In [ ]:

```

```

基于TF-IDF的文本分类和聚类

TF-IDF文本向量化

In [1]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np

corpus = [
        'This is the first document.',
        'This document is the second document.',
        'Beijing is our capital.',
        'Is this the first document?',
        'Beijing is the capital of China.',

]


vectorizer = TfidfVectorizer(max_features=10)
X = vectorizer.fit_transform(corpus).toarray()


print(vectorizer. get_feature_names())
print('\n')
print('tf-idf向量化的结果为：\n', X)
```

```
['beijing', 'capital', 'china', 'document', 'first', 'is', 'of', 'our', 'the', 'this']


tf-idf向量化的结果为：
 [[0.         0.         0.         0.4629834  0.55775063 0.32941651
  0.         0.         0.38947624 0.4629834 ]
 [0.         0.         0.         0.80231952 0.         0.28542847
  0.         0.         0.33746824 0.40115976]
 [0.50733821 0.50733821 0.         0.         0.         0.29964212
  0.         0.62883263 0.         0.        ]
 [0.         0.         0.         0.4629834  0.55775063 0.32941651
  0.         0.         0.38947624 0.4629834 ]
 [0.41137843 0.41137843 0.50989296 0.         0.         0.24296673
  0.50989296 0.         0.2872648  0.        ]]
```

基于TF-IDF的文本分类

In [3]:

```
Y=np.array([0,0,1,0,1])
#print('对应的类别标签为',Y)


from sklearn import svm

X_train=X[:X.shape[0]-1]
#print(X.shape[0]-1)
Y_train=Y[:Y.shape[0]-1]

#'Beijing is the capital of China.',
X_test=X[X.shape[0]-1:]

clf =svm.SVC(kernel='linear', probability=True)
```

```
clf.fit(X_train,Y_train)
print(clf.predict(X_test))
```

```
[1]
```

基于TF-IDF的文本聚类

In [4]:

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2).fit(X)

print(kmeans.predict(X))
print(kmeans.cluster_centers_)
```

```
[0 0 1 0 1]
[[0.00000000e+00 0.00000000e+00 1.38777878e-17 5.76095444e-01
  3.71833752e-01 3.14753829e-01 1.38777878e-17 2.77555756e-17
  3.72140237e-01 4.42375524e-01]
 [4.59358322e-01 4.59358322e-01 2.54946482e-01 0.00000000e+00
  0.00000000e+00 2.71304425e-01 2.54946482e-01 3.14416317e-01
  1.43632401e-01 0.00000000e+00]]
```

基于doc2vec的文本分类和聚类

读取训练数据，并将读取的句子存储到text中（同chapter 7代码）

In [5]:

```
import json

text=[]

f_read=open('./data/体育.json', 'r',encoding='utf8',errors='ignore')

for line in f_read:
    line=line.replace('\\u0009','').replace('\\n','')
    obj=json.loads(line)
    sent=obj['contentClean']
    text.append(sent)
print(len(text))
```

```
500
```

