实验五

基于K-means的AAAI论文聚类分析

2022年7月27日

1. 读取数据

```
import pandas as pd
```

```
df = pd.read_csv('./data/[UCI] AAAI-14 Accepted Papers - Papers.csv').astype(str)
df.head()
```

	title	authors	groups	keywords	topics	abstract
0	Kernelized Bayesian Transfer Learning	Mehmet Gönen and Adam A. Margolin	Novel Machine Learning Algorithms (NMLA)	cross-domain learning\ndomain adaptation\nkern...	APP: Biomedical / Bioinformatics\nNMLA: Bayesi...	Transfer learning considers related but distin...
1	"Source Free" Transfer Learning for Text Class...	Zhongqi Lu, Yin Zhu, Sinno Pan, Evan Xiang, Yu...	AI and the Web (AIW)\nNovel Machine Learning A...	Transfer Learning\nAuxiliary Data Retrieval\nT...	AIW: Knowledge acquisition from the web\nAIW: ...	Transfer learning uses relevant auxiliary data...
2	A Generalization of Probabilistic Serial to Ra...	Haris Aziz and Paul Stursberg	Game Theory and Economic Paradigms (GTEP)	social choice theory\nvoting\nfair division\ns...	GTEP: Game Theory\nGTEP: Social Choice / Voting	The probabilistic serial (PS) rule is one of t...
3	Lifetime Lexical Variation in Social Media	Liao Lizi, Jing Jiang, Ying Ding, Heyan Huang ...	NLP and Text Mining (NLPTM)	Generative model\nSocial Networks\nAge Prediction	AIW: Web personalization and user modeling\nNL...	As the rapid growth of online social media att...
4	Hybrid Singular Value Thresholding for Tensor ...	Xiaoqin Zhang, Zhengyuan Zhou, Di Wang and Yi Ma	Knowledge Representation and Reasoning (KRR)\n...	tensor completion\nlow-rank recovery\nhybrid s...	KRR: Knowledge Representation (General/Other)\...	In this paper, we study the low-rank tensor co...

数据包括标题、作者、关键词、摘要在内的文本信息，需要转化为向量

2. 创建文本特征

2.1 分词

使用nltk分词，并提取词干，去掉文本中的数字、标点符号等

```
import re
import nltk
from nltk.corpus import stopwords
en_stops = set(stopwords.words('english'))
columns = list(df.columns)
print(columns)
def create_feature(x):
    words = []
    for col in columns:
        words.extend(nltk.word_tokenize(x[col].strip()))
    filter_words = []
    porter = nltk.PorterStemmer()
    for word in words:
        if word.isdigit() == False and re.search("\W", word) == None and word not in en_stops:
            filter_words.append(porter.stem(word))
    return " ".join(filter_words)
['title', 'authors', 'groups', 'keywords', 'topics', 'abstract']
```

分词结果如下所示：

```python
df['feature'] = df.apply(create_feature, axis=1)
```

```python
df['feature']
```

```
0      kernel bayesian transfer learn mehmet gönen ad...
1      sourc free transfer learn text classif zhongqi...
2      a gener probabilist serial random social choic...
3      lifetim lexic variat social media liao lizi ji...
4      hybrid singular valu threshold tensor complet ...
                             ...
393    map user across network manifold align hypergr...
394    compact aspect embed for diversifi queri expan...
395    contract revis tbox zhiqiang zhuang zhe wang k...
396    zero pronoun resolut rank chen chen vincent ng...
397    supervis transfer spars code maruan jim wang m...
Name: feature, Length: 398, dtype: object
```

2.2 生成词袋模型

```python
from sklearn.feature_extraction.text import TfidfVectorizer
cv = TfidfVectorizer(stop_words=en_stops, max_features=100)
cv.fit(df['feature'])
```

```
TfidfVectorizer(max_features=100,
                stop_words={'a', 'about', 'above', 'after', 'again', 'against',
                            'ain', 'all', 'am', 'an', 'and', 'any', 'are',
                            'aren', "aren't", 'as', 'at', 'be', 'because',
                            'been', 'before', 'being', 'below', 'between',
                            'both', 'but', 'by', 'can', 'couldn', "couldn't", ...})
```

```python
cv.vocabulary_
```

```
{'novel': 55,
 'machin': 46,
 'nmla': 54,
 'domain': 22,
 'adapt': 0,
 'method': 49,
 'app': 6,
 'object': 57,
 'relat': 72,
 'task': 90,
 'knowledg': 42,
 'improv': 39,
 'gener': 33,
 'perform': 61,
 'label': 44,
 'data': 16,
 'effect': 24,
 'framework': 30,
 'classif': 12,
```

```
data = cv.transform(df['feature'])
data.toarray().shape
```

```
(398, 100)
```

