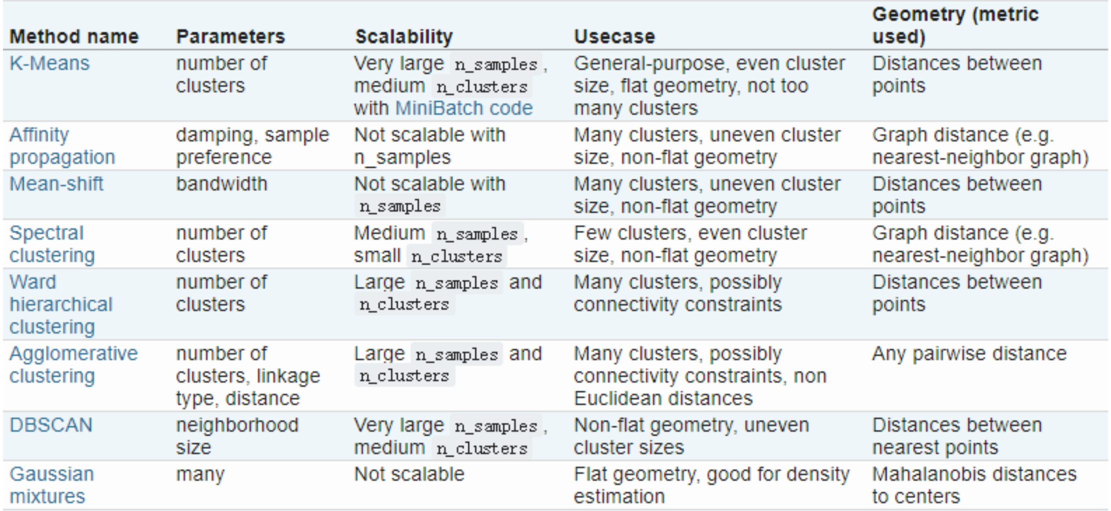
数据挖掘课程实验报告

Homework 3: Clustering

201834879　王士东　2018/12/19

一. 实验主要内容

测试 scikit-learn 中以下聚类算法在 Tweets 数据集上的聚类效果，使用 NMI(Normalized Mutual Information)作为评价指标。