In [6]:

```
import jieba
processed_text=[]

for sent in text:
    processed_sent=jieba.cut(sent.strip(' '))
    processed_text.append(list(processed_sent))

print(processed_text[0])
```

```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\fengl\AppData\Local\Temp\jieba.cache
Loading model cost 0.486 seconds.
Prefix dict has been built successfully.
['远', '在', '土耳其', '打球', '的', '朱婷', '迎来', '自己', '的', '大', '日子', '，',
'今年', '11', '月', '29', '日', '是', '这位', '中国女排', '当家', '球星', '的', '22',
'岁', '生日', '。', '尽管', '在', '国外', '，', '但', '朱婷', '还是', '感受', '到',
'了', '家乡', '的', '温暖', '，', '因为', '29', '日', '她', '有', '一场', '特别',
'的', '生日会', '，', '腾讯', '体育', '也', '对', '这场', '生日会', '进行', '了', '全
程', '直播', '。', '　', '　', '　', '　', '郎导', '携', '女儿', '录像', '送祝福', '，',
'黄晓明', '成', '意外', '惊喜', '　', '　', '当地', '时间', '13', '点', '30', '分',
'，', '朱婷', '的', '生日会', '正式', '开始', '。', '作为', '当天', '的', '绝对', '主
角', '，', '朱婷', '结束', '了', '上午', '的', '训练', '匆匆', '赶来', '。', '她', '身
穿', '运动服', '刚', '进入', '会场', '，', '参加', '生日会', '的', '球迷', '和', '记
者', '就', '颇', '有', '默契', '地', '一起', '为', '朱婷', '高唱', '生日快乐', '，',
'现场', '其乐融融', '。', '谦逊', '的', '朱婷', '开口', '第一句', '就是', '感谢',
'，', '“', '这是', '第一次', '在', '国外', '过生日', '，', '很', '感动', '，', '大
家', '特意', '从', '国内', '飞过来', '，', '让', '我', '在', '海外', '也', '有家',
'的', '感觉', '。', '以后', '我要', '以', '更好', '的', '训练', '和', '比赛', '来',
'回报', '大家', '。', '”', '　', '　', '朱婷', '来到', '土耳其', '比赛', '之后', '，',
'赞助商', '还', '特别', '为', '她', '配备', '了', '一名', '厨师', '随行', '，', '朱
婷', '时不时', '就', '会', '在', '深夜', '“', '放毒', '”', '，', '上传', '各种', '美
食', '照', '。', '朱婷', '辩称', '她', '可不是', '在', '炫耀', '，', '而是', '另有',
'目的', '：', '“', '我', '之所以', '每天', '深夜', '发吃', '的', '，', '是', '希望',
'关心', '我', '的', '朋友', '知道', '，', '我', '在', '土耳其', '生活', '得', '很',
'好', '，', '后盾', '很', '坚强', '。', '”', '　', '　', '虽然', '朱婷', '在', '国内',
'的', '队友', '和', '教练', '不能', '到', '现场', '庆祝', '，', '但', '他们', '还是',
'用', '录像', '视频', '的', '方式', '送上', '了', '祝福', '。', '张', '常宁', '、',
'惠若琪', '、', '龚', '翔宇', '、', '沈静', '思', '、', '单丹娜', '等', '人', '一一',
'出现', '在', '现场', '大屏幕', '上', '，', '朱婷', '恩师', '、', '带领', '中国女排',
'在', '里', '约', '夺冠', '的', '郎平', '教练', '还', '特别', '带', '着', '女儿', '一
起', '为', '爱徒', '录制', '了', '生日歌', '，', '两人', '一边', '唱', '一边', '不',
'忘', '活泼', '搞怪', '，', '看', '得', '原本', '眼眶', '含泪', '的', '朱婷', '破涕为
笑', '。', '　', '　', '　', '　', '郎导', '唱', '完', '生日歌', '后', '，', '视频', '并',
