists(110)
main.get_content()
main.content.to_csv("result.csv",index = None)
```

2.2 （2）分词及数据预处理

```python
import jieba
import pandas as pd
data = pd.read_csv("result.csv")
data["text"] = data["title"]+data["content"]
```

```python
# 中文停用词表：https://github.com/goto456/stopwords
stopwords = []
f = open("cn_stopwords.txt", "r",encoding='utf-8')
line = f.readline() # 读取第一行
with open("cn_stopwords.txt", "r",encoding='utf-8') as f:
    line = f.readline()
    while line:
        stopwords.append(line[:-1]) # 列表增加
        line = f.readline()
```

```python
data = data[data["content"].notna()][:100] # 删除正文为空的数据
words=[]
for i in range(data.shape[0]):
    news = ' '.join(jieba.cut(data.iloc[i]["content"]))
    words.append(news)
```

```python
from sklearn.feature_extraction.text import CountVectorizer
# construct co-occurance matrix
count_model = CountVectorizer(max_features=2000,max_df=0.5,stop_words=stopwords)
word_vector = count_model.fit_transform(words).todense().T      # co-occurance matrix
word_vector.shape
```

```
(2000, 100)
```

2.3 （3）潜在语义分析————共现模型

```python
import numpy as np

class pLSA():
    def __init__(self,step,topic_n,word_vector):
        self.step = step #最大步数
        self.K = topic_n #话题数量
        self.words = word_vector #词向量
        self.M,self.N = word_vector.shape #M是词向量长度 ,N是文本数
        self.p_w_z = np.random.rand(self.K, self.M)   # p(w|z)
        self.p_z_d = np.random.rand(self.N, self.K)   # p(z|d)
        self.p_z_wd = np.zeros((self.N, self.M, self.K))   # p(z|w,d)
        '''
        References
        ----------
        [1] "Bayesian Reasoning and Machine Learning", David Barber (Cambridge
        Press, 2012).
        [2] plsa.PyPI https://github.com/yedivanseven/PLSA
        '''
    def E_step(self):
        for j in range(self.N):
            for i in range(self.M):
```

```python
                temp = np.zeros((self.K))
                for k in range(self.K):
                    temp[k] = self.p_w_z[k, i] * self.p_z_d[j, k]
                self.p_z_wd[j,i] = temp / np.sum(temp)
    def M_step(self):
        ## p(w|z)
        for k in range(self.K):
            temp = np.zeros((self.M))
            for i in range(self.M):
                for j in range(self.N):
                    temp[i] += word_vector[i, j] * self.p_z_wd[j, i, k]
            self.p_w_z[k] = temp / np.sum(temp)

        ## p(z|d)
        for j in range(self.N):
            for k in range(self.K):
                temp = 0
                for i in range(self.M):
                    temp += word_vector[i, j] * self.p_z_wd[j, i, k]
                self.p_z_d[j, k] = temp / np.sum(word_vector[[j]])

    def fit(self):
        for _ in range(self.step):
            self.E_step()
            self.M_step()
        return self.p_w_z, self.p_z_d

```

```python
topic_n = 3
model = pLSA(step = 10,topic_n = topic_n,word_vector = word_vector)
p_w_z, p_z_d = model.fit()
```

```python
dict_ = count_model.get_feature_names()
topic_words = []
for k in range(topic_n):
    topic_ = np.argsort(-p_w_z[k, :])[:10]
    topic_composition = {dict_[i]:p_w_z[k, i] for i in topic_}
    print("主题{k}: {topic_composition}\n".format(k = k+1,topic_composition = topic_composition))
    topic_words.append(topic_composition)
```

```
主题1：{'青年': 0.01862717415293134, '研究': 0.01543787285185595, '青春': 0.009702259875018139, '时代': 0.009210889165625185, '交大': 0.008298109863527114, '团队': 0.0065938244773482285, '科技': 0.006312667471596037, '表示': 0.006157305897420258, '共青团': 0.006076858885300087, '平台': 0.005269242534540203}

主题2：{'学生': 0.01779301337160576, '习近平': 0.0142531717358071, '总书记': 0.014039085573739788, '西迁': 0.013307092219758378, '培养': 0.013076798465364066, '教学': 0.010572538240392362, '课程': 0.009712016007890774, '时代': 0.008596336032387243, '青年': 0.0072026353658971995, '教育': 0.007145105821298227}

主题3：{'学生': 0.012331557568073697, '就业': 0.008983833242868363, '活动': 0.008219019650107387, '服务': 0.007425761075039543, '教育': 0.007236010927243649, '体育': 0.007005019234515966, '学科': 0.006794192250061121, '开展': 0.005732258068710147, '推进': 0.005671051550816049, '加强': 0.005637810487683335}
```

```
C:\Users\zjchenb139\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```