实验七 词向量和 LSA 模型

【实验目的】

1. 了解 Anaconda 和 python 的使用与设置。
2. 掌握词向量的基本原理和实现方法。
3. 掌握 LSA 模型的基本原理和方法

【实验学时】

建议 2 学时

【实验环境配置】

1、 Windows 环境
2、 Anaconda
3、 Pandas
4、 Genism

【实验原理】

算法的实现：

1、Pandas 数据导入
2、LSA 模型（https://blog.csdn.net/zhzhji440/article/details/47193731）

【实验步骤】

1、 打开 Anaconda 的 jupyter notebook，创建实验七，明确标题和步骤。

2、 从数据文件中，导入实验数据。

1. 导入数据

```
n [2]: import os

n [3]: file_path = 'corpus.txt'
       stopword_path = 'baidu_stopwords.txt'

n [4]: with open(file_path, encoding='utf-8') as f:
           ls_corpus = f.read().splitlines()

n [5]: ls_corpus[2]

Out[5]: '不满一岁的永康是个饱经病痛折磨的孩子,2011年7月5日出生的他,患有先天性心脏病、疝气,一出生便被遗弃。2012年1月8日,才5个月大的永康被发现呼吸困难,随后送往医院进行抢救治疗,病情稳定后于1月28日出院。\ue40c2012年2月13号,永康在思源焦点公益基金的帮助下在医院接受手术治疗,术后仅8天,永康突发右侧腹股沟斜疝嵌顿及肠梗阻,又再次进行抢救治疗,术后进重症监护室。3月7日,几经病痛折磨的永康终于康复出院,目前他的病情已经稳定。'

n [6]: with open(stopword_path, encoding='utf-8') as f:
           ls_stopword = f.read().splitlines()

n [7]: ls_stopword
```

3、 对导入的实验文档数据，利用 gensim 库完成文档的 LSA 模型的构建，计算文档相似度。

先对导入得数据中文进行 jieba 分词。

```
]: import jieba

]: ls_jieba = []
   for t in ls_corpus:
       i = jieba.lcut(t)
       ls_jieba.append(i)

   Building prefix dict from the default dictionary ...
   Dumping model to file cache C:\Users\ERICZH~1.ZHI\AppData\Local\Temp\jieba.cache
   Loading model cost 0.655 seconds.
   Prefix dict has been built successfully.

]: ls_jieba[0]
```

```
11]: ls_jieba = []
     for t in ls_corpus:
         i = jieba.lcut(t)
         ls_w = []
         for w in i:
             if w.isalpha()  and w not in ls_stopword:
                 ls_w.append(w)
         ls_jieba.append(ls_w)

12]: ls_jieba[0]
```

计算每个词语的 TF 值，Counter()可以方便、快速的计数，将 ls_jieba 分词里的词频数量进行统计，返回一个字典，键为元素，值为元素个数。然后遍历 dic_t 的 key 即中文词，根据公式计算 TF 值。

```
: from collections import Counter
```

```
: ls_tf = []
for ls_d in ls_jieba:
    dic_t = Counter(ls_d)
    dic_tf = {}
    for k in list(dic_t.keys()):
        dic_tf[k] = 1.0 + np.log(dic_t[k])
    ls_tf.append(dic_tf)
```

```
: ls_tf
```

计算词的 IDF 值

```
In [20]: ls_allwords = []
for d in ls_jieba:
    ls_allwords.extend(d)

dic_allwords = Counter(ls_allwords)

ls_allterms = list(dic_allwords.keys())

N = len(ls_jieba)
dic_idf = {}
for t in ls_allterms:
    for ls_d in ls_jieba:
        if t in ls_d:
            dic_idf[t] = dic_idf.get(t, 0)+1
    dic_idf[t] = np.log(N/dic_idf[t])
```

```
]: dic_idf
 '兔窝': 5.099866427824199,
 '会宁县': 5.099866427824199,
 '新': 2.209494669928034,
 '塬': 5.099866427824199,
 '榆中县': 5.099866427824199,
 '岘': 5.099866427824199,
 '临夏州': 5.099866427824199,
 '康乐县': 5.099866427824199,
 '八松': 5.099866427824199,
 '乡纳沟': 5.099866427824199,
 '捐助': 3.490428515390098,
 '介绍': 1.9643722118950486,
 '订做': 5.099866427824199,
 '约': 2.266653083767982,
 '需': 2.5349170703626616,
 '一个月': 3.713572066704308,
 '才能': 2.7019711550258276,
```

将 ls_jieba 复制一份到 ls_allwords 列表中，遍历 ls_allterms 即所有不重复的 key 值，再遍历 ls_jie，统计包含该词汇的字符串个数，否则为 0，反之加 1。最后利用 IDF=log(N.dic_idf[t]) 计算词汇的 idf 值。

