“智慧政务”中的文本挖掘应用

摘要

随着社会的发展，各类社情民意相关的文本数据量不断攀升。信息量大，文本内容分析困难，人工处理的过程过于繁杂，处理操作较慢，处理时间较长。随着社会的发展，科技的进步，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。因此运用 SVM（Support Vector Machine）算法准确、有效的解决大规模的文本分类问题。

针对问题一，首先对附件 2 的留言详情进行分词处理。第二部对以进行分词处理的留言详情进行特征处理，包括特征选择、特征抽取、特征权重计算。第三步将附件 2 留言详细提取到的文本语义特征，利用 python 软件编写程序代码，建立附件 1 与附件 2 的联系。用于确定附件 2 的留言类型。最后使用 F-Score 对分类方法进行评价。

针对问题二，根据朴素贝叶斯算法将附件 3 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。使用 ROST CM 工具进行文本挖掘，还需要将 Excel 表格转换为文本，方便处理，然后进行文本处理、聚类分析，得出相关数据。

关键词：文本分类； SVM 算法；朴素贝叶斯算法；ROST CM

目 录

1 挖掘目标

1.1 挖掘背景

在互联网迅速发展的时代，工作繁琐且效率低下的人工操作已经逐渐被社会淘汰。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。故而，出现了智慧政务，它指的是可以实现政务服务高效化,数据实时化,响应及时化的政务工具,可以简单、高效的解决百姓的各类问题,数据统计清晰明了化。随之相关的大数据、云计算、人工智能等技术得到广泛应用，我们对于建立基于自然语言处理技术的智慧政务系统已经是迫在眉睫，这个对提升政府的管理水平和施政效率具有极大的推动作用。

1.2 挖掘目标

对于问题一给出的群众留言记录，我们用 Python 处理附件 2 数据使其简单化，再采用决策对数据进行分类，给出最佳的标签分类。

对于问题二，为了更大程度的收集群众最关心的问题，对问题获得最佳处理，先用 ROST CM 对数据进行分类，获得最大相似度，再用朴素贝叶斯算法进行数据的辅助处理，得出排名前五的热点问题。

对于问题三，采用语言描述，对相关部门做出的答复意见进行分类，再从答复中提取一定的相关性，用流程图进行相关的分析解释，做到对群众最好的答复。

1.3 挖掘现状

随着大数据，互联网，人工智能的不断的发展，文本分类已经较成熟，但是却没有较高的查全率和查准率。本文针对各类社情民意相关的文本进行挖掘，在传统的文本挖掘技术上创新。利用 SVM 算法和朴素贝叶斯算法，在解决文本分类的基础上，得到较高的查全率和查准率。

2 分析方法与过程

2.1 问题一分析方法与过程

2.1.1 问题一模型建立流程图

针对问题一，首先对附件 2 的留言详情进行分词处理。第二部对留言详细进行预处理，依次是文本分词，过滤停用词，词频统计，特征选择，文本表示。第三步将附件 2 留言详细提取到的文本语义特征，利用 python 软件编写程序代码，建立附件 1 与附件 2 的联系。用于确定附件 2 的留言类型。再建立一级标签分类模型，最后使用 F-Score 对分类方法进行评价。

