Statistics and Application 统计学与应用, 2020, 9(2), 265-276
Published Online April 2020 in Hans. http://www.hanspub.org/journal/sa
https://doi.org/10.12677/sa.2020.92029

Hans汉斯

The Movie Content of K-Means Clustering Analysis

Lijuan Yuan

Yunnan University of Finance and Economics, Kunming Yunnan
Email: m15182063060@163.com

Received: Mar. 26th, 2020; accepted: Apr. 10th, 2020; published: Apr. 17th, 2020

Abstract

With the improvement of living standards, people's spiritual life is becoming more and more colorful. As a part of people's pursuit of spiritual culture and cultural innovation, film has become the focus of attention. In a fast-paced social environment, being able to choose your favorite movie in a short period of time is undoubtedly the best case. To improve the quality of people searching and selecting movies, one way is to categorize existing movies by theme. There are two ways to classify texts according to topics: supervised and unsupervised learning. Supervised learning requires manual labeling, which is very time-consuming and labor-intensive. Unsupervised learning can actively classify categories based on movie content, which not only saves time, but also reduces the economic consumption caused by manual labeling. Therefore, from the perspective of movie content, this paper proposes to use K-Means clustering method to the unsupervised classification of movies. Finally, the classification results are visualized. In each category, movies have a common theme.

Keywords

Movie Content, Topic Classification, Unsupervised Learning, K-Means Clustering

基于电影内容的K-Means聚类分析

袁丽娟

云南财经大学，云南 昆明
Email: m15182063060@163.com

收稿日期：2020年3月26日；录用日期：2020年4月10日；发布日期：2020年4月17日

摘 要

随着生活水平的日益提高，人们的精神生活越来越丰富多彩。电影作为人们追求精神文化和文化创新的

文章引用: 袁丽娟. 基于电影内容的 K-Means 聚类分析[J]. 统计学与应用, 2020, 9(2): 265-276.
DOI: 10.12677/sa.2020.92029

一部分，成为关注的焦点。在快节奏的社会环境下，能够在较短的时间内，选择喜欢的电影，无疑是最好的情况。为提高人们搜索和选择电影的质量，方式之一是对已有的电影按照主题进行分类。对文本按照主题分类的方式，存在有监督和无监督学习两种方式。有监督的学习，需要人工标注，十分耗时耗力。无监督学习，可以主动根据电影内容进行划分类别，不仅省时，而且降低了人工标注带来的经济消费。因此，本文从电影内容角度出发，提出使用K-Means聚类方法，对电影进行无监督分类；最后，可视化分类结果，每一类别下，电影有共同的主题。

关键词

电影内容，主题分类，无监督学习，K-Means聚类

1. 引言

自改革开放以来，中国经济发展迅速，人民生活的物质条件越来越好，精神生活逐渐丰富多彩，人们追求精神文化，已成为必不可少的生活元素。十三五规划以来，文化创新作为新名词，登上社会生活的舞台。电影作为人们追求精神文化和文化创新的一员，成为关注的焦点。日常生活中，人们会去电影院购买电影票，观看电影，感受电影带来的情绪，写下影评或是与身边的人交流探讨。于是在电影相关的网站上面也会有电影历史上的评分及排行榜，随之而来就会有疑问：什么样的电影吸引着大众？好的电影有什么共同特征？人们如何快速选择偏好的电影等问题。

在这样的背景前提下，本文搜索了猫眼电影上面的全球排名 100 的电影，根据榜单上面给出的电影名称及其剧情的介绍，尝试将 100 部电影进行 K-Means 聚类，分析 100 部电影那些具有相似性，并给出属于那些类型的电影，对想要观看排名榜上前 100 部的电影的观看者提出建议：按其喜好推荐观看。

