山东大学 计算机科学与技术 学院

信息检索与数据挖掘 课程实验报告

学号：201600301304	姓名：贾乘兴	班级：人工智能 16
实验题目：聚类方法与评估		

实验内容：

调用 sklearn 的方法，实现 KMeans,AffinityPropagation,MeanShift,SpectralClustering,AgglomerativeClustering,DBSCAN，高丝混合等方法，并用 nmi 进行评估

结果如下

```
load the data...

method meanshift
compute the result...
NMI:0.65015

method DBSCAN
compute the result...
NMI:0.707871

method k_means
compute the result...
NMI:0.798497

method Spectral
compute the result...
NMI:0.681795

method ward
compute the result...
NMI:0.67917

method Agglomerative
compute the result...
NMI:0.908023

method AP
compute the result...
NMI:0.782623

method GaussianMixture
```

```
compute the result...
NMI:0.813259

from sklearn.feature_extraction.text import
TfidfTransformer,CountVectorizer
from sklearn.cluster import
KMeans,AffinityPropagation,MeanShift,SpectralClustering,AgglomerativeClust
ering,DBSCAN
from sklearn.mixture import GMM
from sklearn.metrics.cluster import normalized_mutual_info_score as NMI
import sklearn
import json
import nltk
import re

# 去除停用词
def cutstopwords(str):
    stopwords = {}.fromkeys([line.rstrip() for line in
open('estopwords.txt')])
    segs = str.replace('\n','').lower().split(' ')
    new_str = ''
    for seg in segs:
        if seg not in stopwords:
            new_str = new_str + " " +seg
    return new_str

# 去除标点
def cutsyms(str):
    new_str = re.sub('[,.\'\"\t\n*_+=?/|!@#$%^&*()`~<>:;\-\[\]]'," ",str)
    return new_str

# 词干提取
def stemming(str):
    s = nltk.stem.SnowballStemmer('english')
    segs = str.replace('\n', '').lower().split(' ')
    new_str = ''
    for seg in segs:
        new_str = new_str + " " + s.stem(seg)
    return new_str

path = '/Users/apple/Desktop/ir/hw5/Homework5Tweets.txt'
file = open(path,'r',encoding='UTF-8',errors='ignore')
tweets = []
cluster = []
```

```
text = []
count = 0
print("load the data...\n")
for line in file:
    tweets.append(json.loads(line))
    cluster.append(tweets[count]['cluster'])
    tweets[count]['text'] = tweets[count]['text'].lower()
    tweets[count]['text'] = cutsyms(tweets[count]['text'])
    tweets[count]['text'] = cutstopwords(tweets[count]['text'])
    tweets[count]['text'] = stemming(tweets[count]['text'])
    text.append(tweets[count]['text'])
    count = count + 1
file.close()
tv = TfidfTransformer()
vec = CountVectorizer()
mv = tv.fit_transform(vec.fit_transform(text))
tm = mv.toarray()

def showresult(result,cluster):
    print("show the result:")
    for i in range(len(result)):
        print("cluster of text %d:%d. and its truecluster:%d" % (i + 1,
result[i], cluster[i]))


print("method meanshift")
ms = MeanShift(bandwidth=0.4,bin_seeding=True,min_bin_freq=2)
print("compute the result...")
result = ms.fit_predict(tm)
#showresult(result,cluster)
nmi=NMI(cluster,result)
print("NMI:%g\n"%(nmi))

print("method DBSCAN")
DB = DBSCAN(eps=0.7,min_samples=1)
print("compute the result...")
result = DB.fit_predict(tm)
#showresult(result,cluster)
nmi=NMI(cluster,result)
print("NMI:%g\n"%(nmi))

print("method k_means")
num_cluster = 150
km = KMeans(n_clusters=num_cluster, max_iter=300, n_init=40, init='k-
```

```
means++')
print("compute the result...")
result = km.fit_predict(mv)
#showresult(result,cluster)
nmi=NMI(cluster,result)
print("NMI:%g\n"%(nmi))

print("method Spectral")
sc = SpectralClustering(n_clusters=max(cluster))
print("compute the result...")
result = sc.fit_predict(tm)
#showresult(result,cluster)
nmi=NMI(cluster,result)
print("NMI:%g\n"%(nmi))

print("method ward")
wh = SpectralClustering(n_clusters=max(cluster))
print("compute the result...")
result = wh.fit_predict(tm)
#showresult(result,cluster)
nmi=NMI(cluster,result)
print("NMI:%g\n"%(nmi))

print("method Agglomerative")
ac = AgglomerativeClustering(n_clusters=max(cluster),linkage='average')
print("compute the result...")
result = ac.fit_predict(tm)
#showresult(result,cluster)
nmi=NMI(cluster,result)
print("NMI:%g\n"%(nmi))

print("method AP")
ap = AffinityPropagation()
print("compute the result...")
result = ap.fit_predict(mv)
#showresult(result,cluster)
nmi=NMI(cluster,result)
print("NMI:%g\n"%(nmi))

print("method GaussianMixture")
GM = GMM(n_components=150)
print("compute the result...")
result = GM.fit_predict(tm)
#showresult(result,cluster)
```

```
nmi=NMI(cluster,result)
print("NMI:%g\n"%(nmi))
```

结论分析与体会：