研究热点的抽取和演变

段学睿

太原理工大学 大数据学院

摘　要：本文研究随着年度变化作者论文关键词的轨迹演变，数据集中有大量信息，这些信息有中文、有英文、有化学符号还有数字，通过提取数据并对数据进行聚类的方式发现所隐藏在论文中潜在的规律，来挖掘大量作者对某个研究方向的倾向和大家对于某些关键词（研究方向）的热度，这个倾向代表作者的兴趣，他随时间发生改变。本文通过处理分析大量论文信息（包括论文标题，所在期刊的页码，作者及作者信息，作者职称和单位的代码，文章摘要，关键词，标题英文名，作者英文名及英文信息，文章的论文摘要，英文关键词组成），提取各个年份论文的关键词，对各年的关键词聚类，发现热点问题，来制作热点轨迹图对热点问题的变化进行分析。它的创新之处在于：（1）只提取论文的关键词部分，将其作为分类论文研究问题的唯一要素。（2）利用 TF-IDF 对关键词进行向量化，计算出关键词权值，方便进行聚类计算。（3）使用 BIRCH 聚类的方法对关键词进行聚类，将大量数据分为三类，可以更好的得出作者对于研究问题的热点判断。

关键词：论文信息；提取关键词；聚类；热点轨迹

Extraction and evolution of research hot spots

Duan Xue-Rui

TAIYUAN UNIVERSITY OF TECHNOLOGY COLLEGE OF DATA SCIENCE

Abstract: This paper studies the evolution of the track of the key words of the author's paper in the annual change. There is a lot of information in the data set. These information has Chinese, English, chemical symbols and numbers. By extracting the data and clustering the data, we find the hidden rules hidden in the paper to dig a large number of authors for a certain research. The inclination of direction and the heat of some of the key words (research directions) represent the author's interest. He changes over time. This paper deals with the analysis of a large number of papers (including the title of the paper, the page number of the journal, the information of the author and the author, the code of the title and unit of the author, the summary of the article, the key words, the title English, the author's English name and English information, the abstract of the article, the English key words), and the extraction of each year's theory. The key words of the paper are clustering the keywords of different years, finding hot spots, making hot spot trajectories and analyzing the changes of hot topics. Its innovation lies in: (1) only extract the keywords part of the paper

as the only factor in the research of classification papers. (2) use TF-IDF to quantify keywords, calculate the weight of keywords, and facilitate cluster calculation. (3) the BIRCH clustering method is used to cluster the key words, and a large number of data are divided into three categories, which can better get the author's hot judgment on the research problem.
Key words：information of original articles; Extraction of key words; clustering; hot spots

1. 引言

大数据背景下的矿产资源研究正处于快速发展变革中，研究热点不断变化。跟踪国内外学者矿产资源领域的学术研究热点并识别其变化规律，可以帮助研究者更好的把握领域研究动态，发现学科领域中最先进、最新颖、最具发展潜力的研究主题，努力形成有自身学科特色的研究方向，引领和推动宽敞资源研究领域的发展。本文以2005年至2018年在各个杂志期刊上发表的关于矿产资源相关领域的论文为研究对象，采用BIRCH聚类方法进行分析，力求形象、客观的揭示各年度该领域的研究概况和热点轨迹，为更好的总结现有问题、提高我国该领域的研究水平和方向提供有益参考。

2. 数据来源和研究方法

期刊近十余年的数据集，内容包括论文标题，所在期刊的页码，作者及作者信息，作者职称和单位的代码，文章摘要，关键词，标题英文名，作者英文名及英文信息，文章的论文摘要，英文关键词。年份为2005年至2018年，除去论文标题、所在期刊页码，作者及作者信息，作者职称和单位的代码，文章摘要，标题英文名，作者英文名及英文信息，文章的论文摘要，英文关键词后，共有1078条数据。本研究主要采用TF-IDF和BIRCH的方法进行文本向量化和聚类分析，在分析过程中，首先对所有数据集中的数据进行识别和提取，清洗不必要的数据，纠正错误信息，然后进行数据格式的转换，生成文本文档，该文本文档中的内容仅包括所有论文的关键词部分；之后对所得到的关键词进行TF-IDF权值计算，得到TF-IDF矩阵；在得到该矩阵之后，便可以对关键词进行聚类分析；得到聚类结果，就可做本研究的主要内容：对论文关键词进行描述统计和可视化分析，以揭示该领域学科的研究热点。

