“智慧政务”中的文本挖掘应用

摘要

智慧政务是信息时代下的产物，它将政务与科技融合，利用互联网的优点处理政务，使得工作人员可以更加高效，准确的解决人民的问题。它是一种创新型治理国家的方式，建立了多元化政务服务供给渠道。于此同时，能够加强法律保障体系，改善促进国家治理现代化的网络环境。最重要的是，“智慧政务”能够解决老百姓办事难的问题，让信息多跑路，群众少跑路。

本文旨在研究“智慧政务”中的文本挖掘应用。我们对给定的数据集进行分类聚类训练，并运用对应的算法，得到相应的模型，将模型投入到使用中，以解决实际问题。

本队伍在做 C 题的过程中，首先对数据进行脱敏处理，去除与文本分类无关的字符。由于中文文本中存在大量无法代表文本特征的词，因此我们需要将这些不能代表文本特征的词去掉，使用到正则表达式和停用词表，正则表达式可以很好地去掉文本中出现的特殊字符，而停用词表则可以去掉大量的中文无表征性词语。对文本进行向量化表达，首先对文本进行转换成词频再将其转换成权值矩阵。

我们通过 KNN 和 K-means 等算法，建立关于留言内容的一级标签分类模型。通过 KNN 和 K-means 等算法，建立关于留言内容的一级标签分类模型。对模型的分类效果的评价本文采用 F1 系数来衡量模型的分类效果，我们同时建立的多种分类模型中最佳的分类模型为 KNN。最终验证了模型的有效性，实现了利用文本挖掘找出有效的信息。

本文采用 LDA 主题模型对第二题的内容进行了聚类，并计算了用于评价模型效果的困惑度，最终得到的聚类效果能达到预期。

关键词：文本挖掘 脱敏处理 自然语言处理（NLP）词频 词云 分类 聚类 KNN 算法 F1 系数 TF-IDF 权值 LDA 主题模型

目录

一、 挖掘目标

本次建模目标是利用 C 题提供的数据，利用 KNN 和 K-means 算法，达到以下目标：

1）目前，大部分电子政务系统还是依靠人工根据经验处理，存在工作量大、效率低，且差错率高等问题。建立关于留言内容的一级标签分类模型，以准确高效的解决问题。
2）将某一时段内反映特定地点或特定人群问题的留言进行归类，定义合理的热度评价指标，并给出评价结果以及排名前 5 的热点问题。与此同时，给出相应热点问题对应的留言信息。

二、分析方法与过程

2.1.1 数据的处理

首先对文本进行脱敏处理，去除与文本分类无关的字符。由于中文文本中存在大量的语气词、代词等无法代表文本特征的词，因此我们需要将这些不能代表文本特征的词去掉。对于数字、英文字母和其他特殊字符时主要采用了 re 模块，使用正则表达式将其剔除。本文采用 jieba 库对文本进行分词，分词时将文本中出现的需要使用的新词添加到 jieba 库的词库，然后再进行分词处理。去除停用词时我们使用了常见停用词表，首先读入停用词表，再通过循环将文本里面在停用词表出现过的词去除，此时得到一个列表存储的数据集，我们在采用函数表达式将其一空格进行连接。对于分类标签，我们使用了 sklearn 库里面的 preprocessing 进行处理。最后将上述操作封装成函数，并返回处理好的数据以及处理好的分类标签。