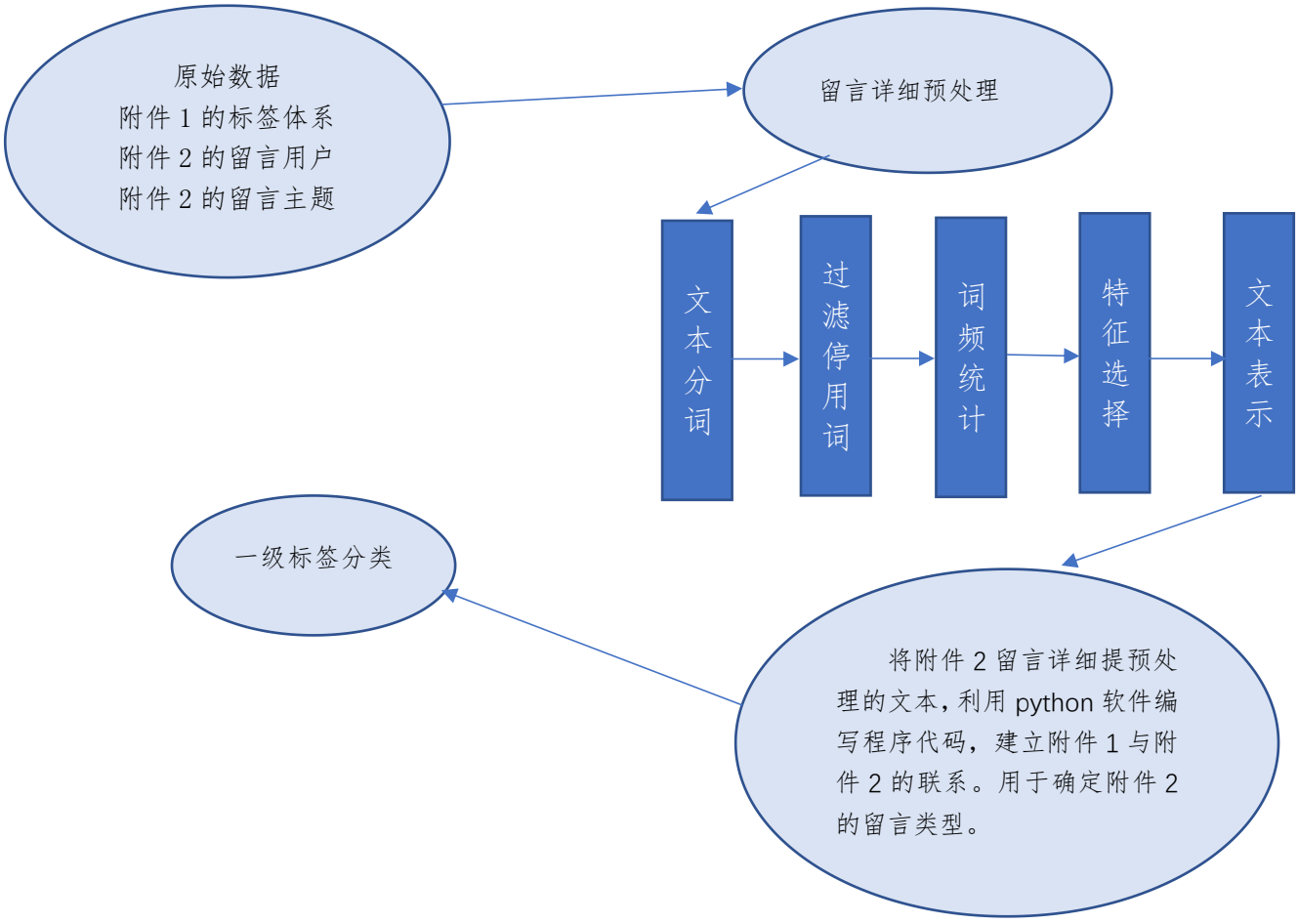


图 2.1.1 模型建立流程图

2.1.2 留言详细预处理

留言详细预处理首先是文本分词，现代分词都是基于统计的分词，而统计的样本内容来自于一些标准的语料库。假如有一个句子：“小明来到荔湾区”，我们期望语料库统计后分词的结果是："小明/来到/荔湾/区"，而不是“小明/来到/荔/湾区”，从统计的角度，我们期望"小明/来到/荔湾/区"这个分词后句子出现的概率要比“小明/来到/荔/湾区”大。如果用数学的语言来说，如果有一个句子 SS,它有 m 种分词选项如下：

$$A_{11}A_{12}....A_{1n}A_{11}A_{12}...A_{1n1}$$

$$A_{21}A_{22}....A_{2n2}A_{21}A_{22}...A_{2n2}$$

$$.....$$

$$A_{m1}A_{m2}....A_{mnm}A_{m1}A_{m2}...A_{mnm}$$

其中下标 $nini$ 代表第 ii 种分词的词个数。如果我们从中选择了最优的第 r 种分词方法，那么这种分词方法对应的统计分布概率应该最大，即：

$$r = argmax\, iP(A_{i1},A_{i2}....,A_{ini})$$

但是我们的概率 $P(A_{i1},A_{i2}....,A_{ini})$分并不好求出来，因为它涉及到 $nimi$ 个分词的联合分布。在 NLP 中，为了简化计算，我们通常使

用马尔科夫假设，即每一个分词出现的概率仅仅和前一个分词有关，即：

$$P(A_{i1},A_{i2},\ldots,A_{i(j-1)},) = P\left(\frac{A_{ij}}{A_{i(j-1)}}\right)P\left(\frac{A_{ij}}{A_{i1},A_{i2},\ldots,A_{i(j-1)}}\right)$$
$$= P(A_{ij}/A_{i(j-1)})$$

相同的假设来简化模型复杂度。使用了马尔科夫假设，则我们的联合分布就好求了，即：

$$P(A_{i1},A_{i2},\ldots,A_{ini})$$
$$= P(A_{i1})P\left(\frac{A_{i2}}{A_{i1}}\right)P\left(\frac{A_{i3}}{A_{i2}}\right)\ldots P(\frac{A_{i(ni)}}{A_{i(ni-1)}P(A_{i1},A_{i2},\ldots,A_{ini})}$$
$$= P(A_{i1})P\left(\frac{A_{i2}}{A_{i1}}\right)P\left(\frac{A_{i3}}{A_{i2}}\right)\ldots P(A_{ini}/A_{i(ni-1)}))$$

而通过我们的标准语料库，我们可以近似的计算出所有的分词之间的二元条件概率，比如任意两个词 $w1$，$w2w1$，$w2$，它们的条件概率分布可以近似的表示为：

$$P\left(\frac{w1}{w2}\right) = P(w1,w2)P(w1) \approx freq(w1,w2)freq(w1)p\left(\frac{w2}{w1}\right)$$
$$= p(w1,w2)p(w1) \approx freq(w1,w2)freq(w1)$$

$$P\left(\frac{w1}{w2}\right) = P(w2,w1)P(w2) \approx freq(w1,w2)freq(w2)p\left(\frac{w1}{w2}\right)$$
$$= p(w2,w1)p(w1) \approx freq(w1,w2)freq(w2)$$

其中 $freq(w1,w2)$表示 $w2$，$w2w1$，$w1$ 在语料库中相邻一起出现的次数，而其中 $freq(w1)$，$freq(w2)$ $freq(w1)$，$freq(w2)$分别表示 $w1$，$w2w1$，$w2$ 在语料库中出现的统计次数。利用语料库建立的统计概率，对于一个新的句子，我们就可以通过计算各种分词方法对应的联合分布概率，找到最大概率对应的分词方法，即为最优分词。留言详细预处理第二步就是过滤停用词。留言详细预处理第三步词频统计，文本分词文本中能观察到的量其实只有两个：词频和文档频率，所有的方法一律以这两个量为计算基础。简单综合这两者的 TF-IDF 选择出来的特征不具有类别区分度。以文档频率为基础的特征选择算法有文档频次方法（直接依据文档频率大小排序的方法）、卡方检验、信息增益、互信息等。

2.1.3 python 编程建立模型

将附件 2 留言详细提预处理的文本，利用 python 软件编写程序代码，建立附件 1 与附件 2 的联系。用于确定附件 2 的留言类型。构造 SVM 多类分类器的方法主要有两类：一类是直接法，直接在目标函数上进行修改，将多个分类面的参数求解合并到一个最优化问题中，通过求解该最优化问题“一次性”实现多类分类。另一类是间接法，主要是通过组合多个二分类器来实现多分类器的构造，常见的方法有 one-against-one 和 one-against-all 两种。 本文所要求的分类较多，所以本文采用间接法求解。