3. 相关工作

a) TF-IDF关键词权重计算

信息检索是当前应用十分广泛的一种技术，论文检索、搜索引擎都属于信息检索的范畴。通常，人们把信息检索问题抽象为：在文档集合 D 上，对于由关键词 w[1] ... w[k]组成的查询串 q，返回一个按查询 q 和文档 d 匹配度 relevance(q, d)排序的相关文档列表 D'。

对于这一问题,先后出现了布尔模型、向量模型等各种经典的信息检索模型，它们从不同的角度提出了自己的一套解决方案。布尔模型以集合的布尔运算为基础，查询效率高，但模型过于简单，无法有效地对不同文档进行排序，查询效果不佳。向量模型把文档和查询串都视为词所构

成的多维向量，而文档与查询的相关性即对应于向量间的夹角。不过，由于通常词的数量巨大，向量维度非常高，而大量的维度都是0，计算向量夹角的效果并不好。另外，庞大的计算量也使得向量模型几乎不具有在互联网搜索引擎这样海量数据集上实施的可行性。

目前，真正在搜索引擎等实际应用中广泛使用的是 tf-idf 模型。tf-idf 模型的主要思想是：如果词 w 在一篇文档 d 中出现的频率高，并且在其他文档中很少出现，则认为词 w 具有很好的区分能力，适合用来把文章 d 和其他文章区分开来。该模型主要包含了两个因素：

1) 词 w 在文档 d 中的词频 tf (Term Frequency)，即词 w 在文档 d 中出现次数 count(w, d)和文档 d 中总词数 size(d)的比值：

tf(w,d) = count(w, d) / size(d)

2) 词 w 在整个文档集合中的逆向文档频率 idf (Inverse Document Frequency)，即文档总数 n 与词 w 所出现文件数 docs(w, D)比值的对数:

idf = log(n / docs(w, D))

tf-idf 模型根据 tf 和 idf 为每一个文档 d 和由关键词 w[1]...w[k]组成的查询串 q 计算一个权值，用于表示查询串 q 与文档 d 的匹配度：

tf-idf(q, d)

= sum { i = 1..k | tf-idf(w[i], d) }

= sum { i = 1..k | tf(w[i], d) * idf(w[i]) }

TF-IDF 的主要思想是：如果某个词或短语在一篇文章中出现的频率 TF 高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。TF-IDF 实际上是：TF * IDF。

词频（Term Frequency，TF）指的是某一个给定的词语在该文件中出现的频率。即词 w 在文档 d 中出现的次数 count(w, d)和文档 d 中总词数 size(d)的比值。这个数字是对词数(term count)的归一化，以防止它偏向长的文件。（同一个词语在长文件里可能会比短文件有更高的词数，而不管该词语重要与否。）

逆向文件频率（Inverse Document Frequency，IDF）是一个词语普遍重要性的度量。某一特定词语的 IDF，可以由总文件数目除以包含该词语之文件的数目，再将得到的商取对数得到。即文档总数 n 与词 w 所出现文件数 docs(w, D)比值的对数。TF-IDF 根据 tf 和 idf 为每一个文档 d 和由关键词 w[1]...w[k]组成的查询串 q 计算一个权值，用于表示查询串 q 与文档 d 的匹配。

某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的 TF-IDF。因此，TF-IDF 倾向于过滤掉常见的词语，保留重要的词语。

b) 在 python 中计算 TF-IDF：

在 Python 中，scikit-learn 包下有计算 TF-IDF 的 api，其效果也很不错。首先得安装 Scikit-clearn。

scikit-learn 包进行 TF-IDF 分词权重计算主要用到了两个类：CountVectorizer 和 TfidfTransformer。其中 CountVectorizer 是通过 fit_transform 函数将文本中的词语转换