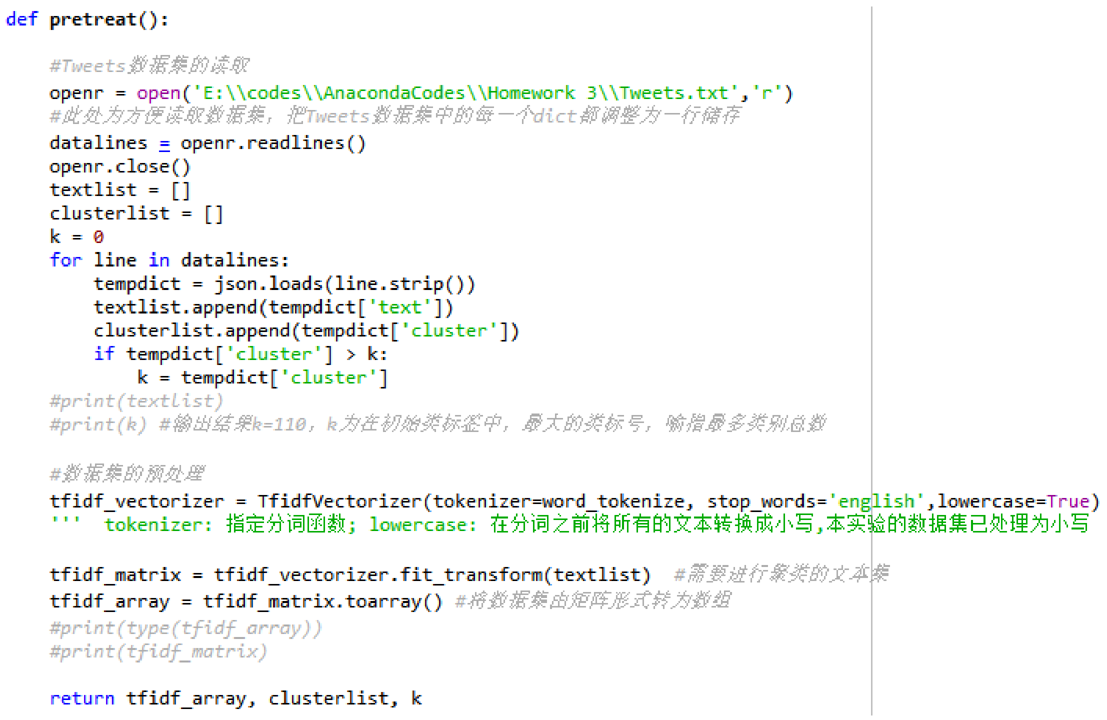


Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large n_samples, medium n_clusters with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium n_samples, small n_clusters	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large n_samples, medium n_clusters	Non-flat geometry, uneven cluster sizes	Distances between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers

二. 实验实现过程

1. 数据集的准备

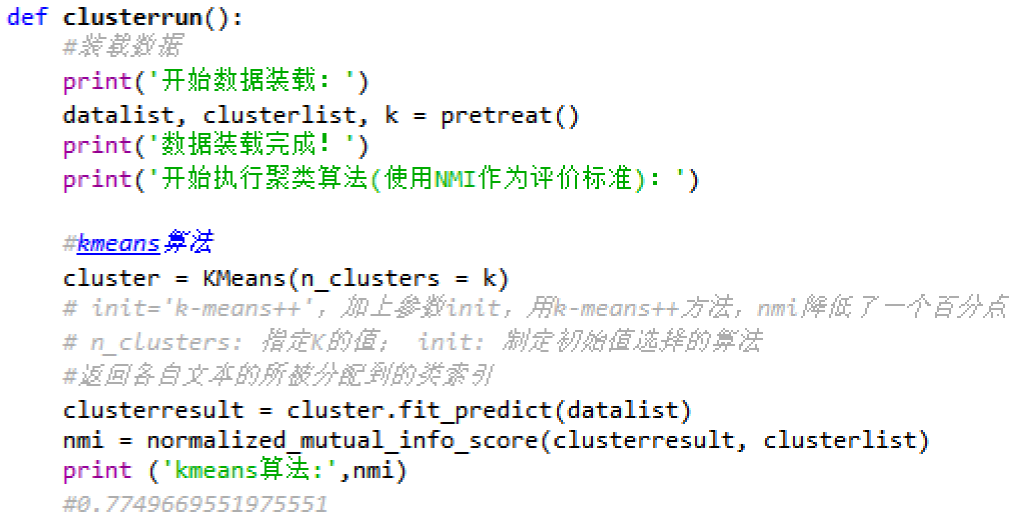
本实验所用数据集为 Tweets 数据集（为了使数据读取过程更易，本实验将数据集处理为一行存取一个字典），预处理过程可细分为以下三个部分。首先从数据集读取数据，并放到指定集合（text 文本集和初始 cluster 集）；然后使用分词函数（包括 jieba 分词函数和 nltk 分词函数）对数据集进行处理，并计算 tfidf 值；最后将计算得到的 tfidf 数组，初始类标签集和类别总数进行输出。代码实现如下：