3. K-Means聚类

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=10, random_state=2022)
kmeans.fit(data)
```

```
KMeans(n_clusters=10, random_state=2022)
```

```
kmeans.labels_
```

```
array([9, 9, 7, 7, 5, 2, 2, 5, 2, 3, 8, 5, 9, 7, 6, 9, 2, 2, 3, 3, 2, 4,
       2, 5, 6, 0, 2, 8, 7, 7, 2, 8, 9, 9, 1, 4, 9, 1, 9, 1, 5, 4, 9, 5,
       2, 4, 5, 2, 9, 8, 2, 5, 9, 0, 5, 9, 4, 1, 2, 5, 1, 5, 2, 9, 7, 2,
       4, 2, 2, 4, 2, 4, 5, 5, 3, 9, 4, 1, 2, 2, 2, 2, 7, 3, 2, 2, 5, 2,
       9, 5, 5, 9, 6, 2, 5, 8, 3, 1, 9, 2, 4, 7, 5, 9, 2, 8, 1, 6, 9, 4,
       1, 8, 9, 4, 2, 8, 2, 1, 5, 5, 2, 5, 2, 6, 1, 9, 5, 6, 9, 6, 5, 4,
       2, 5, 1, 2, 9, 3, 6, 3, 4, 5, 2, 9, 3, 9, 8, 5, 6, 1, 9, 3, 2, 2,
       5, 8, 7, 6, 3, 2, 9, 1, 4, 5, 8, 2, 1, 4, 6, 2, 4, 2, 6, 4, 4, 5,
       7, 8, 7, 2, 4, 2, 9, 2, 4, 8, 0, 2, 1, 1, 1, 2, 8, 5, 0, 2, 6, 2,
       2, 5, 2, 2, 0, 2, 2, 4, 6, 2, 3, 0, 5, 5, 2, 2, 0, 2, 6, 5, 6, 3,
       9, 9, 9, 2, 2, 5, 0, 9, 2, 5, 2, 2, 5, 2, 7, 5, 2, 2, 9, 2, 2, 2,
       5, 5, 7, 1, 9, 8, 5, 5, 8, 3, 1, 5, 5, 7, 6, 9, 5, 9, 9, 1, 7, 2,
       5, 1, 8, 9, 7, 7, 2, 5, 2, 5, 5, 5, 3, 8, 9, 4, 0, 6, 9, 8, 2, 9,
       7, 9, 5, 2, 2, 2, 8, 5, 2, 1, 2, 9, 9, 3, 5, 2, 5, 2, 1, 9, 9, 2,
       4, 9, 5, 4, 6, 1, 8, 6, 5, 2, 9, 8, 1, 4, 9, 2, 4, 1, 8, 5, 9, 3,
       2, 9, 5, 2, 8, 9, 1, 1, 8, 9, 2, 2, 1, 2, 2, 2, 6, 1, 4, 2, 9, 9,
       4, 5, 4, 5, 4, 2, 5, 1, 9, 2, 2, 9, 5, 9, 2, 2, 5, 2, 1, 8, 9, 9,
       4, 2, 2, 6, 9, 2, 1, 0, 2, 4, 2, 0, 8, 8, 0, 8, 1, 6, 5, 7, 2, 4,
       2, 3], dtype=int32)
```

可以用kmeans.cluster_centers_查看聚类中心

```
array([[ 0.7962963 ,  0.94444444,  1.12962963,  1.87037037,  1.37037037,
         3.38888889,  0.55555556,  0.        ,  0.7037037 ,  0.03703704],
       [ 0.32484076,  0.47770701,  0.26751592,  0.17197452,  0.28025478,
         0.12738854,  0.17197452,  0.13375796,  0.43949045,  0.24203822],
       [ 0.4       ,  0.73333333,  0.        ,  0.33333333,  0.2       ,
         0.        ,  2.06666667,  1.2       ,  1.06666667,  8.4       ],
       [ 0.21052632,  1.15789474,  0.10526316,  0.10526316,  0.31578947,
         0.10526316,  0.68421053,  7.68421053,  2.36842105,  0.52631579],
       [ 0.5       ,  0.83333333,  0.9       ,  0.7       ,  4.16666667,
         0.86666667,  0.56666667,  0.        ,  0.86666667,  0.43333333],
       [ 0.5       ,  0.5       ,  0.22222222,  0.55555556,  0.5       ,
         0.72222222,  4.22222222,  1.27777778,  1.44444444,  0.88888889],
       [ 0.5       ,  0.5       ,  0.5       ,  0.        ,  0.        ,
         0.        ,  4.5       , 21.5       ,  5.5       ,  0.5       ],
       [ 3.02631579,  0.84210526,  0.89473684,  2.86842105,  0.89473684,
         0.81578947,  0.44736842,  0.02631579,  0.18421053,  0.31578947],
       [ 1.06666667,  0.63333333,  6.26666667,  1.53333333,  1.46666667,
         1.8       ,  0.56666667,  0.16666667,  1.06666667,  0.6       ],
       [ 0.4       ,  0.6       ,  0.14285714,  0.28571429,  0.45714286,
         0.31428571,  0.77142857,  0.45714286,  4.22857143,  0.62857143]])
```

4. 评价聚类结果

使用轮廓系数评价聚类结果，轮廓系数越接近1则聚类效果越好。实验中的结果只为0.03，并不是很好。