为词频矩阵，矩阵元素 a[i][j] 表示 j 词在第 i 个文本下的词频。即各个词语出现的次数，通过 get_feature_names()可看到所有文本的关键字，通过 toarray()可看到词频矩阵的结果。

```
def idf1(i):
    corpus = []
    f = open(file1[i], 'r+')
    content = f.read()
    f.close()
    corpus.append(content)
    #将得到的词语转换为词频矩阵
    vectorizer = CountVectorizer()
    #统计每个单词的tf-idf权值
    transformer = TfidfTransformer()
    #计算出tf-idf并转换为tf-idf矩阵
    tfidf = transformer.fit_transform(vectorizer.fit_transform(corpus))
    # 所有文本的关键字
    word = vectorizer.get_feature_names()
    # 对应的tfidf矩阵
    weight = tfidf.toarray()
    return word,weight
```

c) BIRCH 聚类算法

BIRCH 的全称是利用层次方法的平衡迭代规约和聚类（Balanced Iterative Reducing and Clustering Using Hierarchies），它是用层次方法来聚类和规约数据。刚才提到，BIRCH 只需要单遍扫描数据集就能进行聚类，那它是怎么做到的呢？BIRCH 算法利用了一个树结构来帮助我们快速的聚类，这个数结构类似于平衡 B+树，一般将它称之为聚类特征树(Clustering Feature Tree，简称 CF Tree)。这颗树的每一个节点是由若干个聚类特征(Clustering Feature，简称 CF)组成。从下图我们可以看看聚类特征树是什么样子的：每个节点包括叶子节点都有若干个 CF，而内部节点的 CF 有指向孩子节点的指针，所有的叶子节点用一个双向链表链接起来。

BIRCH 算法的主要过程，就是建立 CF Tree 的过程。除了建立 CF Tree 来聚类 BIRCH 算法还有一些可选的算法步骤：

1） 将所有的样本依次读入，在内存中建立一颗 CF Tree。

2）（可选）将第一步建立的 CF Tree 进行筛选，去除一些异常 CF 节点，这些节点一般里面的样本点很少。对于一些超球体距离非常近的元组进行合并。

3）（可选）利用其它的一些聚类算法比如 K-Means 对所有的 CF 元组进行聚类，得到一颗比较好的 CF Tree.这一步的主要目的是消除由于样本读入顺序导致的不合理的树结构，以及一些由于节点 CF 个数限制导致的树结构分裂。

4)（可选）利用第三步生成的 CF Tree 的所有 CF 节点的质心，作为初始质心点，对所有的样本点按距离远近进行聚类。这样进一步减少了由于 CF Tree 的一些限制导致的聚类不合理的情况。

BIRCH 算法可以不用输入类别数 K 值，这点和 K-Means，Mini Batch K-Means 不同。如果不输入 K 值，则最后的 CF 元组的组数即为最终的 K，否则会按照输入的 K 值对 CF 元组按距离大小进行合并。我在代码中使用了 K=3 。

```
#设置3个类
clusterer = Birch(n_clusters=3)
y = clusterer.fit_predict(weight)
```

d) BIRCH 算法总结：

BIRCH 算法的主要优点有：

1)节约内存，所有的样本都在磁盘上，CF Tree 仅仅存了 CF 节点和对应的指针。

2)聚类速度快，只需要一遍扫描训练集就可以建立 CF Tree，CF Tree 的增删改都很快。

3)可以识别噪音点，还可以对数据集进行初步分类的预处理

BIRCH 算法的主要缺点有：

1)由于 CF Tree 对每个节点的 CF 个数有限制，导致聚类的结果可能和真实的类别分布不同.

2)对高维特征的数据聚类效果不好。

此时可以选择 Mini Batch K-Means

3)如果数据集的分布簇不是类似于超球体，或者说不是凸的，则聚类效果不好。

4. 研究热点轨迹

使用了 matplotlib 库制作轨迹折线图。

Matplotlib 是一个 Python 的 2D 绘图库，它以各种硬拷贝格式和跨平台的交互式环境生成出版质量级别的图形。

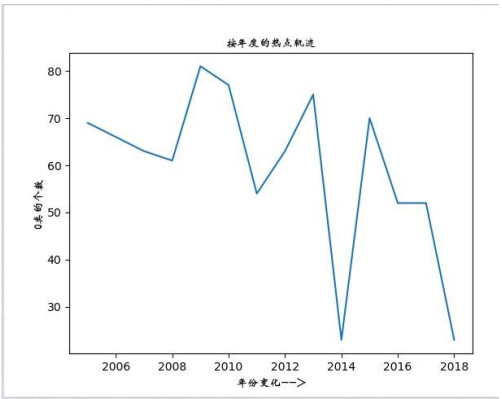
通过 Matplotlib,开发者可以仅需要几行代码，便可以生成绘图，直方图，功率谱，条形图，错误图，散点图等。