2. 问题提出与分析

以往在猫眼电影上面搜索电影来看时，总会搜索出很多电影，想着按照榜单上面 top100 的电影来看，但时间有限时，又没有时间将每部电影都看完，有些类型的电影又不感兴趣，或是只想看一种类型的电影，自己又不知道是榜单上面的哪一部电影时，就会随随便便看一部电影，在这样的情况下，有时候运气好，看到有兴趣的电影，会觉得十分值得；但有时候运气不佳，选择看的电影只想要睡觉，不免会觉得遗憾。而在学习了《机器学习系统设计》后，知道可以通过文本聚类和主题模型的方式识别出哪些电影是一类，从而进行选择。于是尝试对 100 部电影剧情内容进行分析，做分类处理，去运用文本聚类方法。

2.1. 思路分析

本文将对猫眼电影榜单上 top100 的电影进行聚类，根据剧情介绍分为几类。解决此问题，首先需要明确要分析的信息，获取信息；其次是对获取的信息进行处理；然后是进行聚类；最后对聚类结果的分析。其具体步骤如下(见图 1)。

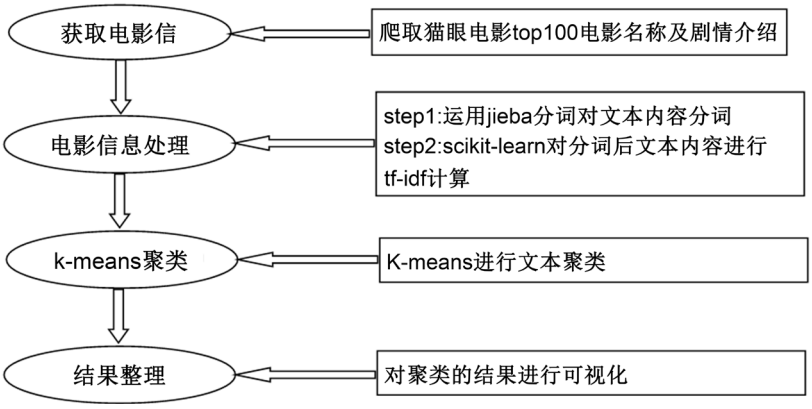

Figure 1. The analysis flowchart of K-Means cluster
图 1. K-Means 聚类分析流程图

2.2. 电影文本内容的选择

在猫眼电影网站榜单 top100 榜上面，每部电影的信息有很多：电影名称、主演、上映时间、电影属于的国家、电影图片以及电影评分等(如下图 2)，那么在这些信息中需要哪些信息做分析呢？考虑本文主要目的是将这 100 部电影进行文本聚类，将 100 部电影划分为几类，以供判断和选择所需看的电影，因此本文选取电影名称及其电影剧情作为文本内容进行聚类。