```
from sklearn import metrics
import numpy as np
metrics.silhouette_score(data, kmeans.labels_, metric='euclidean')
```

```
0.033202975690121664
```

提取这几个聚类结果的关键字可以看出：

```
df['label'] = kmeans.labels_
label = df.groupby('label')
```

```
for k, v in label.groups.items():
    print("group: %d" % k)
    print(cv.get_feature_names_out()[np.array(data[v].todense().sum(axis=0))[0].argsort()[-2:]])
```

```
group: 0
['constraint' 'search']
group: 1
['reason' 'krr']
group: 2
['gtep' 'game']
group: 3
['ps' 'plan']
group: 4
['nmla' 'learn']
group: 5
['social' 'network']
group: 6
['aiw' 'web']
group: 7
['learn' 'method']
group: 8
['learn' 'imag']
group: 9
['design' 'mechan']
```

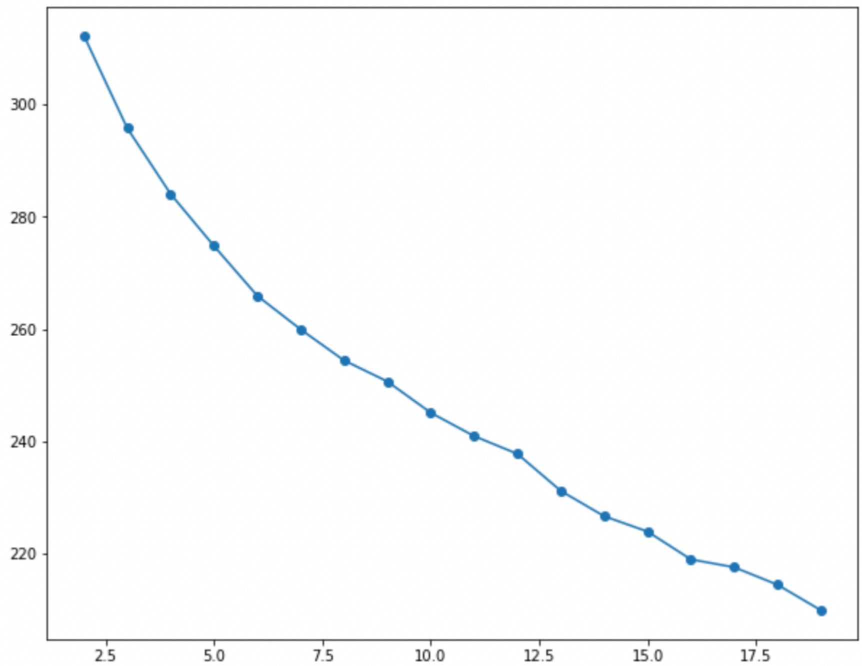
聚类一主要是search相关，聚类二是krr相关，聚类三是game相关，聚类四是plan相关， 聚类五是nmla相关，聚类六是social network相关，聚类七是web相关，聚类八是method相关，聚类九是image相关，聚类十是mechain design相关

5. 寻找K值

使用Elbow Method寻找最好的k值

```python
import matplotlib.pyplot as plt
wcss = []
for k in range(2, 20):
    clf = KMeans(n_clusters=k, random_state=2022).fit(data)
    wcss.append(clf.inertia_)
plt.figure(figsize = (10, 8))
plt.plot(range(2, 20), wcss, marker='o')
```

```
[<matplotlib.lines.Line2D at 0x11f9a5eb0>]
```



可以看出肘部的位置大概在k=6

按k=6使用kmeans聚类，可以看出聚类一是image相关，聚类二是data相关，聚类三是search相关，聚类四是game相关，聚类五是behavior相关，聚类六是robot相关。

