论文关键词聚类技术文档

杨成成

2015005696

关键词聚类技术文档

一、目录

二、具体实现过程

1. 数据分析，文件处理

数据集包含 2005 年到 2018 年搜集的论文的部分信息，每年的信息集中在与年号相对应的文档中。每篇论文格式如下所示。

第一行是标题，括号内是论文在当期期刊的页码

第二行是作者，每个作者后面的数字代表他的职称与单位，与第三行对应

第三行是作者的职称与单位

第四行开始是摘要

第五行是关键词

第六行是标题的英文名

第七行是作者英文名，如果后面跟随数字，与第三行是同样的意义

第八行是单位名字的英文名字

第九行是英文摘要

最后是英文关键词

2. 需求分析

有如下三点需求，即为考试要求:

①关键词的抽取和识别。

②热点聚类，按照年度分析，结果存在文件中。

③研究按照年度的热点演变轨迹分析。

具体分析：

①每篇论文信息中都给定关键词，需要从每篇论文中抽取出关键词，并按照年份存储在相应的文档中。

②我认为论文中的关键词虽在文章信息中占较大的比重，但是并不能完全代表论文信息，仍需与题目，作者，摘要等信息联系在一起。故而我选用 Word2Vec 将关键词与其它信息联系在一起，由此依据全文信息计算出每个关键词的特征向量用来聚类。聚类方法选用 skcikit-learn 中的 K-means 方法，依据词向量通过计算他们之间的距离进而将相似度高的词聚集在一起来代表每年的热点。

③按照年度聚类分析可以得出每年论文的热点信息，由此可计算出每年的热点演变，从而分析演变轨迹得出最终结论。

3.数据预处理

本小结主要讲述提取论文中的关键词，目的是为了最终使用关键词进行聚类分析每年的论文热点信息。并将文档转化为 txt 文件（便于使用 jieba 分词），目的是为了为训练 Word2Vec 模型提供相应的数据。

a).提取关键词

代码部分：（deal_data.py）

```
==================================
filename='./Datasets/origin_data'
stop_words=['', '']
#将关键词和摘要提取为一段，并作分词。结果保存在 spark_ting 目录中
for name in os.listdir(filename):
    txt=docx.Document(filename+name)
    paras=txt.paragraphs
    # 便利一个文档中的所有段
    for i in range(len(paras)):
        duan=[]
        if paras[i].text[0:2]=='关键':
            seg_list=jieba.cut(paras[i].text,cut_all=False)
            # 使用正则表达式切分关键词
            seg_list=re.split('[ :,， ： 关键词\u3000]',paras[i].text)
            for word in seg_list:
                if word not in stop_words:
                    if word !='\t':
                        duan.append(word)
            with open('/Datasets_text/'+name,'a') as fl:
                for line in duan:
                    fl.write(line+' ')
                fl.write('\n')
    ====================================
```

流程详解：

使用 python-docx 打开文档按照段落读取文档做处理。通过观察文档信息，每篇文档的关键字自成一段，并且开端信息为关键词，每个关键词之间又分别用‘:,，：’几种符号分隔，由此可用 python 正贼表达式分开保存在列表中，再写入文件保存。

b).将文档转化成 txt 文件

关键代码：(transform_docx_to_txt.py)

```
====================================
```

```
path = './Datasets/ch_data/'
file_list = os.listdir(path)
f = open("./corpus/1/data.txt", 'a+')
for file in file_list:
    file_path = path + file
    document = Document(file_path)
    for paragraph in document.paragraphs:
        f.write(paragraph.text)
f.close()
```

==========

流程详解:从文档中读进文档信息，按照段的顺序存储进 txt 文件

c)使用 jieba 对文档进行分析

i. 创建自定义字典

按行读取关键词文本，将每个关键词单独放一行构成结巴分词使用的字典。以防止 jieba 分词时将关键词分开。

ii. Jieba 分词

关键代码：segment.py

==========

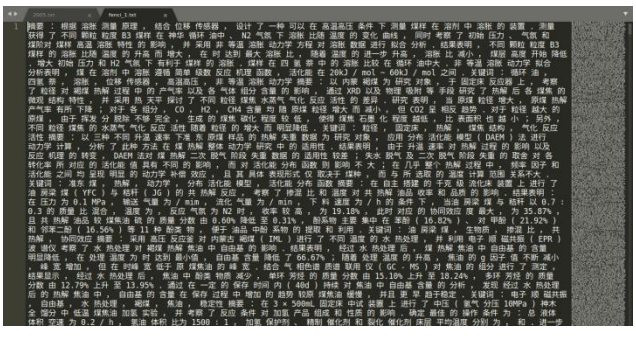
```
content = content.replace('\r\n'.encode('utf-8'),'' ''.encode('utf-8')).strip()  # 删除换行
content = content.replace('\n'.encode('utf-8'), '''.encode('utf-8')).strip()  # 删除换行
content = content.replace(' '.encode('utf-8'), ''''.encode('utf-8')).strip()  # 删除空行
content_seg = jieba.cut(content)  # 为文件内容分词
savefile(seg_dir + file_path, ' '.join(content_seg).encode('utf-8'))
```