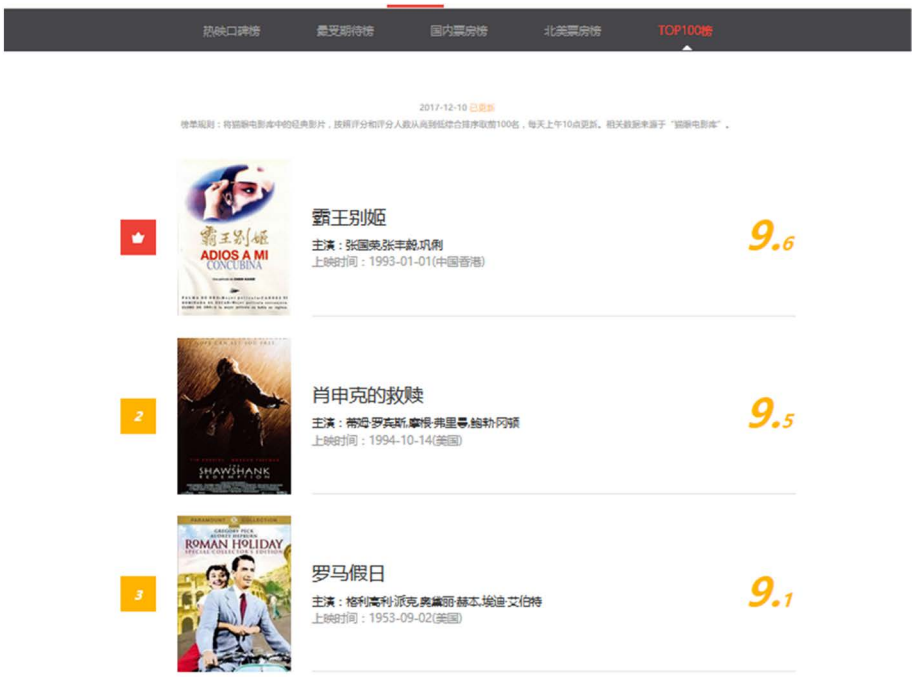

Figure 2. The screenshot of top 100 list movie list
图 2. top100 榜电影名单截图

3. 建立模型及求解

在问题分析中，本文主要步骤分为四步，即获取电影信息，电影信息处理，转为数据矩阵，进行 K-Means 聚类，最后保存在 txt 的文件中。下面进行建模和求解。

3.1. 电影文本信息获取

本文选择的文本信息为电影名称和电影剧情，在猫眼电影上面找到这两块内容，采用网络爬虫的方式抓取内容，并将其保存在 movie_top_1.txt 文件中。网络爬虫框架流程[1] [2]：

1. 首先从互联网页面中选择一部分网页，以这些网页的链接地址作为种子 URL；
2. 将这些种子 URL 放入待抓取 URL 队列中；
3. 爬虫从待抓取 URL 队列依次读取，并将 URL 通过 DNS 解析，把链接地址转换为网站服务器对应的 IP 地址。
4. 然后将 IP 地址和网页相对路径名称交给网页下载器；
5. 网页下载器负责页面内容的下载；
6. 对于下载到本地的网页，一方面将其存储到页面库中，等待建立索引等后续处理；另一方面将下载网页的 URL 放入已抓取 URL 队列中，这个队列记载了爬虫系统已经下载过的网页 URL，以避免网页的重复抓取。
7. 对于刚下载的网页，从中抽取出所包含的所有链接信息，并在已抓取 URL 队列中检查，如果发现链接还没有被抓取过，则将这个 URL 放入待抓取 URL 队列。
8. 在之后的抓取调度中会下载这个 URL 对应的网页，如此这般，形成循环，直到待抓取 URL 队列为空。

根据上面的网络爬虫原理，调用 python 中 requests 模块、xpath 模块以及 re 模块，获取猫眼电影上面的 top100 榜中电影名称和剧情内容，并按一行一部电影的方式存放在 movie_top_1.txt 文件中，爬取信息部分如下图 3。