2.2 问题二分析方法与过程

2.2.1 关于问题二的文本挖掘过程

文本挖掘可以对大量文档集合的内容进行总结、分类、聚类、关联分析、分布分析以及趋势预测。这里主要对文本聚类进行说明，文本聚类与文本分类的不同之处在于,聚类没有预先定义好主题类别,它的目标是将文档集合分成若干个簇,要求同一簇内文档内容的相似度尽可能地大,而不同簇之间的相似度尽可能地小。因此,我们可以利用文本聚类技术将搜索引擎的检索结果划分为若干个簇,用户只需要考虑那些相关的簇,大大缩小了所需要浏览的结果数量。如下图：

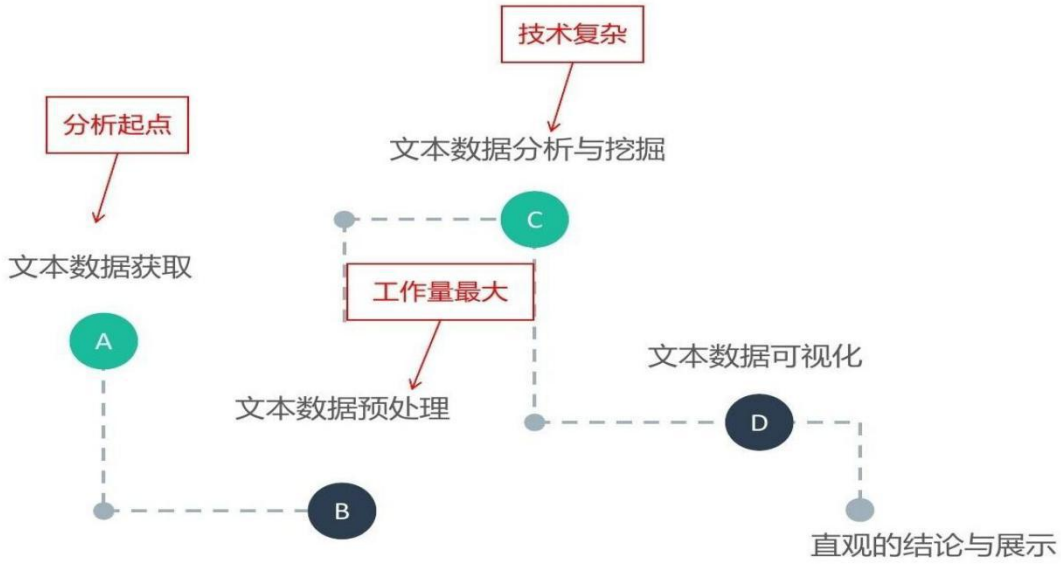


图 2.2.1 文本挖掘的过程

2.2.2 朴素贝叶斯分类技术

(1) 朴素贝叶斯基本描述

朴素贝叶斯分类技术以贝叶斯定理为基础，通过数据的先验概率，利用贝叶斯公式计算出其后验概率，并选择具有最大后验概率的类作为该对象所属的类。

在给定训练数据T时，确定假设空间λ中的最佳假设。贝叶斯算法提供了从先验概率P(λ)以及P(T)和P(T|λ)计算后验概率P(λ|T)的方法，其公式为

$$P(\lambda|T)=\frac{P(T|\lambda)P(\lambda)}{P(T)}$$

其中，P(λ)表示λ的先验概率，P(T)表示待观察训练数据T的先验概率,P(T|λ)表示给定λ时观察数据T对应的概率，P(λ|T)表示λ的后验概率，即给定训练数据T时人成立的概率。假设集合λ并寻找观察数据T在其中的概率最大的假设h∈λ,称为极大后验(MAP)假设。利用贝叶斯公式计算每个待选假设的后验概率，以此来确定MAP假设，即

$$h_{\mathrm{MAP}}=\underset{h\in H}{\operatorname{argmax}}P(\lambda|T)=\underset{h\in H}{\operatorname{argmax}}\frac{P(T|\lambda)P(\lambda)}{P(T)}=\underset{h\in H}{\operatorname{argmax}}P(T|\lambda)P(\lambda)$$

P(T)是不依赖于A的常量。在某些情况下，可假定λ中每个假设有相同的先验概率(即对H中任意λ_i和λ_j,$P(\lambda_i)=P(\lambda_j)$)。进一步简化，只需考虑P(T|λ)来寻找极大可能假设。P(T|λ)常称为给定λ时数据T的似然度，而使F(T|λ)最大的假设被称为极大似然h_{ML}，即

$$h_{\mathrm{ML}}=\underset{h\in H}{\operatorname{argmax}}P(T|\lambda)$$

朴素贝叶斯分类器应用的学习任务中，每个实例x可由属性值的合取描述，而目标函数f(x)从某有限集合V中取值。贝叶斯方法的新实例分类目标是在给定描述实例的属性值$\{a_1,a_2,\ldots,a_n\}$下，得到最可能的目标值υ_{MAP},即

$$v_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j | a_1, a_2, \cdots, a_n)$$

基于贝叶斯公式重写为

$$v_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} \frac{P(a_1, a_2, \cdots, a_n | v_j) P(v_j)}{P(a_1, a_2, \cdots, a_n)} = \underset{v_j \in V}{\operatorname{argmax}} P(a_1, a_2, \cdots, a_n | v_j) P(v_j) \quad ①$$

朴素贝叶斯NB分类器算法基于一个简单的假定：估算目标值时属性$\{a_1, a_2, \cdots, a_n\}$之间条件是相互独立的，则观察到$a_1, a_2, \cdots, a_n$的联合概率正好是对每个单独属性的概率乘积，为

$$P(a_1, a_2, \cdots, a_n | v_j) = \prod_i P(a_i | v_j)$$

将其代入式①中，可得到NB分类器所使用的方法为

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j)$$

当所需的条件独立性能够被满足时，朴素贝叶斯分类器输出的(υ_{NB})等于MAP分类。

(2)朴素贝叶斯群众留言分类流程和步骤

朴素贝叶斯分类算法利用贝叶斯定理的优势，在群众留言分类中有广泛应用，是文本分类最为精确的技术。在智能文本分类技术中，通过贝叶斯分类器的“自我学习"智能技术，能有效保护信息的正常通信，过滤垃圾信息的骚扰。朴素贝叶斯分类分为以下3个阶段。