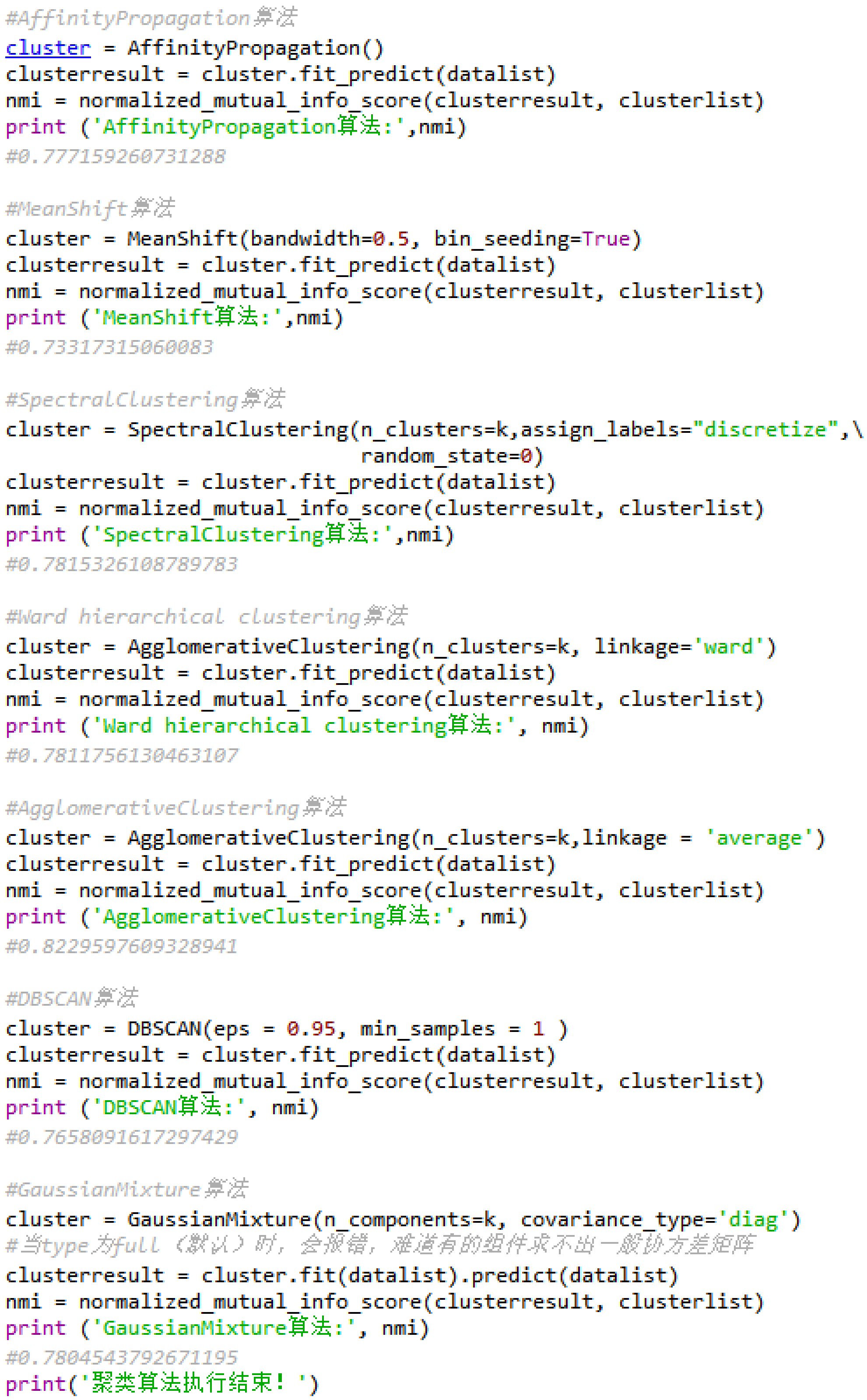
```python
def pretreat():

    #Tweets数据集的读取
    openr = open('E:\\codes\\AnacondaCodes\\Homework 3\\Tweets.txt','r')
    #此处为方便读取数据集，把Tweets数据集中的每一个dict都调整为一行储存
    datalines = openr.readlines()
    openr.close()
    textlist = []
    clusterlist = []
    k = 0
    for line in datalines:
        tempdict = json.loads(line.strip())
        textlist.append(tempdict['text'])
        clusterlist.append(tempdict['cluster'])
        if tempdict['cluster'] > k:
            k = tempdict['cluster']
    #print(textlist)
    #print(k) #输出结果k=110，k为在初始类标签中，最大的类标号，喻指最多类别总数

    #数据集的预处理
    tfidf_vectorizer = TfidfVectorizer(tokenizer=word_tokenize, stop_words='english',lowercase=True)
    '''  tokenizer: 指定分词函数; lowercase: 在分词之前将所有的文本转换成小写,本实验的数据集已处理为小写

    tfidf_matrix = tfidf_vectorizer.fit_transform(textlist)  #需要进行聚类的文本集
    tfidf_array = tfidf_matrix.toarray() #将数据集由矩阵形式转为数组
    #print(type(tfidf_array))
    #print(tfidf_matrix)

    return tfidf_array, clusterlist, k
```