Figure 3. Movie name and plot introduction
图 3. 电影名称及剧情介绍

图 3 中是保存抓取的 100 部电影，并以一行一部电影名称和电影剧情介绍的方式保存。第一列均为电影名称，并以 top100 榜上面的排名顺序进行排列。

3.2. 电影文本内容处理

首先对获取的电影剧情内容进行中文分词处理，然后计算词语的 TF-IDF 值，将词语向量转化为数字矩阵，以此方式进行 K-Means 聚类分析。

3.2.1. 中文分词原理

中文分词是指将连续的汉字序列按照一定的规范重新组合成词序列的过程。

现有的分词算法分为三大类[3]：基于字符串匹配的分词方法、基于理解的分词方法和基于统计的分

词方法。

这种方法又叫做机械分词方法，它是按照一定的策略将待分析的汉字串与一个“充分大的”机器词典中的词条进行配，若在词典中找到某个字符串，则匹配成功(识别出一个词)。

1) 正向最大匹配法(由左到右的方向)；

2) 逆向最大匹配法(由右到左的方向)；

3) 最少切分(使每一句中切出的词数最小)；

4) 双向最大匹配法(进行由左到右、由右到左两次扫描)。

基于理解的分词方法：这种分词方法是通过让计算机模拟人对句子的理解，达到识别词的效果。其基本思想就是在分词的同时进行句法、语义分析，利用句法信息和语义信息来处理歧义现象。它通常包括三个部分：分词子系统、句法语义子系统、总控部分。在总控部分的协调下，分词子系统可以获得有关词、句子等的句法和语义信息来对分词歧义进行判断，即它模拟了人对句子的理解过程。这种分词方法需要使用大量的语言知识和信息。由于汉语语言知识的笼统、复杂性，难以将各种语言信息组织成机器可直接读取的形式，因此目前基于理解的分词系统还处在试验阶段。

基于统计的分词方法：给出大量已经分词的文本，利用统计机器学习模型学习词语切分的规律(称为训练)，从而实现对未知文本的切分。

3.2.2. Jieba 分词原理及文本内容的分词

本文采用 Jieba 分词技术。其基本原理[4]如下：

1) 基于 Trie 树结构实现高效的词图扫描，生成句子中汉字所有可能成词情况所构成的有向无环图(DAG)；

2) 采用了动态规划查找最大概率路径，找出基于词频的最大切分组合；

3) 对于未登录词，采用了基于汉字成词能力的 HMM 模型，使用了 Viterbi 算法；

在了解 Jieba 分词原理后，选择 Jieba 分词中的精确模式，对 movie_top_1.txt 中的文本内容，进行最精确地切开。分词结果保存在 movie_top_jieba.txt 文件中，其部分图如下图 4。

Figure 4. The segmentation results from movie text content
图 4. 电影文本内容分词结果

图 4 中可以看到，第一列均为电影名称，其他为电影剧情内容，经过 Jieba 分词后，文本内容以空格的形式分开。

3.2.3. 计算 TF-IDF 值

在对电影的文本内容分词后，现在需要将文档相似度问题转换为数学向量矩阵问题，通过 VSM 向量空间模型来存储每个文档的词频和权重。首先需要抽取电影文本内容分词后的特征，然后根据每个词语对文档的贡献度不同，对这些词语赋予不同的权重。下面采用 TF-IDF 方法计算词语在文档中的权重方法。

TF-IDF 权重计算法是一种统计方法，用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。

在分词后的电影内容里，词频 TF 指的是一个词语在该电影内容中出现的次数，该词频进行了归一化处理，以防止偏向于较长的某一部电影剧情。而逆向词频率 IDF 是对一个词语进行普遍重要性的度量，是根据 movie_top_jieba.txt 中的电影数目与包含该词语的电影数目之比取对数而来。而在某一电影剧情内容的高词语频率，以及该词语在整个电影剧情集合中的低文件频率，将会产生出高权重的 TF-IDF。因此，TF-IDF 倾向于过滤常见的词语，保留重要的词语。这样既可以除去常见词语(例如，“的”，“地”，“有”…等)，又将文本内容中的关键字提取了出来，并赋予了权重。

在本文需要分析的文本内容中，对于在某一特定的电影中的词语 t_i，其重要性表示为：

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

其中 $n_{i,j}$ 是词语 t_i 在电影剧情 d_j 中出现的次数，$\sum_k n_{k,j}$ 是电影剧情 d_j 中所有字词的出现次数之和。

一个词语普遍重要性的 IDF，可以表示为：

$$idf_i = \log \frac{|D|}{\left|\left\{j : t_i \in d_j\right\}\right|}$$

其中：$|D|$ 是指语料库中存放的电影总数，$\left|\left\{j : t_i \in d_j\right\}\right|$ 是指包含词语 t_i 的电影数目，一般情况下使用 $1+\left|\left\{j : t_i \in d_j\right\}\right|$。