第1阶段:准备工作阶段。收集大量正常留言和垃圾留言作为样本，确定特征属性，并对每个特征属 性进行适当划分，然后提取留言样本中主题和信体中的字符串，建立对应的数据库分类,输出特征属性和训练样本。

第2阶段:分类器训练阶段。计算每个类别在训练样本中的出现频率及每个特征属性划分对每个类 别的条件概率估计，创建贝叶斯概率库统计出每个字串在群众留言中出现的概率以及在正常留言中出 现的概率，然后根据公式计算出留言中含有某字串则为垃圾留言的概率。

第3阶段:应用阶段。使用训练好的Bayes分类器对分类项进行分类，其输入是分类器和待分类项， 输出是待分类项与类别的映射关系。通过对留言样本的练,Bayes分类器可以自动获取垃圾留言的特征，并根据特征的变化，对群众留言进行有效分类。

2.2.3 具体分析

ROST CM 工具里的聚类分析：点击功能性分析下拉列表框中的聚类分析（测试模块）选项，打开聚类分析窗口，在待处理文本框中载入待类聚文件，然后填上类别数量，点击开始聚类即可对所选文件进行聚类分析。如下图：

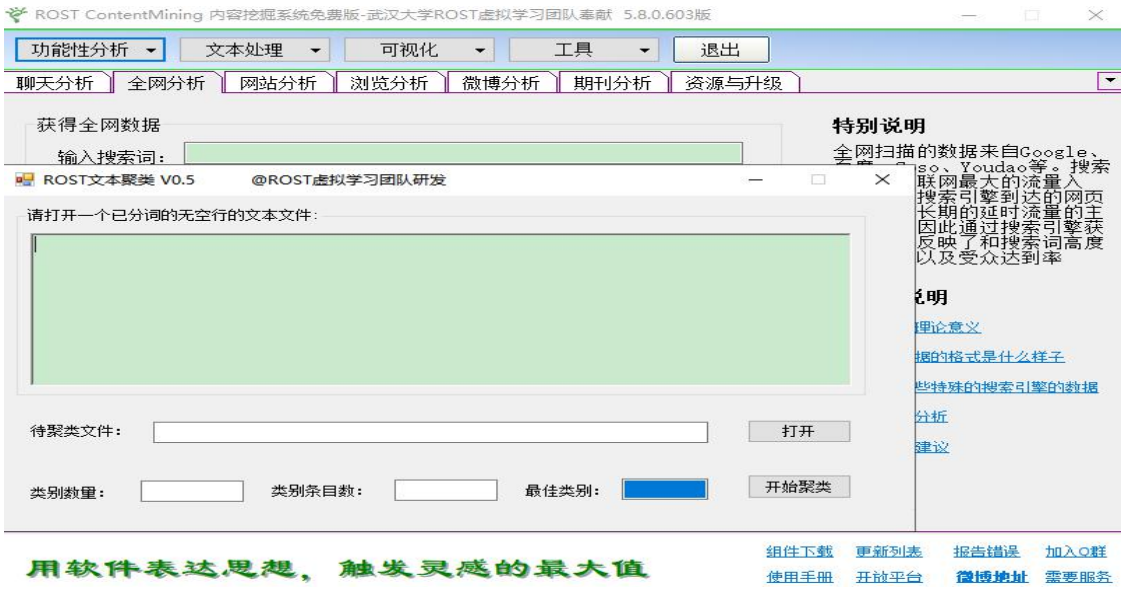