'没有', '就此结束', '，', '生日会', '还给', '了', '朱婷', '一个', '惊喜', '：', '她',
'的', '偶像', '黄晓明', '也', '通过', '大屏幕', '视频', '祝贺', '朱婷', '生日快乐',
'，', '黄', '教主', '还', '将', '朱婷', '称呼', '为', '“', '全民', '女神', '”',
'。', '看到', '偶像', '出现', '，', '22', '岁', '的', '朱婷', '立马', '成', '了', '小
女生', '，', '直接', '爆发', '出', '一声', '尖叫', '：', '“', '啊', '！', '”', '朱
婷', '脸上', '显露出', '害羞', '的', '神情', '，', '她', '还', '清楚', '地', '记得',
'这是', '黄晓明', '第二次', '为', '她', '录制', '视频', '：', '“', '第一次', '是',
'在', '一个', '节目', '里', '，', '（', '看到', '黄晓明', '的', '生日', '祝福', '）',
'，', '这次', '生日会', '本来', '已经', '很', '圆满', '，', '但是', '现在', '更', '圆
满', '了', '。', '”', '　', '　', '土耳其', '粉丝', '送礼物', '，', '朱婷', '贴心',
'为', '他', '煮', '面', '　', '　', '朱婷', '是', '来自', '河南', '的', '姑娘', '，',
'所以', '今天', '的', '长寿面', '也', '很', '特别', '，', '厨师', '特别', '准备',
'了', '河南', '烩面', '。', '看到', '家乡', '的', '美食', '，', '朱婷', '不', '等',
'小料', '上', '齐', '，', '拿', '起', '筷子', '就', '吃', '起来', '。', '嗖嗖', '几
口', '下肚', '，', '朱婷', '吃', '得', '一脸', '满足', '。', '“', '生日', '吃', '长寿
面', '非常', '有', '意义', '，', '能', '吃', '到', '河南', '烩面', '更', '有', '家乡',
'情', '。', '”', '朱婷', '表示', '。', '　', '　', '朱婷', '也', '不是', '光', '吃',
'不', '做', '，', '现场', '有', '媒体', '起哄', '让', '朱婷', '给', '大家', '煮面',
'，', '没想到', '她', '直爽', '地', '一口', '答应下来', '：', '“', '（', '你们',
'）', '愿意', '吃', '吗', '？', '好', '！', '”', '说完', '朱婷', '就', '在', '厨师',
'的', '指导', '下', '，', '将', '面条', '丢进', '锅里', '的', '老汤', '煮起来', '。',
'要', '知道', '这锅', '老汤', '可不', '一般', '，', '是', '厨师', '用', '中国', '空
运', '过来', '的', '羊肉', '，', '加上', '鸡', '、', '鸭', '和', '鲫鱼', '熬制', '了',
'一', '晚上', '做成', '的', '。', '　', '　', '那么', '谁', '能', '成为', '吃', '到',
'朱婷', '这', '碗面', '的', '幸运儿', '呢', '？', '答案', '是', '土耳其', '的', '一
位', '球迷', '。', '作为', '中国女排', '的', '当家', '球星', '，', '朱婷', '也', '吸
引', '了', '众多', '国外', '球迷', '的', '关注', '。', '今天', '的', '生日会', '，',
'一位', '土耳其', '的', '球迷', '就', '特意', '为', '朱婷', '准备', '了', '礼物',
'，', '来到', '现场', '送上', '。', '朱婷', '除了', '和', '他', '合影留念', '，',
'还', '将', '自己', '煮', '的', '面送', '上', '，', '贴心', '为', '这位', '球迷', '往
面', '里加', '小料', '。', '　', '　', '生日会', '的', '最后', '，', '自然', '是', '切',
'蛋糕', '的', '环节', '。', '一个', '大', '蛋糕', '缓缓', '推入', '会场', '，', '准
备', '的', '生日', '刀', '居然', '是', '一把', '武士刀', '，', '朱婷', '看到', '先是',
'一', '愣', '，', '随后', '还', '俏皮', '地', '拿', '着', '武士刀', '摆起', 'POSE',
'。', '最终', '全场', '又', '一次', '齐声高唱', '生日歌', '，', '朱婷切', '下', '蛋
糕', '，', '生日会', '画上', '完满', '的', '句号', '。']
```