==========

流程详解：

使用 python 自带的读写文件的读写工具打开整个 txt 文本，删除多余的换行，空格，而后使用 jieba 分词工具对整个文档进行分词。

结果展示：



4.Word2Vec 建模

a) 算法简介

Word2vec，是为一群用来产生词向量的相关模型。这些模型为浅而双层的神经网络，用来训练以重新建构语言学之词文本。网络以词表现，并且需猜测相邻位置的输入词，在 word2vec 中词袋模型假设下，词的顺序是不重要的。训练完成之后，word2vec 模型可用来映射每个词到一个向量，可用来表示词对词之间的关系，该向量为神经网络之隐藏层。

b) 实现流程

上述文本预处理中已经将需要的文本处理妥当。在此步骤直接使用即可。

代码部分

```
==========================================
"""
    训练 word2vec 模型
"""
# from gensim.models import word2vec
import gensim
import logging
# 训练模型，生成词向量
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
# 加载语料
sentences = gensim.models.word2vec.Text8Corpus('./fenci_result.txt')
print(sentences )
# 训练模型
model = gensim.models.Word2Vec(sentences, size = 2000)
# 保存模型
model.save('./models/word2vec_model.model')
model.wv.save_word2vec_format( "./models/word2vec_model.bin", binary=True)
```

流程简述：

训练 word2vec 模型需要调用 python 的 gensim 包，因此需要在 python 中安装 gensim 包，而后便可调用 gensim.models.word2vec.Text8Corpus 方法加载语料（语料的处理已经在数据预处理阶段完成），调用语料之后打印出来结果观察是否导入成功。之后进行模型的训练，使用 gensim 中的 models.Word2Vec 方法,参数为导入的预料和特征向量的维度。训练成功之后保存模型，我分别保存为 models 格式和二进制（.bin）格式，二进制格式对数据量很大时可以显著提高运行速率。

c) 模型测评

通过人工分析观察模型的预测结果。在这里选取一组测评结果作为效果展示。

代码附上：

==========

```
# 计算两个词的相似度/相关程度
# 计算两个词的相似度/相关程度
y1 = model_1.similarity("褐煤", "煤")
print(u"褐煤和煤的相似性：", y1)
print("--------------------------------\n")
y1_1 = model_1.similarity("褐煤 ", "时间")
print(u"褐煤和时间的相似性：", y1_1)
print("--------------------------------\n")
# 计算某个词的相关词列表
y2 = model_1.most_similar("煤", topn=5)  # 5 个最相关的
print(u"和宗志敏最相关的词有：\n")
for item in y2:
    print(item[0], item[1])
print("--------------------------------\n")
print("煤的特征向量为:")
print(model_1['煤'])
```

==========

测试结果：

```
褐煤和烟煤的相似性： 0.9429777872085167
--------------------------------

褐煤和时间的相似性： 0.8408016024867702
--------------------------------

和煤最相关的词有：

燃烧特性 0.997822642326355
生物质 0.9964704513549805
正交实验 0.9964126348495483
通过 0.9963518977165222
溶损反应 0.9960553646087646
--------------------------------

煤的特征向量为:
[-0.11020746 -0.17360325  0.3540364  -0.02694749 -0.21125142  0.1258093
  0.0980465   0.01503657 -0.00296208 -0.05419734  0.2667779  -0.00582215
 -0.25977713  0.30030477 -0.00340745 -0.29362777 -0.07803449 -0.17330553
```

=======

结果分析：

调用模型的 similarity 方法用来计算两关键词的相似性，在示例中可以看出褐煤和烟煤的相似性为 0.97,褐煤和时间的相似性为 0.84.再去统计与煤最相关的五个词，输出煤的几种特性和处理方法。由此可以看出模型所学得结果与我们所期望结果一致。最后测试模型能够输出聚类所需要的特征向量，测试“煤”的特征向量如上图结果所示。