图 2.2.2 聚类分析界面

根据附件 3 中的用户留言信息表中的用户收视信息，利用 Excel 整理归纳出相关数据分布，如下图所示：

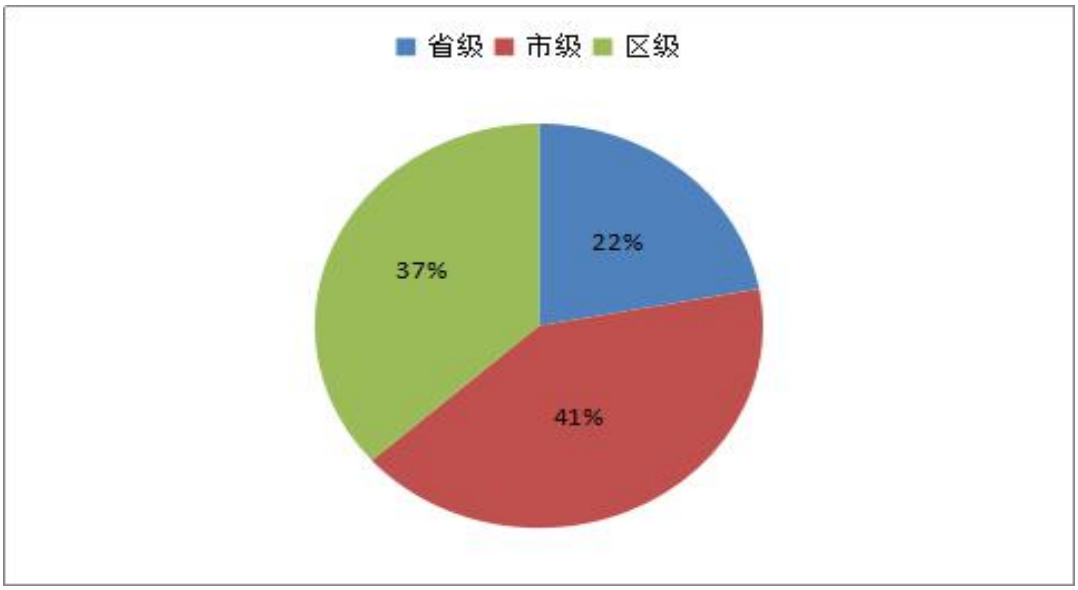


图 2.2.3 省、市、县三级留言数占比

省、市、县三级网民留言数分别占比如图，看得出市区网民发表意见的强烈表现，在留言率上处于领先。

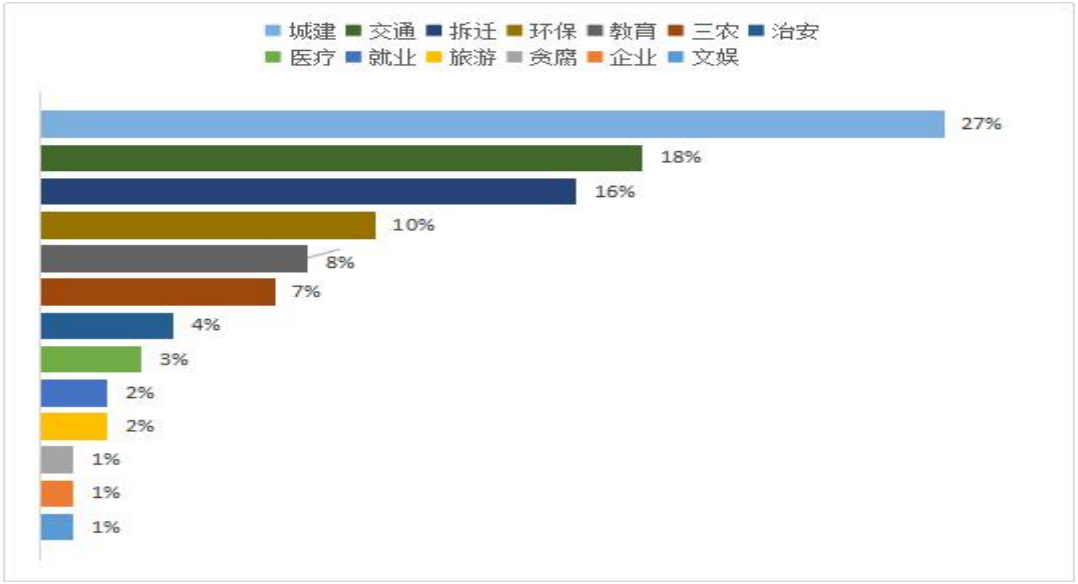


图 2.2.4 各个领域留言量分布

在城建、交通与拆迁领域的网民留言量最多，三个领域留言量一共占比超过 61%。随后，则是环保、教育、三农领域。留言量占比在

5%以下的领域则包括治安、医疗、就业等。

2.3 问题三分析方法与过程

2.3.1 流程图

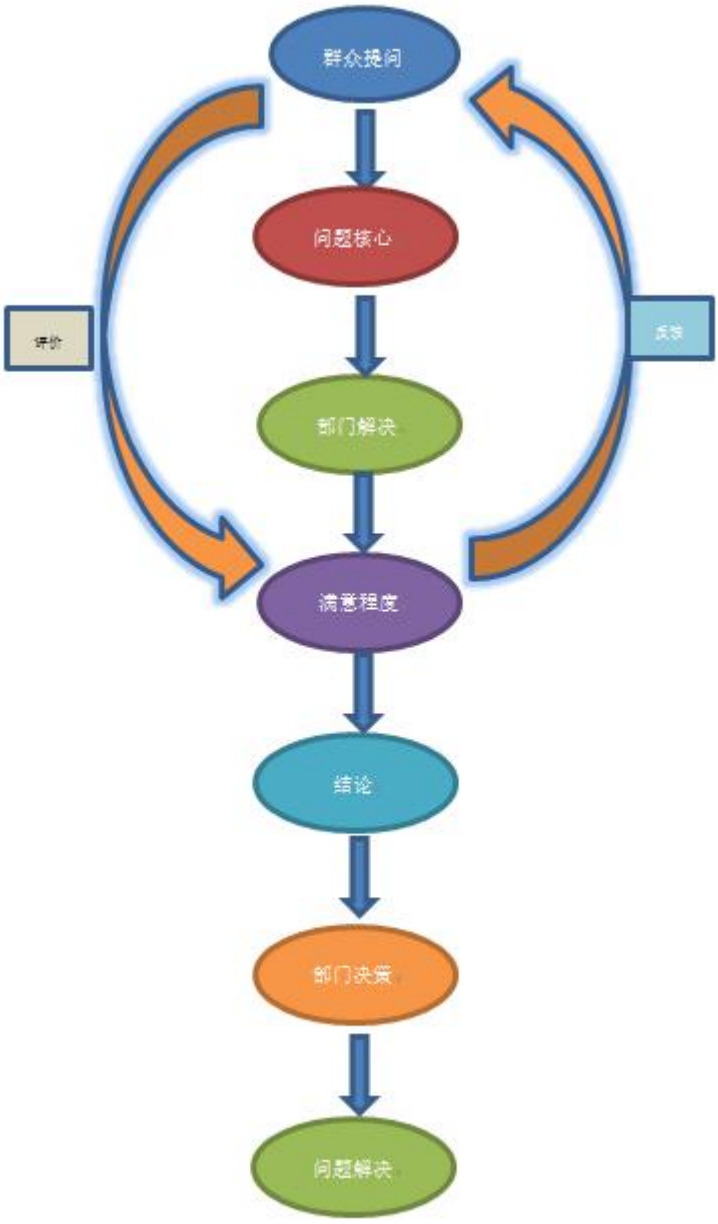
对附件 4 进行一级分类预处理，群众问题大致分为以下 15 类，再对群众提出的问题与部门的回答与解释进行对比，问题分类与解决流程如下。

