第八届“泰迪杯”数据挖掘挑战赛

“智能政务”文本中的挖掘应用

摘要

目前，人工智能已经在公共信息领域中深入应用，政府也在向“互联网+政务”方向转变。问政平台信息增长迅速的同时也引出了问题，人工操作远不及机器算法分析的快，所以我们利用自然语言处理和数据挖掘方法解决以下问题。

问题一使用 jieba 分词算法，将用户信息转化为词组，找到基于词频的最大切分组合，将分好的数据代入文本监督学习模型。我们训练出七种常见的分类算法模型进行测试和对比，成功编写文本分词算法及一级标签分类算法模型，通过实验发现 lightGBM 算法与本问题匹配度较高，运行准确度为 86.3%，耗时 6.6 秒，远高于其他文本分类算法。

问题二则使用了 word2Vec 对留言主题进行文本相似度计算，得出该留言相似的留言数量和相应序号。使用层次分析算法对留言时间段内的点赞数、反对数和文本相似度进行权重对比，计算得出留言热度评分并筛选出热度前五的留言主题。

问题三编写算法，计算出留言答复的相关性、完整性、可解释性和时间差。使用权重分析方法计算出答复质量得分，构建出答复质量评价模型。

关键词：自然语言处理； jieba 分词算法； lightGBM； Word2Vec； 层次分析算法；

目录

一、引言

目前，人工智能已经在公共信息领域中深入应用，政府也逐步转成“群众需求为中心，大众服务为方向”的智能政府，利用人工智能算法，组建高效智能的服务目前，人工智能已经在公共信息领域中深入应用，政府也逐步转成“群众需求为中心，大众服务为方向”的智能政府，利用人工智能算法，组建高效智能的服务体系，推动社会向智能社会迈进[1]。近年来，海量数据多元化，而准确全面的有效数据对政府部门制定可行性决策起决定作用，否则会导致政府不能及时听取民意，做出错误决策[2]。

自然语言处理技术就是提取有效数据的得力工具，它是融合了计算机科学、数学与语言学于一体的学术技术，将人们日常生活中使用的语言进行词性标注和语法解析，再进行语义分析和情感分析[3]。我们将这种技术应用到政务文本挖掘中。

广大群众在微信、微博、电子信箱等网络问政平台上就社区、生活等问题进行投稿提议，表达自己对社会问题的看法与建议。这些线上平台拉近了群众与政府的距离，方便政府及时了解民意、汇聚民气，但是无形之中也增加了难点，给以往依靠人工来进行留言划分和热点问题整理的相关部门带来了巨大的工作量考验，为解决上述问题，创建新型智能政务，我们利用自然语言处理技术的强有力优势，对这些投稿信息进行处理归纳，减少部门人员的工作量，既能快速的收集到民意也能准确的分析出大众最想表达的想法和需求。

群众留言分类归纳后便于统计热点问题，能够凸显出广大群众的紧迫需求和积极建议。对此，我们运用机器算法来统计不同分类项的权重比例。每个人的需求不同，留言动机也不同，单看留言分类不能及时了解民意，统计民情，进而将各项留言分类进行排序筛选，选出最优留言分区，政府及时掌握热点问题，有助于任务分配到位，提高服务效率。

智能政务是当前政府的奋斗目标，通过运用大数据、云计算等先进技术，打造优质服务产品一体化，促使公众参与到政民互动的平台中，实现“互联网+政务”，成为便民利国的好帮手[4]。

二、实验方案

2.1 问题重述

问政平台的发展产生了庞大的数据量，为以往依靠人工处理的问题带来了诸多不便，利用大数据技术打造智慧政务平台成为了解决该问题的有效手段，为政务部门节约工作时间、提高工作效率。本次实验根据互联网群众问政数据，使用自然语言处理和文本挖掘的方法解决以下三个问题。

2.1.1 问题一

工作人员在解决群众留言问题前，需要根据附件 1 中的类别标签，手动划分留言的类别。由于数据量庞大，人工效率较低，本问题通过附件 2 中群众的留言数据以及工作人员已经划分好的一级标签，通过对留言内容的识别及分类算法的建立，训练出自动划分留言分类的**一级标签分类模型**。

2.1.2 问题二

群众通过留言反映出社会中存在的问题，在此类问题中就存在**每一时间段内反映次数较多、获赞量较大**的“热点问题”，及时发现热点问题有利于相关部门更有效的服务群众。本问题通过附件 3 提供的群众留言数据，建立评价留言热度的**权重**指标，建立**热度评价模型**并挖掘出热度排名前 5 的热点问题及相关内容的所有留言信息。

2.1.3 问题三

收到留言后，相关部门会根据留言内容对群众的留言进行答复。本问题根据向相关部门的答复内容，从**相关性、完整性、可解释性、答复效率**四方面进行答复内容的评价。相关性即答复内容与留言内容的相关程度；完整性即答复内容是否完整、符合规范；可解释性即答复内容是否解决了群众留言中的问题；答复效率即相关部门回复的及时程度。

2.2 思路分析

2.2.1 问题一

观察附件 2 中表结构，群众留言信息分别包括留言编号、留言用户 ID、留言主题、留言时间、留言详情及一级标签六列，如表 1 所示。

表 1 附件 2 数据

留言编号	留言用户	留言主题	留言时间	留言详情	一级分类
6	A00043564	为什么处…	2020/1/7 21:31:53	2017 我从省…	劳动和社会保障
37	A000107866	A 市在水…	2020/1/4 11:17:46	位于书院路…	城乡建设
74432	A00063499	A 市经开…	2019/4/14 10:58:09	五年级的女…	教育文体

通过对数据的探索，取留言详情列数据作为训练数据，取一级标签列数据作为训练结果。

① 留言详情

根据附件二中的留言详情列数据，对留言文本进行了分词处理。

首先进行数据预处理，去除“ ”“，”“。”“、”“；”等特殊字符。其次为了结果的准确性建立关于留言详情的停用词表，去除分词中的停用词，最终利用 jieba 库中的 cut 方法对留言详情进行分词，实现文本内容的切割。

② 一级标签

观察留言数据知，附件 2 中共有 7 类一级标签，分别为“城乡建设”、“环境保护”、“交通运输”、“教育文体”、“劳动和社会保障”、“商贸旅游”及“卫生计生”，共 9210 条，选取 70%的数据作为训练集数据，共 6449 条，取剩下 30%的数据作为测试集数据，共 2761 条，如表 2 所示。

表 2 一级标签词条数量

一级标签	留言数	训练集	测试集
城乡建设	2009	602	1407
环境保护	938	281	657
交通运输	613	184	429
教育文体	1589	476	1113
劳动和社会保障	1969	590	1379
商贸旅游	1215	365	850
卫生计生	877	263	614
共计	9210	2761	6449

③ 文本分类算法选择

选取 7 种常用的分类算法，编写机器学习监督算法对模型进行训练，分别为“逻辑回归”算法、“多项式贝叶斯”算法、“#伯努利贝叶斯”算法、“lightGBM”算法、“随机森林”算法、“高斯贝叶斯”算法、“决策树”算法。计算每个算法的准确率、混淆矩阵、召回率以及运算时间，选择最优算法对留言信息进行一级标签分类模型的训练。

2.2.2 问题二

观察附件 3 中表结构，群众留言信息分别包括“留言编号”、“留言用户”、“留言主题”、“留言时间”、“留言详情”、“反对数”及“点赞数”七列，如表 3 所示。

表 3 附件 3 数据

留言编号	留言用户	留言主题	留言时间	留言详情	点赞数	反对数
360103	A0012425	A5 区劳…	2019/9/25 0:31:33	A5 区劳动…	1	0
360108	A0283523	A5 区劳…	2019/8/1 16:20:02	局长:你好…	6	0
360101	A324156	A5 区劳…	2019/7/28 12:49:18	尊敬的政…	4	0

针对附件 3 中的留言详情进行文本相似度匹配，采用 word2vec 算法。首先对附件 3 中的留言详情进行分词操作目的是将字符文本转为 Gensim 模型所能理解的稀疏矩阵，并利用分词后的文本建立语料库。采用较优的 Gensim 模型对语料库进行模型训练建立适合本题的模型。其中为了准确识别语义将 min_count 设置为 1。其次将训练好的模型应用于以分词的留言详情计算出各文本间的相似度得分，将得分大于 0.8 的进行保留以便找出与之相似的文本。

根据留言时间间隔、点赞数与反对数的值以及上述求得的相似度的值，为各个指标分配权重，算出热度得分，观察得分结果，找出评分最高的 5 项并依次查找相关的数据信息。

2.2.3 问题三

观察附件 4 中表结构，群众留言信息分别包括“留言编号”、“留言用户”、“留言主题”、“留言时间”、“留言详情”、“答复意见”及“答复时间”七列，如表 4 所示。

表 4 附件 4 数据

留言编号	留言用户	留言主题	留言时间	留言详情	答复意见	答复时间
2549	A00045581	A2 区景…	2019/4/25 9:32:09	2019 年 4 月…	您好,首先…	2019/5/10 14:56:53
35467	A00044412	请求农行…	2019/11/2 15:41:22	“大行德广”…	尊敬的网…	2019/12/2 16:37:17
39494	A00012216	醴娄高速…	2019/12/1 10:49:07	沪昆高速扩…	您好!您…	2019/12/31 5:23:30

根据所给数据，计算相关性、完整性、可解释性、答复效率四个指标的得分，计算思路如下：

① 相关性

对留言详情和答复意见进行语义上的比较，判断所答是否所问。利用word2vec 算法进行语义相关的比较。首先对分词后的留言详情和答复意见建立语料表并利用 Gensim 模型对语料库进行模型训练。应用以训练好的模型进行答复意见和语料库中留言详情进行比较，其中建立一个全局变量准确控制答复意见和留言详情的对应性并最终保留得分。

② 完整性

主要通过查看句子结构是否完整给出相应的得分。关于答复意见结构完整性的判断通过建立完整性表查看答复意见中是否存在必要的称呼如“网友”“市民”等，是否存在问候语如“您好”“你好”等，是否对问题进行了相应的答复如“回答/答复如下”“我们的建议是”，是否表示感谢如“谢谢”“感谢”等。最后根据合理的尺度得出答复意见完整性的分数。

③ 可解释性

”解释“即在观察的基础上进行思考即指答复意见给出有关的建议。由此将可解释性的评判标准设置为 0-1 评判。0 表示没有给出相应的建议，1 表示给出相应建议。对于是否给出相应建议的计算，通过观察数据得知，绝大多数都给出相应建议，没有给出建议的有一定规律例如回复仅为“已收悉”、“网友您好留言已收悉”、“您好你所反映的问题已转交相关单位调查处置”等，或只给出日期如“2016 年 6 月 12 日”、“2019 年 1 月 14 日”等。建立回复无关性表找出以上答复意见将其可解释性设置为 0，其余为 1。

④ 答复效率

对于留言答复质量的评价，留言与答复的时间差可反省答复的效率，是一个评价答复质量的重要指标。对于留言与答复时间差的计算，首先将留言时间与答复时间转为时间类型，由于留言时间与答复时间都存在着大量的格式不统一如“2019-1-12”“2018/2/14”“2017/3/21 13:32:09”等几种不同形式。采用或略时间的方法保证数据的完整性。并利用 try-except 将进行 XXXX-XX-XX 和“XXXX/XX/XX”进行格式上的统一。最终算得留言时间与答复时间的时间差。