2.1.2.1TF 算法：

TF-IDF 权重策略 TF：Term Frequency 即关键词频，是指一篇文章中关键词出现在所有文档的频率。

$$TF = \frac{词频}{文本总数}$$

IDF：Inverse Document Frequency 指逆向文本频率，是用来衡量文本中关键词权重的指数。
I

$$IDF = log(\frac{D}{D_W})$$

（D 文档总数，

D_W

某一个单词在文本中出现的次数）

$$TF - IDF = TF \times IDF$$

当 TF-IDF 值越大时说明越具有表征性。

2.1.2.2#问题 1 代码

```
#模型构建
from data_pro import dataProcess
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer,TfidfTransformer
from sklearn.metrics import precision_score
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier
#导入处理好的数据
d1,d2,label =dataProcess()
#切分训练集和测试集
d1_tr,d1_te,d2_tr,d2_te,label_tr,label_te = train_test_split(d1,d2,label,test_size=0.2)
countVectorizer = CountVectorizer()
#获取 idf 权值向量
#留言主题文本向量化
d1_tr = countVectorizer.fit_transform(d1_tr)
d1_te = CountVectorizer(vocabulary=countVectorizer.vocabulary_).fit_transform(d1_te)
x1_tr = TfidfTransformer().fit_transform(d1_tr.toarray()).toarray()
x1_te = TfidfTransformer().fit_transform(d1_te.toarray()).toarray()
#留言详情文本向量化
d2_tr = countVectorizer.fit_transform(d2_tr)
x2_tr = TfidfTransformer().fit_transform(d2_tr.toarray()).toarray()
d2_te = CountVectorizer(vocabulary=countVectorizer.vocabulary_).fit_transform(d2_te)
x2_te = TfidfTransformer().fit_transform(d2_te.toarray()).toarray()

#模型训练

#贝叶斯算法
#以留言主题位依据
model1 = GaussianNB()
model1.fit(x1_tr,label_tr)
y = model1.predict(x1_te)
res = precision_score(label_te,y,average=None)
print(res)
print(model1.score(x1_te,label_te)) ##测试集上的准确率
# print(model1.predict(x1_te))   #预测结果
#以留言详情位分类依据
model1.fit(x2_tr,label_tr)
print(model1.score(x2_te,label_te)) ##测试集上的准确率
# print(model1.predict(x2_te))   #预测结果
```

```
#支持向量机
#以留言主题位依据
model2 = SVC()
model2.fit(x1_tr,label_tr)
print(model2.score(x1_te,label_te)) ##测试集上的准确率
# print(model2.predict(x1_te))   #预测结果
#以留言详情位分类依据
model2.fit(x2_tr,label_tr)
print(model2.score(x2_te,label_te)) ##测试集上的准确率
# print(model2.predict(x2_te))   #预测结果

#逻辑斯蒂回归
# #以留言主题位依据
model3 = LogisticRegression()
model3.fit(x1_tr,label_tr)
print(model3.score(x1_te,label_te)) ##测试集上的准确率
# print(model3.predict(x1_te))   #预测结果
#以留言详情位分类依据
model3.fit(x2_tr,label_tr)
print(model3.score(x2_te,label_te)) ##测试集上的准确率
# print(model3.predict(x2_te))   #预测结果

#K 最近邻分类
# #以留言主题位依据
model4 = KNeighborsClassifier()
model4.fit(x1_tr,label_tr)
print(model4.score(x1_te,label_te)) ##测试集上的准确率
# print(model4.predict(x1_te))   #预测结果
#以留言详情位分类依据
model4.fit(x2_tr,label_tr)
print(model4.score(x2_te,label_te)) ##测试集上的准确率
# print(model4.predict(x2_te))   #预测结果

#分类决策树
# #以留言主题位依据
model5 = DecisionTreeClassifier()
model5.fit(x1_tr,label_tr)
print(model5.score(x1_te,label_te)) ##测试集上的准确率
# print(model5.predict(x1_te))   #预测结果
#以留言详情位分类依据
model5.fit(x2_tr,label_tr)
print(model5.score(x2_te,label_te)) ##测试集上的准确率
# print(model5.predict(x2_te))   #预测结果
```

```
#随机森林分类
# #以留言主题位依据
model6 = RandomForestClassifier()
model6.fit(x1_tr,label_tr)
print(model6.score(x1_te,label_te)) ##测试集上的准确率
# print(model6.predict(x1_te))   #预测结果
#以留言详情位分类依据
model6.fit(x2_tr,label_tr)
print(model6.score(x2_te,label_te)) ##测试集上的准确率
# print(model6.predict(x2_te))   #预测结果
#梯度提升分类树
# #以留言主题位依据
model7 = GradientBoostingClassifier()
model7.fit(x1_tr,label_tr)
print(model7.score(x1_te,label_te)) ##测试集上的准确率
# print(model7.predict(x1_te))   #预测结果
#以留言详情位分类依据
model7.fit(x2_tr,label_tr)
print(model7.score(x2_te,label_te)) #测试集上的准确率
# print(model7.predict(x2_te))   #预测结果
```

2.1.2.3 对数据的处理

```
#  数据处理
import pandas as pd
import re
import jieba


def dataProcess(file='附件 2.xlsx'):
    data = pd.read_excel(file)
    co = data.columns

    a=data['留言主题']
    b=data['留言详情']
    c = data['一级分类']
    #重组
    data_new = pd.concat([a,b,c],axis=1)
    ifo1 = data_new['留言主题']
    ifo2 = data_new['留言详情']
    label = data_new['一级分类']
```

```
    #分词
    ifo1_cut = ifo1.apply(lambda x: jieba.lcut(x))
    ifo2_cut = ifo2.apply(lambda x: jieba.lcut(x))

    stopWords = pd.read_csv('stopword.txt', encoding='GB18030', sep='hahaha', header=None,engine='python')
    stopWords = ['E7', 'A3','E8', 'A6', 'K8', 'B9', '会', '月', '日', 'G8'] + list(stopWords.iloc[:, 0])
    # print(stopWords)
    ifo1_end = ifo1_cut.apply(lambda x: [i for i in x if i not in stopWords])
    ifo2_end = ifo2_cut.apply(lambda x: [i for i in x if i not in stopWords])

    #使用空格进行连接
    adata1 =ifo1_end.apply(lambda x:' '.join(x))
    adata2 =ifo2_end.apply(lambda x:' '.join(x))
    return adata1,adata2,label
```

2.1.3 模型的建立:

模型构建，导入数据集并将数据和分类标签拆分成 80%的训练集，20%的测试集，使用了 sklearn 库中 CountVectorizer 将数据进行向量化处理，再使用 TfidfTransformer 进行 IT-IDF 权值化处理，由于我们把数据集拆分成测试集，因此再将文本进行向量化时得到的矩阵产生差异，故我们需要将训练集的维度共享到测试集上。导入 sklearn 库中的分类函数，本文使用了朴素贝叶斯分类、KNN、决策树、支持向量机等算法进行分类的操作，最终得到 KNN 的算法效果最佳。

LDA 模型构建，读入已处理好的数据，调用 gensim 库中 Dictionary 进行词袋构建，再将其转换成语料库，最后导入 LDA 模型传入参数实现数据的聚类。

2.1.3.1 文档的处理:

LDA 主题模型 LDA 主题模型是一个无监督的机器学习聚类模型，是一种文档主题生成模型，也称为一个三层贝叶斯概率模型，包含词、主题和文档三层结构。可以用来识别大规模文档集（document collection）或语料库（corpus）中潜藏的主题信息。采用了词袋的方法，将每一篇文档视为一个词频向量，从而将文本信息转化为了易于建模的数字信息。但是词袋方法没有考虑词与词之间的顺序，这简化了问题的复杂性，同时也为模型的改进提供了契机。每一篇文档代表了一些主题所构成的一个概率分布，而每一个主题又代表了很多单词所构成的一个概率分布。LDA 主题模型只用确定聚类的主题数目（number of topics），就够得到聚类结果。

$$P(wld) = P(wlt) \times P(tld)$$

其中，P(wld)表示专利文档 d 生成词语 w 的概率；P (wlt)表示词语 w 在主题 t 下的概率；P(tld)表示生成话题 t 的概率。在主题向量的生成过程中，假设每个主题的 P(wld)相等，则选择 argmax(P(wld))，即使词语生成概率最大的主题作为该词所属主题，累加后可得到新的主题向量 P(tld)。根据新的主题向量对 P(wld)进行更新迭代，最终可以得到稳定的主题向量。

我们用 LDA 模型聚成 40 个类的结果：

0.020*"溪湖" + 0.016*"烧烤" + 0.014*"夜宵" + 0.013*"油烟" + 0.013*"小时" + 0.013*"游玩" + 0.012*"桃花" + 0.012*"领导" + 0.010*"区山" + 0.009*"扰民"

0.022*"居民" + 0.020*"解决问题" + 0.018*"油烟" + 0.017*"柴火" + 0.013*"店面" + 0.012*"味" + 0.012*"烧" + 0.012*"维权" + 0.012*"找" + 0.012*"灶台"

0.016*"系统" + 0.013*"占地" + 0.012*"面积" + 0.012*"土地" + 0.012*"分割" + 0.011*"不动产" + 0.011*"记录" + 0.010*"划拨" + 0.009*"一套" + 0.009*"补交"

0.007*"提供" + 0.007*"材料" + 0.007*"社保" + 0.007*"拒" + 0.007*"补贴" + 0.007*"购房" + 0.006*"服务" + 0.005*"不通" + 0.004*"人才" + 0.004*"希望"

0.015*"居民" + 0.014*"店面" + 0.014*"柴火" + 0.011*"解决问题" + 0.009*"楼顶" + 0.009*"油烟" + 0.009*"事" + 0.009*"帮" + 0.009*"维权" + 0.009*"烧"

0.035*"系统" + 0.016*"记录" + 0.015*"养老保险" + 0.015*"工作人员" + 0.015*"缴费" + 0.014*"参保" + 0.012*"社保" + 0.012*"社保局" + 0.011*"询问" + 0.010*"市人"

0.017*"购房" + 0.017*"材料" + 0.016*"提供" + 0.012*"人才" + 0.012*"拒" + 0.011*"社保" + 0.011*"补贴" + 0.010*"时" + 0.010*"居民" + 0.008*"店面"

0.015*"学院" + 0.014*"公交" + 0.014*"暮云" + 0.013*"城轨" + 0.013*"站" + 0.012*"公交车" + 0.012*"南塘" + 0.011*"服务" + 0.011*"小区" + 0.010*"标准"

0.040*"学生" + 0.026*"实习" + 0.016*"学校" + 0.014*"回来" + 0.013*"物流" + 0.012*"学院" + 0.010*"企业" + 0.009*"强制" + 0.009*"套餐" + 0.008*"专业"

0.083*"套餐" + 0.030*"飞享" + 0.022*"取消" + 0.020*"业务" + 0.015*"返回" + 0.015*"客服" + 0.015*"优惠" + 0.014*"发现" + 0.014*"生效" + 0.014*"短信"

0.009*"系统" + 0.006*"记录" + 0.005*"养老保险" + 0.004*"工作人员" + 0.004*"社保" + 0.004*"月份" + 0.003*"缴费" + 0.003*"询问" + 0.003*"市人" + 0.003*"社保局"