图 2.3.1 问题分类与解决流程

2.3.2 数据预处理

对群众提出的问题，经过专一的分类处理，按照下图的操作流程，进行具体化的分析，对于相似度较高的答复，采用单一分析的方式，

对数据进行预处理，运用多边程序选择最符合群众的处理方法。



2.3.3 具体分析

合理化建议的推行与采纳能有效地凝聚部门的管理能力，增进群众对部门的信任，推广合理化建议是发挥部门的作用，凝聚部门的有效形式。积极地向群众提出合理化建议，也反映了部门对群众的忠诚，反映了群众对部门的热爱，也是部门凝聚力与向心力的一种良好的体

现。作为有责任感并寻求发展壮大的物业管理企业，必须在重视常规物业服务的基础上，寻找企业发展的增长点，但在企业的发展过程中必然存在一定的管理盲点与缺陷，如何有效可持续地发展，不断进行企业管理创新，才能建设好企业发展的平台，而在物业服务企业中推行合理化建设，将是突破企业管理瓶颈的有效载体。

下面我将从一个例子说明程序图的相关操作。例如在群中留言当中，群众诉说 261 公交车很少，上下班拥挤，等车时候比较长。其他公交车有很多趟，车内拥挤。这些情况说明：相关部门做出的回复是，261 路公交车全程 24km，配车 20 台。每班发车间距为七到八分钟每趟，平均为十到 15 分钟每趟。经过最期的查看，时间间隔正常，部门推迟说驾驶员工作时间长，劳动强度大。造成车队驾驶短缺，解决方法为现在极力调配人员实施该线路运力，大量招募驾驶人员，条件具备后将增加线路配车。

在这个问题当中。群众提出的关键词有拥挤，车少等车时间长。我们用表格算选出，问题的核心。再抽取主要关键词“车很少”。相关部门的答复路程短时间间隔短，每班车发车时间为七到八分钟。在解释方面说驾驶员工作时间长，劳动强度大。解决方法为：正在进行驾驶员招募，部门配车，没有进行具体的招募时间，实施方案和具体操作，所以答复不完整。

只有通过合理化建议的有效推行，积极倡导群众为部门多提合理化建议，通过合理化建议的采纳，提高了群众的效益，增进了部门的活力，那么部门就能产生一系列有效地经济与管理效益。从而增进企

业员工的责任感与荣誉感，激发群众的参与意识，增进部门的向心力。

当前，随着城市建设进程的加快，物业管理企业规模的也将进一步扩大，物业管理小区的规模也进一步扩大，小区的服务与管理也将进一步复杂化、科技化、精细化，相应地对小区的服务与管理提出了更高的要求。因此，为适应社会发展对物业服务与管理的新要求，在提高物业管理企业员工各项素质与技能的基础上，充分调动员工参与企业发展的积极性，发挥企业员工的主动参与精神，围绕企业发展目标，深入发动企业的管理人员、普通员工对物业小区服务与管理过程中存在的问题提出合理化建议，通过合理化。

3 结论

对于问题一给出的群众留言记录，我们用 Python 处理附件 2 数据使其简单化，再采用决策对数据进行分类，给出最佳的标签分类。

对于问题二，为了更大程度的收集群众最关心的问题，对问题获得最佳处理，先用 ROST CM 对数据进行分类，获得最大相似度，再用朴素贝叶斯算法进行数据的辅助处理，得出排名前五的热点问题。

对于问题三，采用语言描述，对相关部门做出的答复意见进行分类，再从答复中提取一定的相关性，用流程图进行相关的分析解释，做到对群众最好的答复。

4 参考文献

[1] 胡佳妮，徐蔚然，郭军，等．中文文本分类中的特征选择算法研究 [J]．光通信研究，2005(3)：44-46．

[2] HAN Jiawei，KAMBERM．数据挖掘概念与技术 [M]．北京：机械工业出版社，2010：184—211．

[3]田晓宇，梁静国．支持向量机在文本自动分类中的应用研究．情报学报．2006(2):208-214．

[4]唐小力，吕宏伟．SVM 增量学习算法研究．电脑知识与技术.2006(5):160-162．

[5]赵晖，荣莉莉．基于模糊核聚类的 SVM 多类分类方法．系统工程与电子技术.2006(5):770-774．

[6] VapnikVladimir N 著．张学工译．统计学习理论的本质．清华大学出版社．2000．9．

[7] Jiawei Han,，Micheline Kamber 著范明，孟小峰等译.数据挖掘:概念与技术，北京:机械工业出版社,2001: 3-6.

程序

第一题代码：

数据预处理代码：

```
def loadDataSet(fileName):
dataMat=[];labelMat=[];
data=[]
fr=open(fileName)
for line in fr.readlines():
 line=line.replace(',','\t')#去除逗号
line=line.replace('M','-1')#对标签进行替换
line=line.replace('R','1')
  lineArr=line.strip('\n').split('\t')#分割数据
    data.append([float(lineArr[i])
  for i in range(len(lineArr))]):
  random.shuffle(data)   #随机打乱列表
  for l in range(len(data)):
    dataMat.append(data[i][0:len(data)-1])
    labelMat.append(data[i][-1])
    return dataMat,labelMat
```