一个词语的 TF-IDF 值，可以表示为：

$$TFIDF_{i,j} = tf_{i,j} \times idf_{i,j}$$

因此，根据 TF-IDF 的原理，可以将词语的权重计算出，并将结果保留在 movie_TF-IDF.txt 文件中，部分截图如下图 5。

Figure 5. The features and TF-IDF values from movie text
图 5. 特征词及 TF-IDF 值

图 5 中上部分的词语即从电影文本内容中提取出的特征词，共 5087，而下面的数字即为每个词语在不同电影剧情中所占的权重，也是 TF-IDF 值。

3.3. K-Means 聚类

在计算出词语的 TF-IDF 值矩阵后，下面进行电影文本内容的 K-Means 聚类分析。K-Means 算法[5] [6]是一种聚类算法，其基本思想是：通过迭代寻找 k 个聚类的一种划分方案，使得用这 k 个聚类的均值代表相应各类样本时所得的总体误差最小。K-Means 算法的基础是最小误差平方和准则。其代价函数是：

$$J(c,\mu)=\sum_{i=1}^{k}\left\|x^{(i)}-\mu_{c^{(i)}}\right\|^2$$

上式中，$\mu_{c^{(i)}}$ 表示第 i 个聚类的均值。

具体算法如下：

Step 1：随机选取 k 个聚类质心点；

Step 2：对每一个电影 i，计算其应该属于的类：$c^{(i)}:=\arg\min_j\left\|x^{(i)}-\mu_j\right\|^2$；

Step 3：对每一个类 j，重新计算该类的质心：$\mu_j:=\dfrac{\sum_{i=1}^{m}\mathrm{I}\left\{c^{(i)}=j\right\}x^{(i)}}{\sum_{i=1}^{m}\mathrm{I}\left\{c^{(i)}=j\right\}}$；

Step 4：重复 Step 2 和 Step 3，直至收敛。

本文需对 100 部电影进行聚类，选取 k = 8，进行聚类，结果保存在 movie_kmeans.txt 文件中，部分的结果如图 6：

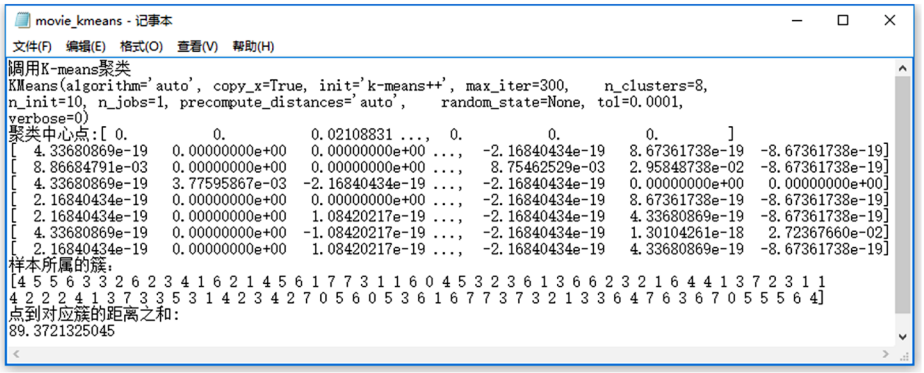

Figure 6. The results of K-Means clustering
图 6. K-Means 聚类结果

图 6 能够看到 8 个聚类中心的坐标，以及 100 个电影所属的聚类标签或簇，并得出了每部电影到其所属簇的距离之和为 89.3721。但这样的结果并不够直观，很难看出这 100 部电影，哪些被分为了一类，也不知道每一个类里有多少部电影，为能够使结果更直观，对上图中的结果进行可视化处理。

4. 结果分析

在对猫眼电影 top100 榜上的电影进行了聚类后，需对聚类的结果进行合理的解释和分析。

为能够直观看出 100 部电影的 K-Means 聚类结果，现对 K-Means 聚类结果处理，并将结果写入 name_label.txt 文件中，结果图如图 6。