0.026*"服务" + 0.018*"标准" + 0.017*"收费" + 0.010*"居民" + 0.009*"城市" + 0.009*"物业" + 0.009*"垃圾清运" + 0.008*"装修" + 0.008*"拟定" + 0.008*"听证会"

0.016*"飘" + 0.016*"烧烤" + 0.015*"休息" + 0.014*"影响" + 0.014*"小区" + 0.009*"工作" + 0.009*"孩子" + 0.009*"嚣张" + 0.009*"部门" + 0.009*"权利"

0.001*"万科" + 0.001*"餐馆" + 0.001*"店铺" + 0.001*"安静" + 0.001*"声" + 0.001*"吆喝声" + 0.001*"关" + 0.001*"充斥" + 0.001*"整顿" + 0.001*"没见"

0.016*"药品" + 0.015*"癌症" + 0.015*"医保" + 0.013*"小时" + 0.013*"工作" + 0.013*"学校" + 0.013*"政府" + 0.012*"孩子" + 0.011*"世界" + 0.010*"烧烤"

0.015*"居民" + 0.014*"服务" + 0.013*"小区" + 0.010*"收费" + 0.009*"物业" + 0.009*"油烟" + 0.008*"标准" + 0.007*"解决问题" + 0.007*"柴火" + 0.007*"公交车"

0.024*"小区" + 0.021*"小孩" + 0.016*"居民" + 0.016*"架空层" + 0.015*"很大" + 0.014*"门诊" + 0.013*"报销" + 0.013*"医院" + 0.008*"做法" + 0.008*"融洽"

0.005*"小孩" + 0.004*"医院" + 0.004*"报销" + 0.003*"老百姓" + 0.003*"贵" + 0.003*"门诊" + 0.003*"肺炎" + 0.003*"花费" + 0.003*"住院" + 0.003*"一笔"

0.019*"安全隐患" + 0.019*"住户" + 0.019*"水" + 0.019*"电梯" + 0.017*"孩子" + 0.017*"学校" + 0.015*"工作" + 0.012*"学生" + 0.010*"打工" + 0.010*"外省"

0.024*"溪湖" + 0.014*"游玩" + 0.012*"区山" + 0.010*"桃花" + 0.010*"有利于" + 0.009*"坐线" + 0.008*"副" + 0.008*"拥堵" + 0.008*"桥" + 0.008*"中心"
0.020*"长期" + 0.020*"油烟" + 0.020*"直排" + 0.020*"合作社" + 0.020*"失业" + 0.020*"相关" + 0.018*"政府" + 0.017*"职工" + 0.017*"供销" + 0.015*"缴纳"
0.016*"公司" + 0.013*"店面" + 0.013*"高尔夫" + 0.010*"油烟" + 0.010*"居民" + 0.010*"柴火" + 0.009*"解决问题" + 0.009*"大四" + 0.008*"味" + 0.008*"异味"
0.031*"土地" + 0.020*"补交" + 0.020*"一套" + 0.018*"划拨" + 0.017*"不动产" + 0.017*"分割" + 0.016*"面积" + 0.015*"占地" + 0.011*"该交" + 0.011*"八年"

0.028*"过年" + 0.022*"学生" + 0.020*"家长" + 0.015*"本该" + 0.015*"外来" + 0.014*"学子" + 0.013*"本科" + 0.013*"孩子" + 0.013*"求学" + 0.009*"新"

0.033*"原因" + 0.033*"证件" + 0.033*"卫生室" + 0.022*"情况" + 0.022*"上级" + 0.022*"新" + 0.022*"迟迟" + 0.022*"不见" + 0.022*"许可证" + 0.022*"发"

0.029*"小区" + 0.027*"烧烤" + 0.023*"油烟" + 0.021*"经营" + 0.015*"临街" + 0.013*"魅力" + 0.012*"夜宵" + 0.011*"居民" + 0.010*"小时" + 0.009*"餐饮"

0.073*"学生" + 0.054*"实习" + 0.044*"学校" + 0.029*"安排" + 0.016*"几个" + 0.015*"毕业" + 0.015*"领导" + 0.015*"经济" + 0.015*"学院" + 0.015*"劳动力"

0.019*"系统" + 0.018*"小区" + 0.014*"记录" + 0.014*"工作人员" + 0.013*"整顿" + 0.012*"每到" + 0.012*"社保局" + 0.011*"社保" + 0.010*"公众" + 0.009*"养老保险"

0.033*"实习" + 0.033*"企业" + 0.031*"物流" + 0.031*"回来" + 0.024*"学校" + 0.023*"学院" + 0.023*"寝室" + 0.022*"专业" + 0.022*"暑假" + 0.021*"强制"

0.015*"系统" + 0.011*"记录" + 0.009*"工作人员" + 0.009*"养老保险" + 0.009*"询问" + 0.007*"社保局" + 0.007*"月份" + 0.007*"缴费" + 0.007*"参保" + 0.006*"社保"