2. sklearn 中各个聚类算法的实现

首先调用预处理函数，获得数据集向量，初始类标签和聚类数；然后调用 sklearn 中各个聚类函数对 Tweets 数据集进行聚类处理；最后比较各个算法在数据集上的聚类效果，使用 NMI 作为聚类效果的评价标准。代码实现如下：



```python
def clusterrun():
    #装载数据
    print('开始数据装载：')
    datalist, clusterlist, k = pretreat()
    print('数据装载完成！')
    print('开始执行聚类算法(使用NMI作为评价标准)：')

    #kmeans算法
    cluster = KMeans(n_clusters = k)
    # init='k-means++', 加上参数init, 用k-means++方法, nmi降低了一个百分点
    # n_clusters: 指定K的值;  init: 制定初始值选择的算法
    #返回各自文本的所被分配到的类索引
    clusterresult = cluster.fit_predict(datalist)
    nmi = normalized_mutual_info_score(clusterresult, clusterlist)
    print ('kmeans算法:',nmi)
    #0.7749669551975551
```

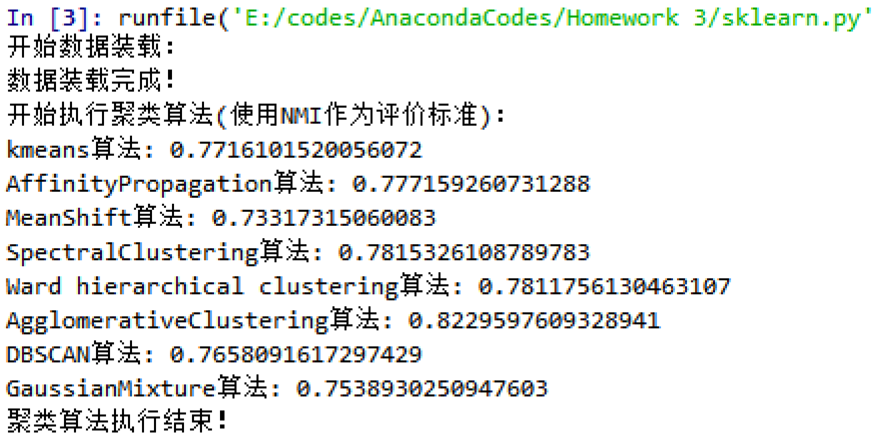


```
#AffinityPropagation算法
cluster = AffinityPropagation()
clusterresult = cluster.fit_predict(datalist)
nmi = normalized_mutual_info_score(clusterresult, clusterlist)
print ('AffinityPropagation算法:',nmi)
#0.777159260731288

#MeanShift算法
cluster = MeanShift(bandwidth=0.5, bin_seeding=True)
clusterresult = cluster.fit_predict(datalist)
nmi = normalized_mutual_info_score(clusterresult, clusterlist)
print ('MeanShift算法:',nmi)
#0.73317315060083

#SpectralClustering算法
cluster = SpectralClustering(n_clusters=k,assign_labels="discretize",\
                             random_state=0)
clusterresult = cluster.fit_predict(datalist)
nmi = normalized_mutual_info_score(clusterresult, clusterlist)
print ('SpectralClustering算法:',nmi)
#0.7815326108789783

#Ward hierarchical clustering算法
cluster = AgglomerativeClustering(n_clusters=k, linkage='ward')
clusterresult = cluster.fit_predict(datalist)
nmi = normalized_mutual_info_score(clusterresult, clusterlist)
print ('Ward hierarchical clustering算法:', nmi)
#0.7811756130463107

#AgglomerativeClustering算法
cluster = AgglomerativeClustering(n_clusters=k,linkage = 'average')
clusterresult = cluster.fit_predict(datalist)
nmi = normalized_mutual_info_score(clusterresult, clusterlist)
print ('AgglomerativeClustering算法:', nmi)
#0.8229597609328941

#DBSCAN算法
cluster = DBSCAN(eps = 0.95, min_samples = 1 )
clusterresult = cluster.fit_predict(datalist)
nmi = normalized_mutual_info_score(clusterresult, clusterlist)
print ('DBSCAN算法:', nmi)
#0.7658091617297429

#GaussianMixture算法
cluster = GaussianMixture(n_components=k, covariance_type='diag')
#当type为full（默认）时，会报错，难道有的组件求不出一般协方差矩阵
clusterresult = cluster.fit(datalist).predict(datalist)
nmi = normalized_mutual_info_score(clusterresult, clusterlist)
print ('GaussianMixture算法:', nmi)
#0.7804543792671195
print('聚类算法执行结束！')
```