计算 TF-IDF 值

遍历 dic_tf 通过其中的键值 key 与 dic_idf[key]计算 TF-IDF 值，等于 TF*IDF。再将 5 个 5 个数据组成一组，每一组中的 tfidf 值递减，加入到一个列表 ls_all 中。

```
ls_all = []
for dic_tf in ls_tf:
    dic_tfidf = {}
    for k in list(dic_tf.keys()):
        dic_tfidf[k] = dic_tf[k]*dic_idf[k]
    ls_tfidf = sorted(dic_tfidf.items(), key=lambda d: d[1], reverse=True) # 按照tfidf值从小到大排序
    ls_all.append(ls_tfidf[0:5])
```

```
[23]: ls_all

t[23]: [[('坐票', 12.169782499181526),
         ('座位', 12.169782499181526),
         ('设', 10.702642355997801),
         ('层次', 10.702642355997801),
         ('行李', 10.702642355997801)],
        [('干旱', 10.702642355997801),
         ('同心县', 5.099866427824199),
         ('核心区', 5.099866427824199),
         ('冬寒', 5.099866427824199),
         ('春暖迟', 5.099866427824199)],
        [('永康', 13.30778480511433),
         ('病痛', 8.634824463502863),
         ('抢救', 7.461224269024715),
         ('出院', 7.461224269024715),
         ('折磨', 6.774712164415943)],
        [('康师傅', 18.558706304232437),
         ('茶叶', 16.842742840792994),
         ('商行', 15.023748268553774),
         ('废料', 14.237600391676462),
```

利用 gensim 库完成文档的 LSA 模型的构建，首先基于 gensim 计算 tfidf 值

```
from gensim import corpora
from gensim import models
```

```
dict_allterms = corpora.Dictionary(ls_jieba)
corpus = list(dict_allterms.doc2bow(t) for t in ls_jieba)
tfidf_model = models.TfidfModel(corpus)
```

```
lsi = models.LsiModel(corpus, id2word=dict_allterms, num_topics=10)
```

```
query_vec = lsi[corpus]
```

```
for vec in query_vec:
    print(vec)
```

```
from gensim.similarities import MatrixSimilarity
```

```
sim = MatrixSimilarity(query_vec)
```

```
c=0
for i in sim:
    print(c)
    c +=1
    print(i)
```

```
 0.6768515  0.6339379 ]
1
[0.6500841  1.         0.6213866  0.6325331  0.58516246 0.86755335
 0.7282894  0.44135404 0.6289516  0.71896344 0.5581691  0.8551792
 0.45821348 0.6780881  0.6764243  0.36864895 0.5119069  0.5040775
 0.788567   0.6260694  0.7880432  0.60021985 0.5798384  0.6956307
 0.8581878  0.7463431  0.8847906  0.15134606 0.26456678 0.7492955
 0.7516763  0.62837696 0.70958275 0.20218386 0.7110268  0.8890736
 0.6010006  0.8428112  0.6504067  0.73437554 0.5324428  0.7119749
 0.25129592 0.60486364 0.5978676  0.25920412 0.21304573 0.7611144
 0.7270267  0.60996425 0.7961842  0.5032209  0.78582996 0.41321
 0.6957882  0.24815881 0.5032209  0.6908061  0.64631474 0.80117744
 0.77083075 0.6500528  0.22433865 0.80011475 0.80999    0.61956495
 0.6342307  0.80102384 0.85540044 0.7059687  0.51505065 0.82763016
 0.06351744 0.82042295 0.24681559 0.748902   0.41684824 0.7525088
 0.66781425 0.63437325 0.6802786  0.6957882  0.7690987  0.7661769
 0.8299645  0.64631474 0.592406   0.7398433  0.84634846 0.7301375
 0.5596947  0.76695514 0.7234677  0.7234081  0.7214709  0.63473475
 0.8655587  0.56157744 0.8014811  0.44943935 0.84975284 0.7222502
```

计算文本相似度

```
arr_sim = sim[query_vec]
```

```
t = 0.9
k = 0
len_s = len(arr_sim[k])
arr_i = np.zeros(len_s)
for i in range(len_s):
    if arr_sim[k,i] > t:
        arr_i[i] = 1
        print(i)
print(arr_i)
```

```
0
43
87
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

其中相似矩阵中为 1 的代表相似度大于 0.9 的，只有两个，大部分都低于 0.9。

【思考题】

1. 查阅文献，思考 TF-IDF 模型和 LSA 模型的差别？

TF-IDF 是根据词频，即某个词出现的次数，以及逆文档频率来计算其 TF-IDF 值的，而 LSA 是将词和文档映射到潜在语义空间，再比较其相似性，后者更复杂一些。