0.004*"公司" + 0.002*"管理" + 0.002*"高尔夫" + 0.002*"大四" + 0.002*"工作" + 0.002*"系里" + 0.002*"实训" + 0.002*"儿童" + 0.002*"老师" + 0.002*"回"

0.018*"游玩" + 0.018*"溪湖" + 0.014*"区山" + 0.013*"桃花" + 0.010*"副" + 0.010*"坐线" + 0.010*"领导" + 0.009*"有利于" + 0.009*"商业空间" + 0.009*"中心"

0.018*"小区" + 0.018*"影响" + 0.018*"换证" + 0.017*"休息" + 0.017*"申请" + 0.017*"没人" + 0.017*"烧烤" + 0.016*"受理" + 0.014*"商铺" + 0.014*"油烟"

0.009*"系统" + 0.005*"记录" + 0.004*"养老保险" + 0.004*"社保" + 0.003*"工作人员" + 0.003*"公众" + 0.003*"询问" + 0.003*"群众" + 0.003*"社微信" + 0.003*"月份"

0.001*"万科" + 0.001*"餐馆" + 0.001*"店铺" + 0.001*"安静" + 0.001*"声" + 0.001*"吆喝声" + 0.001*"关" + 0.001*"充斥" + 0.001*"整顿" + 0.001*"没见"

0.025*"套餐" + 0.018*"领导" + 0.017*"独生子女" + 0.014*"油烟" + 0.013*"小区" + 0.013*"居民" + 0.012*"西地省" + 0.012*"真的" + 0.012*"护理假" + 0.012*"试行"

0.028*"标准" + 0.025*"服务" + 0.016*"物业" + 0.015*"系统" + 0.013*"收费" + 0.011*"记录" + 0.010*"发展" + 0.009*"城市" + 0.008*"费" + 0.008*"月份"

0.026*"服务" + 0.018*"标准" + 0.018*"收费" + 0.014*"物业" + 0.012*"管理" + 0.011*"委员会" + 0.011*"公司" + 0.010*"发展" + 0.010*"停放" + 0.010*"听证会"

0.025*"网络" + 0.023*"领导" + 0.020*"社保" + 0.018*"供销" + 0.018*"职工" + 0.017*"覆盖" + 0.017*"基础" + 0.016*"相关" + 0.016*"诉求" + 0.016*"养老保险"

0.015*"系统" + 0.011*"记录" + 0.011*"游玩" + 0.010*"养老保险" + 0.010*"溪湖" + 0.009*"社保" + 0.008*"询问" + 0.007*"工作人员" + 0.007*"月份" + 0.007*"社保局"

2.1.3.2 问题二代码:

```
#LDA 主题模型聚类
from d_pro import dPro
from gensim.corpora import Dictionary
from gensim.models import LdaModel
import re
d1,d2 = dPro()

#LDA  模型构建
mid = list(d1)
dic = Dictionary(mid)
bow = [dic.doc2bow(d) for d in mid] #词袋
# print(dic)
# print(bow)
m = LdaModel(corpus=bow,id2word=dic,num_topics=35)
n = m.num_topics
print(n)
for i in range(n):
  print(m.print_topic(i))



#问题 2 数据预处理
import pandas as pd
import jieba
```

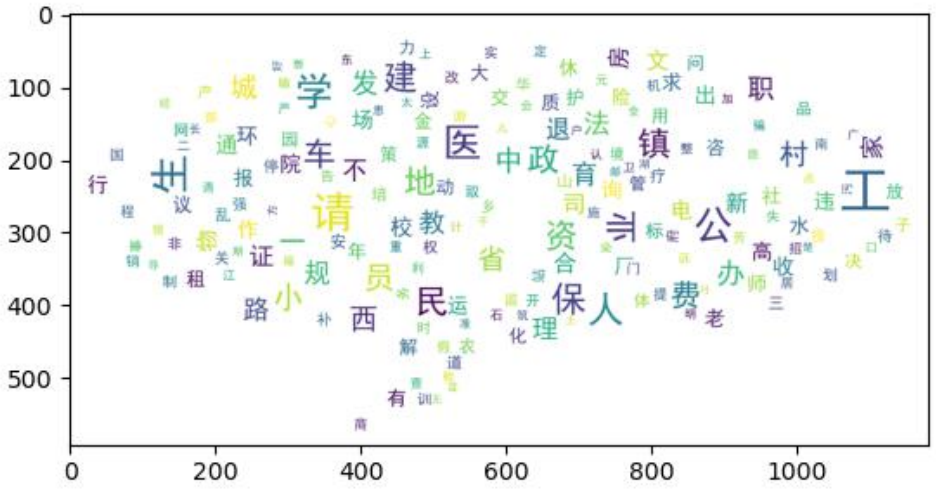
```
import re

#读入数据
def dPro(file='C:/Users/步嘎/Desktop/C 题/data/附件 3.xlsx'):
    data = pd.read_excel(file)
    d = data['留言详情']
    d1 = d.dropna().apply(lambda x: re.sub('\n','',x))
    d1 = d1.dropna().apply(lambda x: re.sub('\t','',x))
    d1 = d1.dropna().apply(lambda x: re.sub('[A-Za-z0-9#$%^&*]{1,}', '', x))
    # print(d1)
    jieba.load_userdict('newwords.txt')
    #分词
    d1_cut = d1.apply(lambda x: jieba.lcut(x))
    # print(d1_cut)
    # #加载停用词
    stopWords = pd.read_csv('stopword.txt', encoding='GB18030', sep='hahaha',
header=None,engine='python')
    stopWords = ['尊敬','\u3000','号','窗口','说','证','号','年','请','元','倍','岗','城','道','摊','办','
做','岭','查','搞','梅','点','开'] + list(stopWords.iloc[:, 0])
    d2 = d1_cut.apply(lambda x: [i for i in x if i not in stopWords])
    d3 = d2.apply(lambda x:' '.join(x))
    # print(d2)
    #文本去重

    return d2,d3
#
```