```python
kmeans = KMeans(n_clusters=6, random_state=2022)
kmeans.fit(data)
df['label'] = kmeans.labels_
label = df.groupby('label')
for k, v in label.groups.items():
    print("group: %d" % k)
    print(cv.get_feature_names_out()[np.array(data[v].todense().max(axis=0))[0].argsort()[-2:]])
```

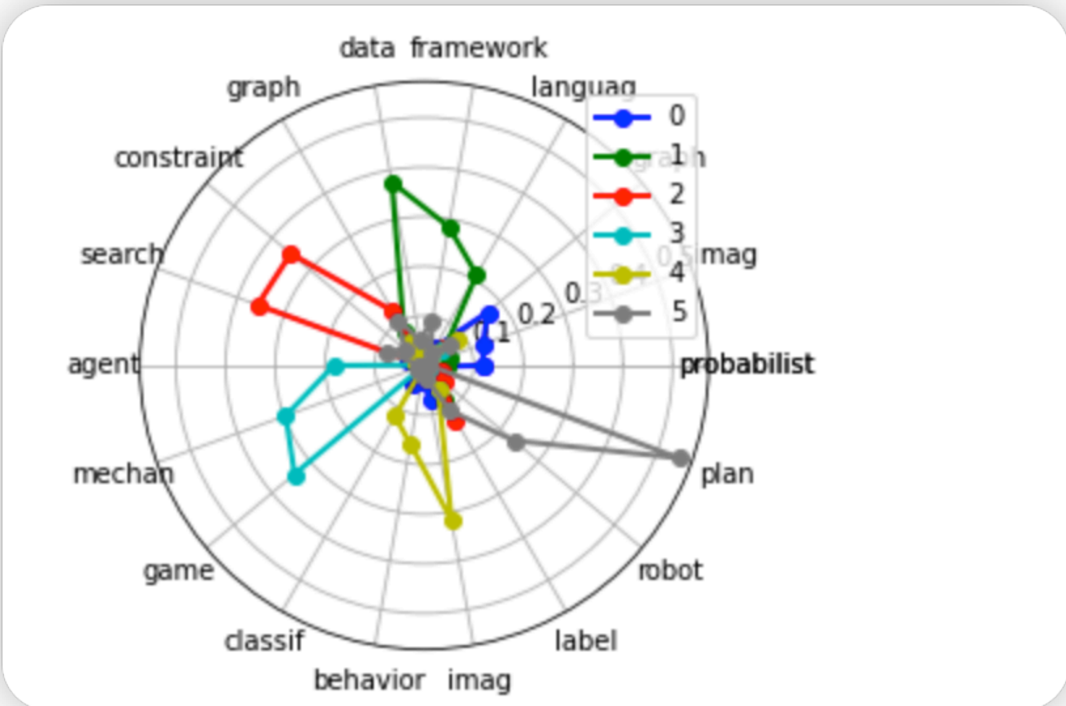
```
group: 0
['imag' 'graph']
group: 1
['framework' 'data']
group: 2
['constraint' 'search']
group: 3
['mechan' 'game']
group: 4
['behavior' 'imag']
group: 5
['robot' 'plan']
```

提取其中关键字，做雷达图分析：

```python
imp_features = []
imp_idx = []
for k, v in label.groups.items():
    imp_features.extend(cv.get_feature_names_out()[np.array(data[v].todense().max(axis=0))[0].argsort()[-3:]])
    imp_idx.extend(np.array(data[v].todense().sum(axis=0))[0].argsort()[-3:])
#标签
labels = imp_features
#每个类别中心点数据
plot_data = kmeans.cluster_centers_[:,imp_idx]
#指定颜色
color = ['b', 'g', 'r', 'c', 'y', 'grey']
# 设置角度
angles = np.linspace(0, 2*np.pi, len(plot_data[0]), endpoint=False)
# 闭合
angles = np.concatenate((angles, [angles[0]]))
plot_data = np.concatenate((plot_data, plot_data[:,[0]]), axis=1)

fig = plt.figure()
ax = fig.add_subplot(111, polar=True) # polar参数为True即极坐标系
for i in range(len(plot_data)):
    ax.plot(angles, plot_data[i], 'o-', color = color[i], label = i, linewidth=2)

ax.set_thetagrids(angles * 180/np.pi, np.append(labels, labels[0]))
plt.legend()
plt.show()
```



可以看出基本上各个类别的特征都比较明显

6. 实验总结

本次实验用用 K-Means 的方法进行聚类分析，比较了不同k值，对聚类结果的影响，学习了肘部法则寻找最佳k值的方法，最后用雷达图展现聚类结果。