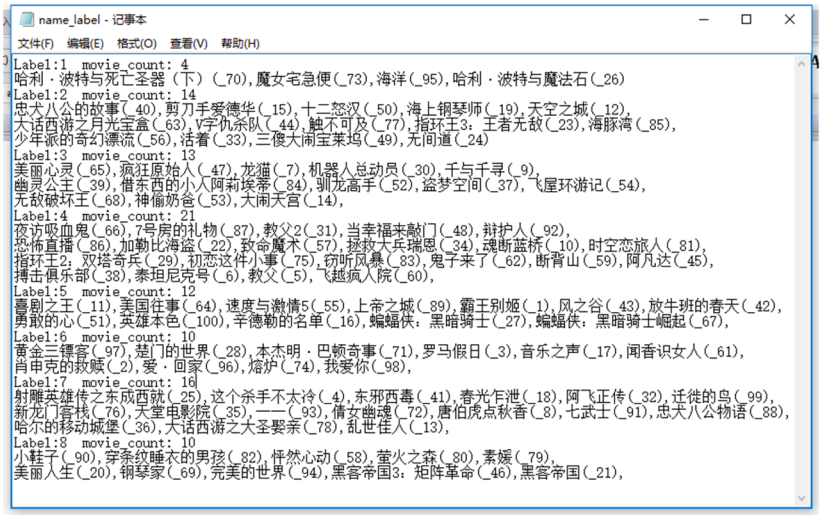

Figure 7. The graph of clustering result
图 7. 聚类结果图

图 7 中能够直观看出 100 部电影被聚类为 8 类。Label：1 中有 4 部电影，分别为：哈利·波特与死亡圣器(下)，魔女宅急便，海洋，哈利·波特与魔法石；这 4 部电影在猫眼电影的 top100 榜上的排名为 70，73，95，26；查看这 4 部电影的剧情介绍，发现这 4 部电影剧情介绍均是与魔法相关的，那么可以给出这 4 部电影一个共同的标签为：“魔法”。这样若是对魔法故事感兴趣的朋友就可以根据“魔法”标签，选择看下面的哪一部电影了。

根据查找每一类中电影的共同点，可以给 8 类电影加上标签；可以根据用户在猫眼电影 top100 榜上搜索一部电影时，给出相似类型的电影名称，以供用户在类型相似的电影中选择自己想看的电影。

5. 模型评价

本文从当下人们观看电影这一视角，思考如何查找相似类型的电影。首先从猫眼电影网站上，获取所需要的信息；然后运用 Jieba 分词对获取的信息进行中文分词；接着运用 scikit-learn 方法计算词语的 TF-IDF 值；最后进行 K-Means 聚类并作分析。但在对 100 部电影聚类结果分析的过程中，发现了如下的一些不足：

1) K-Means 聚类的 k 值是人为给定，具有主观性和随意性；

2) K-Means 聚类的结果与初始选择的中心有关，初始选择的中心点越好，其聚类效果会越好，因此很难得出那一次的聚类结果最好；

3) 在对 8 类电影进行贴标签时发现，属于一个类的电影并不一定具有共同点或具有的共同点不如其他类多；考虑到聚类方法只是对文字的相似性进行了判断，但未能对文字隐藏的意思作出判断；

4) 聚类结果为 8 类，并不能直接提供每一类的标签特征。

对于上述中的不足，前两条的很难以实现修改，第三条和第四条中的问题，可以通过对 100 部电影建立主题模型，进而实现对文字隐藏意思的判断并给出主题的标签。

虽然本文有不足之处，但也有可取之处：

1) 提供了对电影剧情的分类方法；

2) 为用户在选择观看影片时，缩小了选择的范围；

3) 提供了电影推荐的思路；

4) 可以根据用户以往浏览的影片信息与分类的电影进行匹配，给用户推荐电影。

参考文献

[1] 胡松涛. Python 网络爬虫实战[M]. 北京: 清华大学出版社, 2017, 83-84.

[2] 韦伟. 精通 Python 网络爬虫[M]. 北京: 机械工业出版社, 2017, 52-61.