基于doc2vec的文本向量化

In [ ]:

```
import gensim
from gensim.models.doc2vec import Doc2Vec,LabeledSentence
```

```
# 生成固定格式的训练文档集合

train_text=[]

for i,sent in enumerate(processed_text):
    #改变成Doc2vec所需要的输入样本格式,
    #由于gensim里Doc2vec模型需要的输入为固定格式，输入样本为：[句子，句子序号],这里需
    tagged_doc=gensim.models.doc2vec.TaggedDocument(sent,tags=[i])
    train_text.append(tagged_doc)
    #print(tagged_doc)

d_model=Doc2Vec(train_text,min_count=5,windows=3,vector_size=100,sample=0.001,nagetiv



d_model.train(train_text,total_examples=d_model.corpus_count,epochs=10)
# 保存模型，以便重用
d_model.save("doc2vec_model")  #保存模型
```

In [7]:

```
import gensim
from gensim.models.doc2vec import Doc2Vec
#load doc2vec model...
d_model= gensim.models.doc2vec.Doc2Vec.load("doc2vec_model")
#load train vectors...
text_vecs= d_model.docvecs.vectors_docs
print("专利向量的个数为",len(text_vecs))
#print(text_vecs[0])
```

```
专利向量的个数为 500
```

基于doc2vec的文本分类

In [8]:

```
import numpy as np
import jieba

#假设的标签集合为
Y2=np.random.randint(0,2,size=500)
#print(Y2)

from sklearn import svm
clf =svm.SVC(kernel='linear', probability=True)
clf.fit(text_vecs,Y2)


new_text='我们是中国人'

new_tokens=jieba.cut(new_text)

v1 = d_model.infer_vector(new_tokens)

print(v1)




print(clf.predict([v1]))
```

```
[ 0.0002241   0.00446998  0.00029483  0.00075348 -0.0047614  -0.00268221
 -0.00205908  0.00036671  0.00079796  0.00314765 -0.00170187 -0.00293674
  0.00146427  0.00261809  0.00241462  0.00284832  0.00096851  0.00360352
  0.00353645 -0.00201437 -0.00134309  0.00019463  0.0014989  -0.0024608
 -0.00023513  0.00483441 -0.00113317  0.00300425  0.00245507  0.00356231
 -0.00365815 -0.00191115  0.00388785 -0.00195725  0.00034981 -0.00367536
  0.00131385  0.00352633 -0.00457557  0.00229518  0.00499548  0.00231187
 -0.00352152 -0.00241269 -0.00237846  0.00385078 -0.00203886 -0.00470292
 -0.00137619 -0.00383429  0.0024196  -0.00371323 -0.00219846  0.00313968
 -0.00344617 -0.00220617 -0.00352555 -0.00119135 -0.00484645  0.00196214
  0.00437842  0.00420597 -0.00324782  0.00484177  0.00383371 -0.00402128
 -0.00087797  0.00318496 -0.00160891  0.00251811 -0.00342429  0.00229055
 -0.00194213 -0.00458774 -0.0023172   0.00255128  0.00433975 -0.00493233
 -0.00011894  0.00020929  0.00389752 -0.00253374 -0.00219487  0.00345268
  0.00207684 -0.00495024  0.00401319 -0.00333675  0.00157402 -0.00237876
 -0.00379086 -0.00213833  0.00256704 -0.00433717 -0.00288563  0.00337329
  0.00037133  0.00347325  0.0001705   0.00489581]
[0]
```

基于doc2vec的文本聚类

In [9]:

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3).fit(text_vecs)

print(kmeans.predict(text_vecs))
```

```
[0 2 0 2 2 2 2 2 2 2 1 2 2 2 2 2 0 2 2 2 2 1 0 2 2 0 2 2 0 2 2 1 0 2 2 1 2
 0 2 2 0 2 2 1 1 0 0 0 2 2 1 1 2 1 1 2 2 2 2 2 1 2 2 1 1 0 2 2 2 2 2 2 0 2
 2 1 1 0 1 2 2 2 0 0 2 2 0 0 2 1 2 0 2 2 1 0 2 1 2 0 0 2 0 0 2 0 0 2 0 0 1
 0 2 1 2 2 1 2 2 0 0 2 1 2 2 0 2 2 2 2 2 2 2 0 2 2 2 0 1 0 2 2 0 2 0 2 2 0
 2 0 0 1 2 2 0 0 0 2 1 0 0 0 2 0 0 2 1 2 0 1 2 2 2 2 1 0 0 1 1 1 2 2 0 0 0
 0 0 0 0 2 0 1 1 1 2 2 1 1 0 2 0 2 2 2 1 2 1 2 2 0 0 2 0 2 2 2 2 1 1 0 2 2
 2 2 0 2 2 2 2 1 2 2 0 0 2 2 2 0 2 0 0 1 2 2 2 1 2 0 0 2 0 1 2 1 2 1 2 0 1
 2 2 2 2 2 1 2 0 2 0 0 0 2 0 2 2 2 2 2 2 2 2 2 1 2 1 0 2 2 2 1 0 0 1 1 2 2
 1 2 1 0 1 0 2 0 0 2 2 0 0 2 2 2 2 0 2 1 2 1 1 2 2 2 1 2 1 0 1 0 1 1 2 1 1
 0 2 0 2 0 0 2 2 2 0 0 0 2 0 0 0 0 0 1 1 2 2 2 2 2 0 2 2 2 2 2 0 2 2 1 2 2
 2 2 2 2 2 0 2 0 0 2 2 2 2 2 2 1 2 0 0 1 2 2 2 2 1 2 2 0 2 0 2 2 2 0 2 0 2
 1 2 0 2 2 2 0 2 2 2 0 1 1 2 1 2 2 0 2 1 0 1 2 2 2 2 2 2 2 2 2 2 0 2 0 2 2
 2 2 0 1 2 0 1 2 2 0 1 2 1 2 2 2 2 1 0 0 2 2 2 0 1 2 2 0 2 2 2 1 2 2 0 2 2
 2 1 2 1 2 2 2 2 2 0 0 2 0 2 0 2 0 0 0]
```

In [ ]: