实验六 关键字提取

【实验目的】

1. 了解 Anaconda 和 python 的使用与设置。
2. 掌握关键字提取的基本原理和实现方法。
3. 掌握常用关键字提取的基本原理和方法

【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、TF-IDF 模型

【实验步骤】

1. 打开 Anaconda 的 jupyter notebook，创建实验六，明确标题和步骤。

2. 从数据文件中，导入实验数据。

从 corpus.txt 中导入实验所需数据以及 baidu_stopwords.txt 文件中导入停用词数据

对数据使用 jieba 进行分词，对于不是汉字的以及出现在 ls_stopwords 列表中的停顿词进行忽略，仅保留有关的关键中文信息。

```
]: 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: # 去除停顿词和字母序列
            ls_w.append(w)
    ls_jieba.append(ls_w)
```

```
]: ls_jieba
```

分词数据结果

```
In [15]: ls_jieba
```

```
Out[15]: [['南',
           '都',
           '讯',
           '记者',
           '刘凡',
           '周昌',
           '任笑',
           '继',
           '推出',
           '日票',
           '后',
           '深圳',
           '设',
           '地铁',
           'VIP',
           '头等',
           '车厢',
           '设',
           '坐票',
           '制'
```

3. 对导入的实验文档数据，根据 TF-IDF 模型完成文档的关键字的提取

计算方式为：

$$TF = \frac{\text{词汇在文本中出现的次数}}{\text{文本词汇的总个数}} \quad IDF = \log\left(\frac{\text{语料库中文本的总个数}}{\text{包含该词汇文本个数}+1}\right)$$

$$TF - IDF = TF * IDF$$

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),
```

4. 对导入的实验文档数据，利用 gensim 库完成文档的 TF-IDF 模型的构建，和关键字的提取

调用 Gensim 提供的 API 建立语料特征(索引字典)，将文本特征的原始表达转化为词袋模型对应的稀疏向量的表达。其中 corpus 是一个返回 bow 向量的迭代器。这两行代码完成对 corpus 中出现的每一个特征的 IDF 值的统计工作。

```
26]: from gensim import corpora
     from gensim import models

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

29]: ls_tfidf=list(tfidf_model[corpus])
```

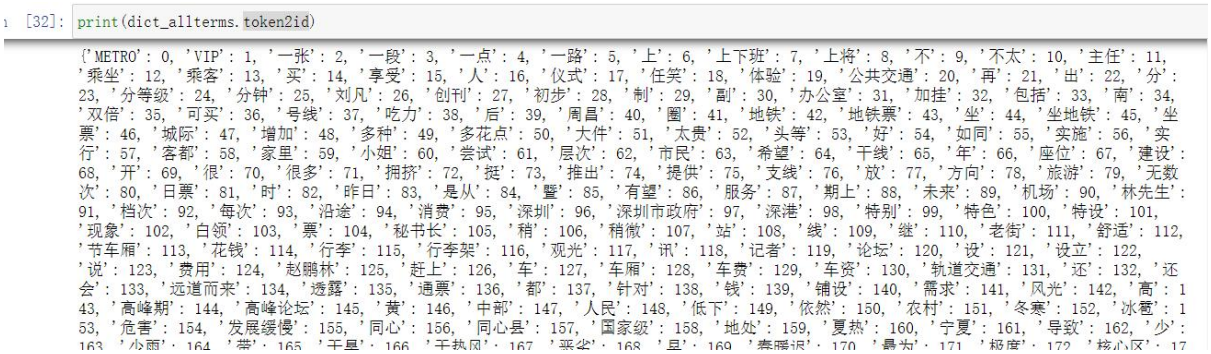
接下来和第 3 个步骤类似，也是将 5 个 5 个数据组成一组，每一组中的 tfidf 值递减，加入到一个列表 ls_all 中。Ls_all 中列表中的元组第一个代表词的序号，右边代表对应 tfidf 值。

```
: ls_all

: [[(42, 0.30162123930279794),
    (46, 0.2518279469463696),
    (67, 0.2518279469463696),
    (87, 0.20898459186357748),
    (62, 0.1888709602097772)],
   [(166, 0.46867183343578334),
    (149, 0.15622394447859447),
    (152, 0.15622394447859447),
    (153, 0.15622394447859447),
    (155, 0.15622394447859447)],
   [(228, 0.5768339095824079),
    (234, 0.23073356383296317),
    (203, 0.19937346421176586),
    (222, 0.19937346421176586),
    (229, 0.19698659999401227)],
   [(330, 0.5422735263556431),
    (376, 0.38733823311117366),
    (307, 0.27113676317782154),
    (300, 0.23428527194709178),
```

```
ls_all = []
for ls_d in ls_tfidf:
    ls_sort = sorted(ls_d, key=lambda d: d[1], reverse=True)
    ls_all.append(ls_sort[0:5])
```

token2id 将统计次数过后的词汇字典输出，分别对应一个词在文档中出现的数量。



【思考题】

1. 查阅文献，思考 TF-IDF 模型可以应用到哪些信息内容安全场景？

TD-IDF 可以用于寻找相似文章、穿衣搭配推荐、互联网情绪指标挖掘、依据用户轨迹的商户精准推销、基于兴趣的实时新闻推荐、气象关联分析等等。