Matplotlib 中基本的图标包含的元素有：x 轴、y 轴、水平和垂直的轴线、x 轴和 y 轴的刻度、刻度标识坐标轴的分隔（包括最小刻度和最大刻度）、绘图区域、实际绘图区域。

.tittle 方法是为给图表加标题，.xlable 方法是 x 轴的注释，.ylable 方法是 y 轴的注释。注意：需要先保存图片后再用.show 方法展示图片，否则存储的图片将是空白图。

```
plt.figure(1, dpi=100)
plt.plot(Transverse,Longitudinal)
plt.title(u'按年度的热点轨迹',fontproperties=myfon
plt.xlabel('年份变化-->',fontproperties=myfont)
plt.ylabel('0类的个数',fontproperties=myfont)
plt.savefig('hot.jpg')
# plt.show()
```

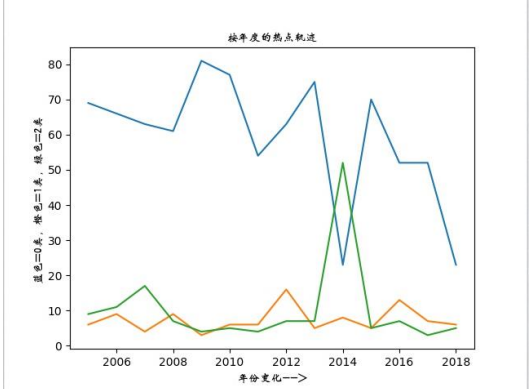
得到的热点轨迹图为：



根据热点轨迹图可知，从各年度的热点出现次数来看，类别 0 在 2014 年之前一直是研究的热点方向，但是在 2104 年这年，类别 0 的研究热度有很大程度的下滑。

```
这是2014年的聚类结果：
[0 0 0 2 2 1 2 0 2 2 2 2 2 0 2 2 2 2 2 2 0 0 2 2 2 2 0 0 2 2 2 2 0 2 0 2 0 2 0 2 2 2
 1 1 2 2 0 0 2 1 2 2 2 1 0 1 2 2 2 2 2 0 2 2 2 0 2 2 2 1 0 2 2 0 1 0 2 2 2 2 2
 0 0 2 2 2 2 0 2 2 2]
(83,)
第0类的个数为：23
第1类的个数为：8
第2类的个数为：52
```

打印 2014 年的聚类结果，发现 2014 年第二类聚类比较热门。分析第二类在各年度的热点轨迹图。



发现 2014 年是比较特殊的一年,在这一年第 0 类聚类的热度显著下降，而第 2 类聚类的热度明显上升。

5. 方法总结：

除了 BIRCH 聚类方法之外，其实还有很多效果很好的聚类方法，在这里我再简单介绍一下 K-Means 和 Mini Batch K-Means 方法，这两种聚类方法的聚类效果也非常好。

K-means 通常被称为劳埃德算法。在基本术语中，算法有三个步骤。第一步选择初始质心，最基本的方法是k从数据集中选择样本 X。初始化之后，K-means 包括在其他两个步骤之间循环。第一步将每个样品分配到最近的质心。第二步通过获取分配给每个先前质心的所有样本的平均值来创建新质心。计算旧

的和新的质心之间的差异，并且算法重复这最后两个步骤，直到该值小于阈值。换句话说，它重复直到质心不显着移动。 K-means 等价于具有小的，全等于的对角协方差矩阵的期望最大化算法。如果有足够的时间，K-means 将始终收敛，但这可能是局部最小值。这很大程度上取决于质心的初始化。结果，计算通常进行若干次，其中质心的初始化不同。帮助解决此问题的一种方法是 k-means ++初始化方案，该方案已在 scikit-learn（使用 init='k-means++' 参数）中实现。这使得质心初始化（通常）彼此远离，导致比随机初始化更可靠的结果。