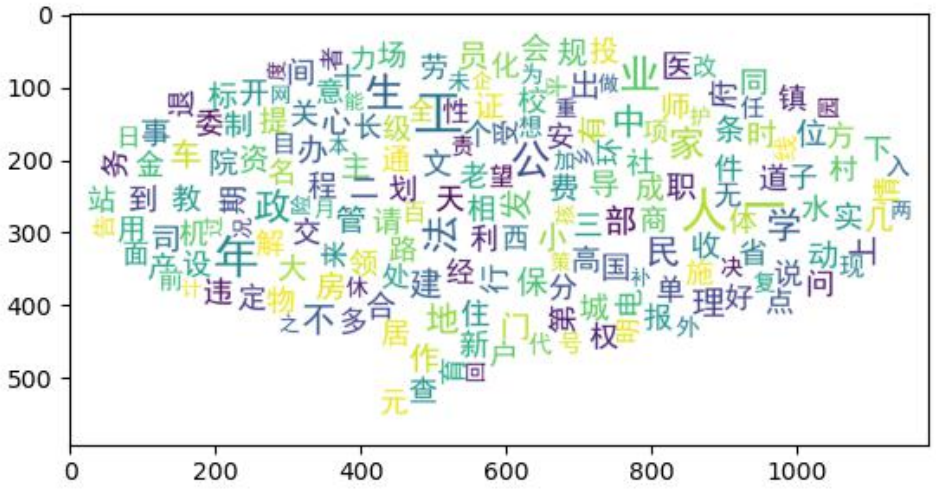
2.1.4 词云图统计

绘制词云图，首先读入已处理好的数据集，再统计文本中每个词出现的频次，将其存储成一个字典。导入绘图所需的背景图片，导入绘制词云图模块生成一个绘制词云的对象，再

调用其拟合函数进行词云的绘制。

问题 1 词云图:



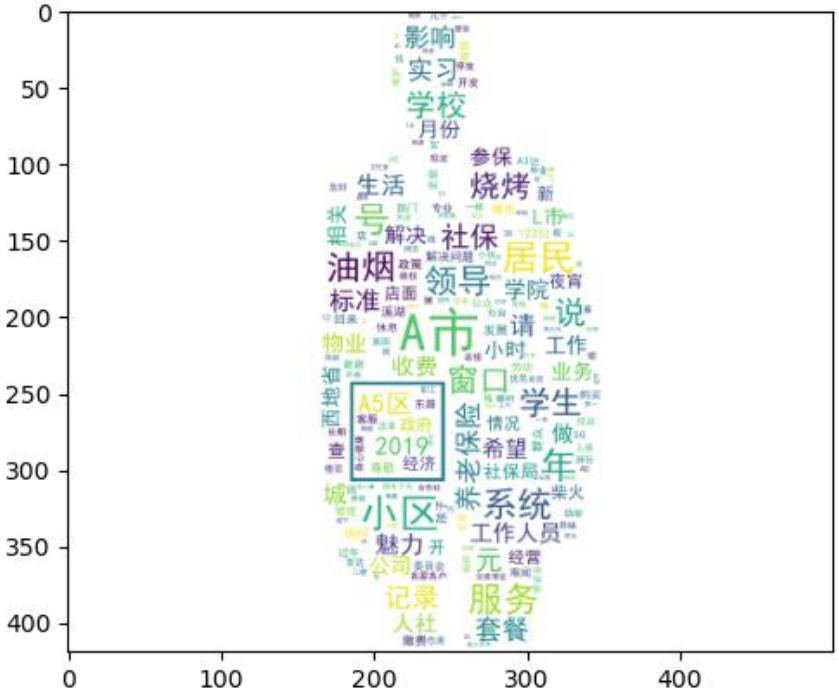


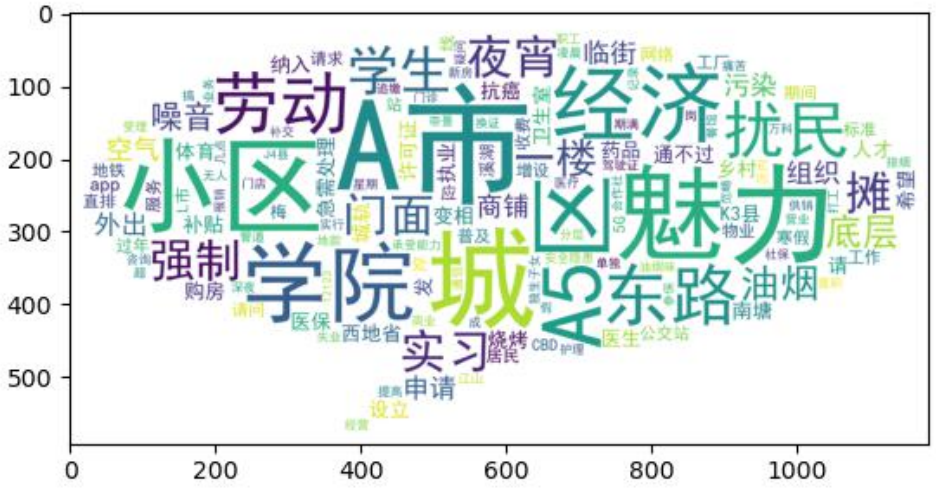
对应代码：

```
#词云图统计
from data_pro import dataProcess
import matplotlib.pyplot as plt
from wordcloud import WordCloud
data1,data2,data3 = dataProcess()      #数据读入
mask =plt.imread('duihuakuan.jpg')     #轮廓图
#词频统计
word_fre1={} #用于存放留言主题词频信息
word_fre2={} #用于留言详情词频信息
word_fre3={} #用于存放标签词频信息
for i in data1 :
    for j in i :
        if j not in word_fre1.keys():
            word_fre1[j]=1
        else:
            word_fre1[j]+=1

for i in data2 :
    for j in i :
        if j not in word_fre2.keys():
            word_fre2[j]=1
        else:
            word_fre2[j]+=1

for i in data3 :
    for j in i :
        if j not in word_fre3.keys():
            word_fre3[j]=1
        else:
            word_fre3[j]+=1

#生成一个绘制词云的对象
wc = WordCloud(mask=mask,background_color='white',font_path=r'C:\Windows\Fonts\simhei.ttf')
wc.fit_words(word_fre1) #留言主题词云图
plt.imshow(wc)
plt.show()
wc.fit_words(word_fre2) #留言详情词云图
plt.imshow(wc)
plt.show()
```

```
wc.fit_words(word_fre3) #一级标签词云图
plt.imshow(wc)
plt.show()
```

问题 2 词云图:





对应代码:

```
#问题 2 词云
import pandas as pd
import jieba
import re
import matplotlib.pyplot as plt
from wordcloud import WordCloud
data = pd.read_excel('附件 3.xlsx')
d1 = data['留言详情']
d2 = data['留言主题']

# nw = re.search('[A-Z]市|[A-Z][0-9]区|[A-Z][0-9]县',d1)
c =  re.findall('[A-Z]市|[A-Z0-9]{1,}区|[A-Z0-9]{1,}县',  str(d1))
print(c)
jieba.load_userdict('nw.txt')
d1 = d1.apply(lambda x: jieba.lcut(x))
d2 = d2.apply(lambda x: jieba.lcut(x))
stopWords = pd.read_csv('stopword.txt', encoding='GB18030', sep='hahaha', header=None,engine='python')
stopWords = list(stopWords.iloc[:, 0])+[',','！ ','?','/','\\','.', ' ','。',"]
d1 = d1.apply(lambda x: [i for i in x if i not in stopWords])
d2 = d2.apply(lambda x: [i for i in x if i not in stopWords])
# d1 = d1.dropna().apply(lambda x: re.sub('[0-9]|[a-zA-Z]{1,}', '', x))
# d2 = d2.dropna().apply(lambda x: re.sub('[0-9]|[a-zA-Z]{1,}', '', x))
```

```
    #词频统计
    word_fre1={} #用于存放留言主题词频信息
    word_fre2={} #用于留言详情词频信息
    word_fre3={} #用于存放标签词频信息
    for i in d1 :
        for j in i :
            if j not in word_fre1.keys():
                word_fre1[j]=1
            else:
                word_fre1[j]+=1

    for i in d2 :
        for j in i :
            if j not in word_fre2.keys():
                word_fre2[j]=1
            else:
                word_fre2[j]+=1

    mask1 =plt.imread('详情.jpg')    #轮廓图
    mask2 =plt.imread('duihuakuan.jpg')    #轮廓图
    #生成一个绘制词云的对象
    wc1 = WordCloud(mask=mask1,background_color='white',font_path=r'C:\Windows\Fonts\simhei.ttf')
    wc2 = WordCloud(mask=mask2,background_color='white',font_path=r'C:\Windows\Fonts\simhei.ttf')
    wc1.fit_words(word_fre1) #留言主题词云图
    plt.imshow(wc1)
    plt.show()
    wc2.fit_words(word_fre2) #留言详情词云图
    plt.imshow(wc2)
    plt.show()
```