建立模型代码：

```
<code class="language-python">#svm 算法的实现
```

```
from numpy import*
import random
from time import*
def loadDataSet(fileName):#输出 dataArr(m*n),labelArr(1*m)其中 m 为数据集的个数
dataMat=[];labelMat=[]
fr=open(fileName)
for line in fr.readlines():
lineArr=line.strip().split('\t')#去除制表符，将数据分开
dataMat.append([float(lineArr[0]),float(lineArr[1])])#数组矩阵
labelMat.append(float(lineArr[2]))#标签
return dataMat,labelMat
def selectJrand(i,m):#随机找一个和 i 不同的 j
j=i
while(j==i):
j=int(random.uniform(0,m))
return j
def clipAlpha(aj,H,L):#调整大于 H 或小于 L 的 alpha 的值
if aj>H:
aj=H
if aj<L:
aj=L
```

```
        return aj
    def smoSimple(dataMatIn,classLabels,C,toler,maxIter):
dataMatrix=mat(dataMatIn);labelMat=mat(classLabels).transpose()#
转置
    b=0;m,n=shape(dataMatrix)#m 为输入数据的个数，n 为输入向
量的维数
    alpha=mat(zeros((m,1)))#初始化参数，确定 m 个 alpha
iter=0#用于计算迭代次数
    while (iter<maxIter):#当迭代次数小于最大迭代次数时(外循环)
    alphaPairsChanged=0#初始化 alpha 的改变量为 0
for i in range(m):#内循环
fXi=float(multiply(alpha,labelMat).T*\
(dataMatrix*dataMatrix[i,:].T))+b#计算 f(xi)
Ei=fXi-float(labelMat[i])#计算 f(xi)与标签之间的误差
if ((labelMat[i]*Ei<-toler)and(alpha[i]<C))or\
((labelMat[i]*Ei>toler)and(alpha[i]>0)):#如果可以进行优化
j=selectJrand(i,m)#随机选择一个 j 与 i 配对
fXj=float(multiply(alpha,labelMat).T*\
(dataMatrix*dataMatrix[j,:].T))+b#计算 f(xj)
Ej=fXj-float(labelMat[j])#计算 j 的误差
alphaIold=alpha[i].copy()#保存原来的 alpha(i)
alphaJold=alpha[j].copy()
```

```
if(labelMat[i]!=labelMat[j]):#保证 alpha 在 0 到 c 之间
L=max(0,alpha[j]-alpha[i])
H=min(C,C+alpha[j]-alpha[i])
else: L=max(0,alpha[j]+alpha[i]-C)
H=min(C,alpha[j]+alpha[i])
if L==H:print('L=H');continue
eta=2*dataMatrix[i,:]*dataMatrix[j,:].T-\
dataMatrix[i,:]*dataMatrix[i,:].T-\
dataMatrix[j,:]*dataMatrix[j,:].T
if eta>=0:print('eta=0');continue
  alpha[j]-=labelMat[j]*(Ei-Ej)/eta
alpha[j]=clipAlpha(alpha[j],H,L)#调整大于 H 或小于 L 的 alpha
if (abs(alpha[j]-alphaJold)<0.0001):
print('j not move enough');continue
alpha[i]+=labelMat[j]*labelMat[i]*(alphaJold-alpha[j])
b1=b-Ei-labelMat[i]*(alpha[i]-alphaIold)*\
dataMatrix[i,:]*dataMatrix[i,:].T-\
labelMat[j]*(alpha[j]-alphaJold)*\
dataMatrix[i,:]*dataMatrix[j,:].T#设置 b
  b2=b-Ej-labelMat[i]*(alpha[i]-alphaIold)*\
dataMatrix[i,:]*dataMatrix[i,:].T-\
labelMat[j]*(alpha[j]-alphaJold)*\
```

```
dataMatrix[j,:]*dataMatrix[j,:].T                    if
(0<alpha[i])and(C>alpha[j]):b=b1
elif(0<alpha[j])and(C>alpha[j]):b=b2
else:b=(b1+b2)/2
alphaPairsChanged+=1
print('iter:%d i:%d,pairs changed%d'%(iter,i,alphaPairsChanged))
if (alphaPairsChanged==0):iter+=1
else:iter=0
print('iteraction number:%d'%iter)
return b,alpha   #定义径向基函数
def kernelTrans(X, A, kTup):#定义核转换函数（径向基函数）
m,n = shape(X)
K = mat(zeros((m,1)))
if kTup[0]=='lin': K = X * A.T     #线性核 K 为 m*1 的矩阵
elif kTup[0]=='rbf':
for j in range(m):
deltaRow = X[j,:] - A
K[j] = deltaRow*deltaRow.T
K = exp(K/(-1*kTup[1]**2)) #divide in NumPy is element-wise not
matrix like Matlab
else: raise NameError('Houston We Have a Problem -- \
 That Kernel is not recognized')
```

```
return K
class optStruct:
def __init__(self,dataMatIn, classLabels, C, toler, kTup):    # Initialize the
structure with the parameters
self.X = dataMatIn
self.labelMat = classLabels
self.C = C
self.tol = toler
self.m = shape(dataMatIn)[0]
self.alphas = mat(zeros((self.m,1)))
self.b = 0
self.eCache = mat(zeros((self.m,2))) #first column is valid flag
self.K = mat(zeros((self.m,self.m)))
for i in range(self.m):
self.K[:,i] = kernelTrans(self.X, self.X[i,:], kTup)
def calcEk(oS, k):            fXk =
float(multiply(oS.alphas,oS.labelMat).T*oS.K[:,k] + oS.b)
Ek = fXk - float(oS.labelMat[k])
return Ek
def selectJ(i, oS, Ei):
maxK = -1;
maxDeltaE = 0; Ej = 0
```

```
oS.eCache[i] = [1,Ei]
validEcacheList = nonzero(oS.eCache[:,0].A)[0]
if (len(validEcacheList)) > 1:
for k in validEcacheList:      #loop through valid Ecache values and
find the one that maximizes delta E
if k == i: continue #don't calc for i, waste of time
Ek = calcEk(oS, k)
deltaE = abs(Ei - Ek)
if (deltaE > maxDeltaE):
maxK = k;
maxDeltaE = deltaE; Ej = Ek
return maxK, Ej
else:      #in this case (first time around) we don't have any valid
eCache values
j = selectJrand(i, oS.m)
Ej = calcEk(oS, j)
return j, Ej
def updateEk(oS, k):#after any alpha has changed update the new
value in the cache
Ek = calcEk(oS, k)
oS.eCache[k] = [1,Ek]
def innerL(i, oS):