3. 各个聚类算法的效果对比：

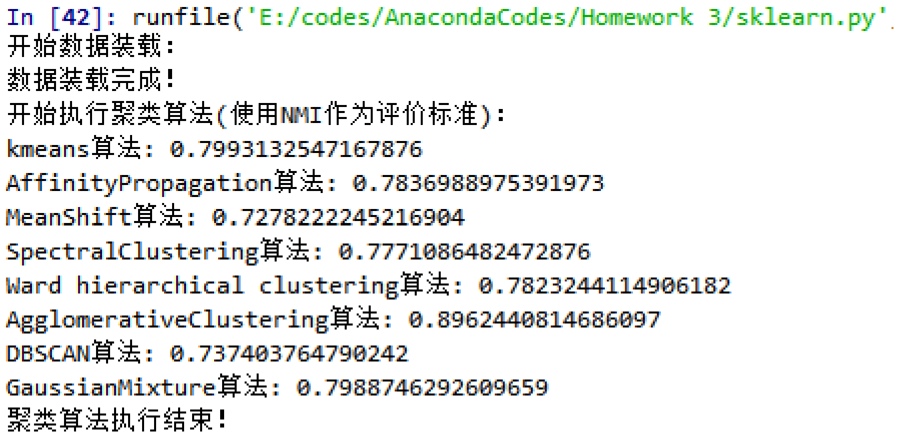
输出 jieba 分词的结果，发现其中含有大量的空格字符，于是又用 nltk 做了分词，试验结果表明使用 nltk 分词对于某些聚类算法会出现较好的结果。

1）使用 jieba 分词的聚类效果：



```
In [3]: runfile('E:/codes/AnacondaCodes/Homework 3/sklearn.py'
开始数据装载:
数据装载完成!
开始执行聚类算法(使用NMI作为评价标准):
kmeans算法: 0.7716101520056072
AffinityPropagation算法: 0.777159260731288
MeanShift算法: 0.73317315060083
SpectralClustering算法: 0.7815326108789783
Ward hierarchical clustering算法: 0.7811756130463107
AgglomerativeClustering算法: 0.8229597609328941
DBSCAN算法: 0.7658091617297429
GaussianMixture算法: 0.7538930250947603
聚类算法执行结束!
```

2）使用 nltk 分词的聚类效果：



```
In [42]: runfile('E:/codes/AnacondaCodes/Homework 3/sklearn.py'
开始数据装载:
数据装载完成!
开始执行聚类算法(使用NMI作为评价标准):
kmeans算法: 0.7993132547167876
AffinityPropagation算法: 0.7836988975391973
MeanShift算法: 0.7278222245216904
SpectralClustering算法: 0.7771086482472876
Ward hierarchical clustering算法: 0.7823244114906182
AgglomerativeClustering算法: 0.8962440814686097
DBSCAN算法: 0.737403764790242
GaussianMixture算法: 0.7988746292609659
聚类算法执行结束!
```

三. 总结

实验前期主要是结合课件内容和网络资源对各个聚类算法的原理进行了系统的学习，然后是熟悉了在 python 中 sklearn 的使用，最后是进行代码编写，对各个聚类算法进行调用，并对各个算法在 Tweets 数据集上的聚类效果进行比较。实验之中也遇到了各种各样的问题，比如 sklearn 工具的首次执行必须在 c 盘下，数据要由矩阵形式转为数组形式，K 值的选取问题和不同的分词方式对聚类效果的影响等等。最后，感谢老师在这一学期对于我们的指导和帮助，最好的祝愿给您！