Mini Batch K-Means 算法是 K-Means 算法的变种，采用小批量的数据子集减小计算时间，同时仍试图优化目标函数，这里所谓的小批量是指每次训练算法时所随机抽取的数据子集，采用这些随机产生的子集进行训练算法，大大减小了计算时间，与其他算法相比，减少了 k-均值的收敛时间，小批量 k-均值产生的结果，一般只略差于标准算法。

该算法的迭代步骤有两步：

1：从数据集中随机抽取一些数据形成小批量，把他们分配给最近的质心。

2：更新质心。

与 K 均值算法相比，数据的更新是在每一个小的样本集上。对于每一个小批量，通过计算平均值得到更新质心，并把小批量里的数据分配给该质心，随着迭代次数的增加，这些质心的变化是逐渐减小的，直到质心稳定或者达到指定的迭代次数，停止计算。

Mini Batch K-Means 比 K-Means 有更快的 收敛速度，但同时也降低了聚类的效果，但是在实际项目中却表现得不明显。而比较 BIRCH 与 Mini Batch K-Means 的话：

BIRCH 算法可以不用输入类别数 K 值，这点和 K-Means，Mini Batch K-Means 不同。如果不输入 K 值，则最后的 CF 元组的组数即为最终的 K，否则会按照输入的 K 值对 CF 元组按距离大小进行合并。

一般来说，BIRCH 算法适用于样本量较大的情况，这点和 Mini Batch K-Means 类似，但是 BIRCH 适用于类别数比较大的情况，而 Mini Batch K-Means 一般用于类别数适中或者较少的时候。BIRCH 除了聚类还可以额外做一些异常点检测和数据初步按类别规约的预处理。但是如果数据特征的维度非常大，比如大于 20，则 BIRCH 不太适合，此时 Mini Batch K-Means 的表现较好。

对于调参，BIRCH 要比 K-Means，Mini Batch K-Means 复杂，因为它需要对 CF Tree 的几个关键的参数进行调参，这几个参数对 CF Tree 的最终形式影响很大。

6. 结语

基于以上对矿产资源研究领域论文分析结合关键词热点聚类所得的结果，可知我国煤炭领域研究随着年份的增长，研究的热点方向大体没有发生变化，学者针对这些热点问题一直在不断研究。

参考文献

[1] Python docx 库的使用方法 http://python-

docx.readthedocs.io/en/latest/

[2] Python TF-IDf 计算关键词权重 https://www.cnblogs.com/chenbjin/p/3851165.html

[3] Python scikit learn 中的BIRch聚类算法 https://www.cnblogs.com/pinard/p/6200579.html

[4] “k-means ++：小心播种的优势” Arthur，David和Sergei Vassilvitskii，第18届年度ACM-SIAM离散算法研讨会论文集，工业与应用数学学会（2007）

[5] “Web Scale K-Means 聚类” D. Sculley，第19届万维网国际会议论文集（2010）

[6] Tian Zhang，Raghu Ramakrishnan，Maron Livny BIRCH：一种适用于大型数据库的高效数据聚类方法。http://www.cs.sfu.ca/CourseCentral/459/han/papers/zhang96.pdf

[7] Roberto Perdisci JBirch - BIRCH集群算法的Java实现 https://code.google.com/archive/p/jbirch

[8] Sun Wei, Zhao Shikui, Zhang Yantong（1. Beihang University, Beijing 100191; 2. Counselors′ Office of the State Council, Beijing 100006）《Analysis of Chinese scholars′ research on science funding》“Bulletin of National Natural Science Foundation of China”

[9] 陈淋, 屈文建. 基于共词分析的我国图书情报学研究主题演化分析[J]. 新世纪图书馆,2017,(12):13-18.doi:10.16810/j.cnki.1672-514X.2017.12.003.

[10]“中国煤炭科技创新网“ http://www.mtkj.org/#