5. K-Means 建模

a) 算法简介

K-means 算法的基本思想是：

以空间中 k 个点为形心进行聚类，对最靠近他们的对象归类。通过迭代的方法，逐次更新各簇的形心的值，直至得到最好的聚类结果。

假设要把样本集分为 c 个簇，算法描述如下：

（1）适当选择 c 个簇的初始形心；

（2）在第 k 次迭代中，对任意一个样本，求其到 c 个形心的欧氏距离或曼哈顿距离，将该样本归类到距离最小的形心所在的簇；

（3）利用均值等方法更新该簇的形心值；

（4）对于所有的 c 个簇形心，如果利用（2）（3）的迭代法更新后，当形心更新稳定或误差平方和最小时，则迭代结束，否则继续迭代。（误差平方和即簇内所有点到形心的距离之和）

该算法的最大优势在于简洁和快速。算法的关键在于初始中心的选择和距离公式。

b) 实现流程：

本阶段使用的是 scikit-lean 中的 kmeans 包。 K-means 聚类需要进行数值计算，在数据预处理阶段已经把关键词抽出并且保存在 txt 文本中，本步骤按照年份取出关键词并通过 word2vec 模型计算出其特征向量，再将特征向量导入 K-means 计算出聚类中心和每个关键词所属的类，从而得到每年的 热点关键词，以及关键词所属类。

具体实现过程如下代码所示：

```
==================================
    # 将文字转化成特征向量并保存在 vector 列表中
    vector = []
```

```
for line in year:
    for word in line:
        if word in model:
            vector.append(model[word])
        else:
            pass
# 构造聚类数为三类的聚类器
estimator = KMeans(n_clusters=3)# 构造聚类器
estimator.fit(vector) # 导入数据进行聚类
label_pred = estimator.labels_ # 获取聚类标签
print("==========聚类标签==============")
print(label_pred)
centroids = estimator.cluster_centers_ #获取聚类中心

print("==============聚类中心================")
inertia = estimator.inertia_ # 获取聚类准则的总和
print(inertia)
print('==========中心词==========')
# 通过对比离中心最近的向量获取中心点的文字表示
center_topic = []
for _ in centroids:
    center = 1000000
    temp = []
    for b in vector:
        if abs(np.linalg.norm(b- _)) < center:
            center = abs(np.linalg.norm(b-_))
            temp = b
    for line in year:
        for word in line:
            if word in model:
                if (model[word] == temp).all():
                    if word not in center_topic:
                        center_topic.append(word)
                else:
                    pass
print(center_topic)
```

==========

流程详解：

如上代码所示，需要先加载 word2vec 模型，读取每年你的关键词数据，逐一导入，生成特征向量保存在列表 vector 中，而后，调用 python 机器学习库 scikit-learn 中的 Kmeans 方法进行聚类。完成聚类之后调用 labels_方法查看

每个关键词所属的类，而后，调用 cluster_centers_方法查看聚类中心，并根据所得矩阵与关键词的特征向量逐一对比得出聚类中心的关键词。即为得到每年的论文热点。

c) 结果测评

上图为 2006 年的论文关键词聚类。可以看出热点信息是煤，复合催化剂，矿物质。之后几年的类似，轨迹分析阶段给出详细信息。

6. 轨迹分析

在一步中，将每年的聚类信息都保存在 topic_centers.csv 中，本阶段主要是按照年度的变化分析出近年来论文热点演变的轨迹。

所得每年论文热点关键词为：

['粉尘', '煤', '黏结剂', '2005']
['煤', '复合催化剂', '矿物质', '2006']
['性能', '氮氧化物', '炭化', '2007']
['燃煤', '活性炭', '含氧官能团', '2008']
['SCR', '炼焦煤', '煤', '2009']
['瘦煤', '型煤', '分布特征', '2010']
['节能', '煤矸石', '添加剂', '2011']
['煤', '五彩湾煤', 'SO2', '2012']
['型煤', '数学模型', '腐植酸', '2013']
['粉煤', 'CO2 吸附量', '生物质', '2014']
['含氧官能团', '微量元素', '气化', '2015']
['脱硫', '燃煤电厂', '流化床', '2016']
['配煤', '过渡金属', 'KOH', '2017']
['缩聚反应', '型煤', '微观结构', '2018']

==========

综合分析以上热点信息可以发现对煤研究热点的逐年变化：

05 年研究的热点是煤，粉尘，粘合剂。

06 年关注煤中的矿物质

07 年关注煤的性能。

08 年关注煤的燃烧。

09 年关注炼焦煤。

10 年关注型煤。

11 年关注煤的能耗

12 年关注煤中的二氧化硫

13 年使用数学模型研究煤

14 年关注粉煤

15 年关注煤中的微量元素

16 年研究煤的脱硫技术

17 年关注配煤

18 年型煤的微观结构和缩聚反应

结论：

我国近十年来对煤的研究越来越细化，经历了从宏观到微观的阶段。并且越来越关注煤的经济价值和环境价值。