三、结果分析

3.1 结果分析

本文中主要使用 f1 系数来评价分类结果，本文得出的最佳分类结果 F1 系数在 0.7 以上，分析结果可知本文在对数据进行处理时仍然存在处理不当的地方，

F1 score

F1 分数是用来衡量二分类模型精确度的一种指标,同时考虑到分类模型的准确率和召回

率。可看做是准确率和召回率的一种加权平均。

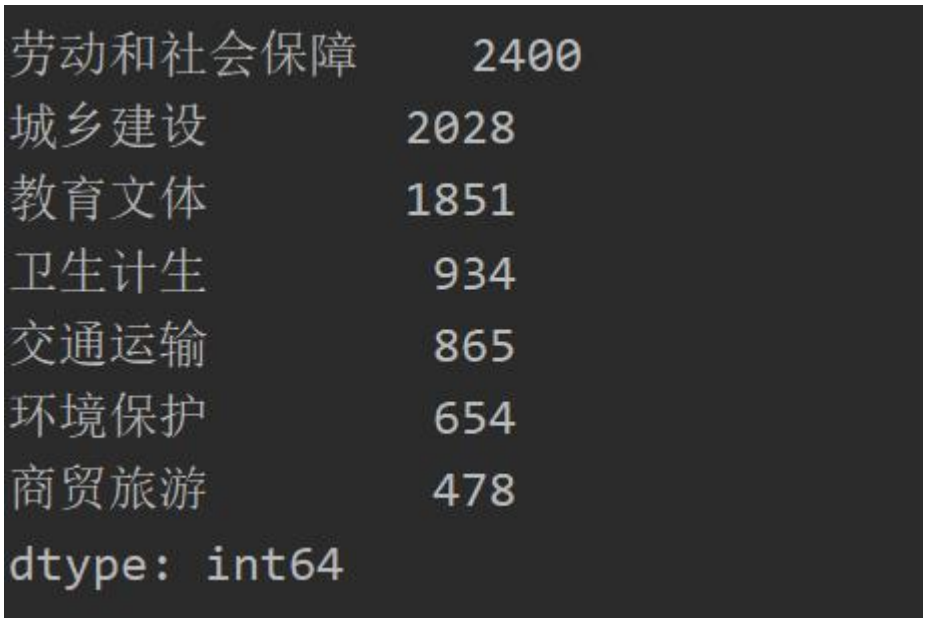
在已知精确率和召回率的情况下 求得的一种平均的结果。
Precision：预测为对的当中，原本是对的比例越大越好，1 为理想状态）
recall：原本为对的当中，预测是对的比例（越大越好，1 为理想状态）
F-measure：由于 precision 和 recall 两个指标不想管，所以用 F-measure 将他们合并成一个衡量指标（越大越好，1 为理想状态）

3.2 结果分析

困惑度（perplexity）是一种信息理论的测量方法，用来衡量语言模型好坏的指标。b 的 perplexity 值定义为基于 b 的熵的能量（b 可以是一个概率分布，或者概率模型），通常用于概率模型的比较。
概率分布的 perplexity 公式：
困惑度值越小则模型的效果越好，本文中将分类主题设置为 40 时得到的困惑度值为 -46.559775237179444，因此本文得到比较不错的聚类效果

四、结论

近年来，随着微信、微博、市长信箱、阳光热线等网络问政平台逐步成为政府了解民意、汇聚民智、凝聚民气的重要渠道，各类社情民意相关的文本数据量不断攀升，给以往主要依靠人工来进行留言划分和热点整理的相关部门的工作带来了极大挑战。同时，随着大数据、云计算、人工智能等技术的发展，建立基于自然语言处理技术的智慧政务系统已经是社会治理创新发展的新趋势，对提升政府的管理水平和施政效率具有极大的推动作用。我们通过 KNN 和 K-means 等算法，建立关于留言内容的一级标签分类模型，能够自动将群众问政留言记录分类到对应的主题下。得出了以下结论：群众反映的问题基本集中在劳动和社会保障，城乡建设等



面对群众反映的问题，我们通过 LDA 算法模型，及时将其中的热点问题总结归类，定

义合理的热度评价指标，并给出评价结果。得出了以下结论：群众反映的热点问题可集中在社区服务，学校操作，村委作为，公共建设，政府规划等。即政府需多多作为来使民众的幸福感，人民的生活水平提高。

五、参考文献

[1]. 艾楚涵，江迪，吴建德.基于主题模型和文本相似度计算的专利推荐研究.昆明理工大学.2020
[2].王千，王成，冯振元，叶金凤. K_means 聚类算法研究综述.2012
[3].李秀娟. KNN 分类算法研究.2009
[4].王德宝. 基于 KNN 算法的改进研究及其在数据分类中的应用。2018