```

```
Ei = calcEk(oS, i)
if ((oS.labelMat[i]*Ei < -oS.tol) and (oS.alphas[i] < oS.C)) or
((oS.labelMat[i]*Ei > oS.tol) and (oS.alphas[i] > 0)):
j,Ej = selectJ(i, oS, Ei) #this has been changed from selectJrand
alphaIold = oS.alphas[i].copy();
  alphaJold = oS.alphas[j].copy()
if (oS.labelMat[i] != oS.labelMat[j]):
L = max(0, oS.alphas[j] - oS.alphas[i])
H = min(oS.C, oS.C + oS.alphas[j] - oS.alphas[i])
else:                          L
  = max(0, oS.alphas[j] + oS.alphas[i] - oS.C)
  H = min(oS.C, oS.alphas[j] + oS.alphas[i])
if L==H: print("L==H"); return 0
eta = 2.0 * oS.K[i,j] - oS.K[i,i] - oS.K[j,j] #changed for kernel
if eta >= 0: print("eta>=0"); return 0
oS.alphas[j] -= oS.labelMat[j]*(Ei - Ej)/eta
oS.alphas[j] = clipAlpha(oS.alphas[j],H,L)
updateEk(oS, j) #added this for the Ecache                    if
(abs(oS.alphas[j] - alphaJold) < 0.00001):
print("j not moving enough");
return 0
oS.alphas[i] += oS.labelMat[j]*oS.labelMat[i]*(alphaJold -
```

```
oS.alphas[j])#update i by the same amount as j
updateEk(oS, i) #added this for the Ecache
#the update is in the oppostie direction
b1 = oS.b - Ei- oS.labelMat[i]*(oS.alphas[i]-alphaIold)*oS.K[i,i] -
oS.labelMat[j]*(oS.alphas[j]-alphaJold)*oS.K[i,j]
b2 = oS.b - Ej- oS.labelMat[i]*(oS.alphas[i]-alphaIold)*oS.K[i,j]-
oS.labelMat[j]*(oS.alphas[j]-alphaJold)*oS.K[j,j]
if (0 < oS.alphas[i]) and (oS.C > oS.alphas[i]): oS.b = b1                elif
(0 < oS.alphas[j]) and (oS.C > oS.alphas[j]): oS.b = b2                else:
oS.b = (b1 + b2)/2.0
return 1
else: return 0    #smoP 函数用于计算超平的 alpha,b    def
smoP(dataMatIn, classLabels, C, toler, maxIter,kTup=('lin', 0)):
#完整的 Platter SMO
oS = optStruct(mat(dataMatIn),mat(classLabels).transpose(),C,toler,
kTup)
iter = 0#计算循环的次数
entireSet = True; alphaPairsChanged = 0
while (iter < maxIter) and ((alphaPairsChanged > 0) or (entireSet)):
alphaPairsChanged = 0
if entireSet:     #go over all
for i in range(oS.m):
```

```
alphaPairsChanged += innerL(i,oS)
print("fullSet, iter: %d i:%d, pairs changed %d" %
(iter,i,alphaPairsChanged))
iter += 1
else:#go over non-bound (railed) alphas
nonBoundIs = nonzero((oS.alphas.A > 0) * (oS.alphas.A < C))[0]
for i in nonBoundIs:
alphaPairsChanged += innerL(i,oS)
print("non-bound, iter: %d i:%d, pairs changed %d" %
(iter,i,alphaPairsChanged))
iter += 1
if entireSet: entireSet = False #toggle entire set loop
elif (alphaPairsChanged == 0): entireSet = True
print("iteration number: %d" % iter)
return oS.b,oS.alphas    #calcWs 用于计算权重值 w
def calcWs(alphas,dataArr,classLabels):#计算权重 W
X = mat(dataArr); labelMat = mat(classLabels).transpose()
m,n = shape(X)
w = zeros((n,1))
for i in range(m):
w += multiply(alphas[i]*labelMat[i],X[i,:].T)
return w       #值得注意的是测试准确与 k1 和 C 的取值有关。
```

```
def testRbf(k1=1.3):#给定输入参数 K1          #测试训练集上的准确率
dataArr,labelArr = loadDataSet('testSetRBF.txt')#导入数据作为训练集
b,alphas = smoP(dataArr, labelArr, 200, 0.0001, 10000, ('rbf', k1))
#C=200 important
datMat=mat(dataArr); labelMat = mat(labelArr).transpose()
svInd=nonzero(alphas.A>0)[0]#找出 alphas 中大于 0 的元素的位置
#此处需要说明一下 alphas.A 的含义          s
Vs=datMat[svInd] #获取支持向量的矩阵，因为只要 alpha 中不等于 0 的元素都是支持向量
labelSV = labelMat[svInd]#支持向量的标签
print("there are %d Support Vectors" % shape(sVs)[0])#输出有多少个支持向量
m,n = shape(datMat)#数据组的矩阵形状表示为有 m 个数据，数据维数为 n
errorCount = 0#计算错误的个数
for i in range(m):#开始分类，是函数的核心
kernelEval = kernelTrans(sVs,datMat[i,:],('rbf', k1))#计算原数据集中各元素的核值
predict=kernelEval.T * multiply(labelSV,alphas[svInd]) + b#计算预测结果 y 的值
if sign(predict)!=sign(labelArr[i]): errorCount += 1#利用符号判断类
```

别 ### sign（a）为符号函数：若 a>0 则输出 1，若 a<0 则输出-1.###

print("the training error rate is: %f" % (float(errorCount)/m)) #2、测试测试集上的准确率

dataArr,labelArr = loadDataSet('testSetRBF2.txt')

errorCount = 0

datMat=mat(dataArr)#labelMat = mat(labelArr).transpose()此处可以不用

m,n = shape(datMat)

for i in range(m):

kernelEval = kernelTrans(sVs,datMat[i,:],('rbf', k1))

predict=kernelEval.T * multiply(labelSV,alphas[svInd]) + b

if sign(predict)!=sign(labelArr[i]): errorCount += 1

print("the test error rate is: %f" % (float(errorCount)/m)) def main():

t1=time()

dataArr,labelArr=loadDataSet('testSet.txt')

b,alphas=smoP(dataArr,labelArr,0.6,0.01,40)

ws=calcWs(alphas,dataArr,labelArr)

testRbf()

t2=time()

print("程序所用时间为%ss"%(t2-t1))

if __name__=='__main__':

main()</code>