[3] 百度百科. 中文分词[EB/OL] https://baike.baidu.com/item/%E4%B8%AD%E6%96%87%E5%88%86%E8%AF%8D/371496?fr=aladdin

[4] 中文分词与 Jieba 分词原理[EB/OL]. http://blog.csdn.net/john_xyz/article/details/54645527, 2017-01-21.

[5] [Python]基于 K-Means 和 TF-IDF 的文本聚类代码简单实现[EB/OL]. http://blog.csdn.net/eastmount/article/details/50473675, 2016-01-08.

[6] K-Means 聚类过程的动态可视化[EB/OL]. http://blog.csdn.net/happyyear1/article/details/50973675, 2016-03-24.

附录

附录 1：电影文本内容的分词

```
# encoding=utf-8
from __future__ import print_function
import sys
import codecs
sys.path.append("../")
import jieba
reload(sys)
sys.setdefaultencoding('utf8')
#  实现分词
def cuttest(test_sent):
    fp = open('movie_top_jieba.txt','a')
    result = jieba.cut(test_sent)
    output = " ".join(result)
    output = output.decode('utf8')
    fp.writelines(str(output))
    fp.close()
    print (output)
#  读取电影文本内容
def testcase():
    fp = codecs.open('movie_top_1.txt','r','utf-8')
    f = fp.read()
    # print (f)
    cuttest(f)
#执行程序
if __name__ == "__main__":
    testcase()
    jieba.set_dictionary("test.txt")
    print("================================")
    # testcase()
```

附录 2：TF-IDF 值计算及 K-Means 聚类

```
# coding=utf-8
import  time
import  sys
import  codecs
import  random
from  sklearn  import  feature_extraction
from  sklearn.feature_extraction.text  import  TfidfTransformer
from  sklearn.feature_extraction.text  import  CountVectorizer
from  sklearn.cluster  import  KMeans
```

```
from sklearn.externals import joblib
reload(sys)
sys.setdefaultencoding('gbk')

if __name__ == "__main__":
    corpus = []
    for line in open('movie_top_jieba.txt', 'rb').readlines():
        corpus.append(line.strip())
    time.sleep(1)
    vectorizer = CountVectorizer() # 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 i 类文本下的词频
    transformer = TfidfTransformer() # 该类会统计每个词语的 tf-idf 权值
    tfidf = transformer.fit_transform(vectorizer.fit_transform(corpus))
    word = vectorizer.get_feature_names()
    weight = tfidf.toarray()
    resName = "movie_TF-IDF.txt"
    result = codecs.open(resName, 'a', 'utf-8')
    for j in range(len(word)):
        result.write(word[j] + ' ')
    result.write('\r\n\r\n')
    # 打印每类文本的 tf-idf 词语权重，第一个 for 遍历所有文本，第二个 for 便利某一类文本下的词语权重
    for i in range(len(weight)):
        for j in range(len(word)):
            result.write(str(weight[i][j]) + ' ')
        result.write('\r\n\r\n')
    result.close()
    # K-Means 聚类
    random.seed(999)
    print 'Start Kmeans:'
    clf = KMeans(n_clusters=8)
    s = clf.fit(weight)
    result = codecs.open('movie_kmeans.txt','a','utf8')
    result.writelines('调用 K-Means 聚类:'+ '\n' +str(s)+'\n')
    for each in clf.cluster_centers_:
        result.writelines(str(each)+'\n')
    result.writelines(str(clf.labels_)+'\n')
    result.writelines(str(clf.inertia_)+'\n')
    result.close()
    print '调用 K-Means 聚类:'
    print s
    print '聚类中心点：'
```

```
    print clf.cluster_centers_ # 8 个中心点
    print '样本所属的簇：'
    print clf.labels_   # 每个样本所属的簇
    print '点到对应簇的距离之和:'
    print clf.inertia_ # 用来评估簇的个数是否合适，距离越小说明簇分的越好，选取临界点的簇个数
```