计算出指标得分后，分配权重，计算答复质量的评分，判断本次答复质量。

三、实验过程

3.1 问题一

编写 python 代码实现文本切割并建立一级标签分类模型

3.1.1 文本切割处理

对于文本的切割主要进行去除异常字符例如“*****”、“ ”，建立适用于本题的停用词表，在基本停用词表的基础上加入了如“感谢”、“您好”、“尊敬”等频繁出现且与本题无关的词，利用 jieba 分词，最终将结果保留到新 excel 文件中。

对于异常值的处理，建立 format_str()方法进行异常字符的剔除，由于文本中存在未知的异常字符且其他字符与本题目无关，所以利用 is_chinese 只对中文进行了保留,如图 1 所示。

```
42      # 删除异常字符
43      def is_chinese(uchar):
44          """判断一个unicode是否是汉字"""
45          if uchar >= u'\u4e00' and uchar <= u'\u9fa5':
46              return True
47          else:
48              return False
49      def format_str(content):
50          content_str = ''
51          for i in content:
52              if is_chinese(i):
53                  content_str = content_str+i
54          return content_str
55      #去除多余字符
56      captalizer1 = lambda x: format_str(x)
57      df["留言详情"]=df["留言详情"].apply(captalizer1 )
```

图 1

针对分词操作，首先利用 jieba 库的 cut 方法进行分词，得到如“大道 西行 便道”的结果，其次将文本中的停用词去除，最终返回分词结果，如图 2 所示

```
# 对句子进行中文分词
def seg_depart(sentence):
    # 对文档中的每一行进行中文分词
    print("正在分词")
    sentence_depart = jieba.cut(sentence.strip())
    # 创建一个停用词列表
    stopwords = stopwordslist()
    # 输出结果为outstr
    outstr = ''
    # 去停用词
    for word in sentence_depart:
        if word not in stopwords:
            if word != '\t':
                outstr += word
                outstr += " "
    return outstr
#分词
captalizer2 = lambda x: seg_depart(x)
df["留言详情"]=df["留言详情"].apply(captalizer2 )
```

图 2

最终将留言详情的分词结果“留言详情”和提取标签中用到的“一级标签”保存在新的 excel 中，如图 3 所示。

```
test={"留言详情":df["留言详情"],"一级分类":df["一级标签"]}
df1=pd.DataFrame(data=test)
df1.to_excel("D://学习//数学建模//泰迪杯//c//分词结果.xlsx",index=0)
print(df1)
```

图 3

3.1.2 训练一级标签分类模型

编写监督学习算法，分别带入 7 种常见算法计算准确度及运行时间，选择出一种最适合的算法进行模型的训练。

① 导入数据

```

#导入数据
train_data = pd.read_csv('data_train.csv', lineterminator='\n')
test_data = pd.read_csv('data_text.csv', lineterminator='\n')

```

图 4

② 对数据标签进行格式化处理

```
18  #利用LabelEncoder对数据标签进行规格化处理
19  def encodeLabel(data):
20      listLable = []
21      for lable in data['lable']:
22          listLable.append(lable)
23      le = LabelEncoder()
24      resultLable = le.fit_transform(listLable)
25      return resultLable
26
27  trainLable=encodeLabel(train_data)
28  testLable=encodeLabel(test_data)
```

图 5

③ 获取所有 review 的集合

```
30  #获取所有review的集合:
31  def getReview(data):
32      listReview=[]
33      le = LabelEncoder()
34      for review in data['review']:
35          listReview.append(review)
36      return listReview
37
38  trainReview=getReview(train_data)
39  testReview=getReview(test_data)
```

图 6

④ 计算频数向量：

```
41  #计算频数向量:
42  stoplist=['.', '?', '!', ':', '-', '+', '/', '"', ',',
43  cv=CountVectorizer(stop_words=stoplist)
44  wordBag=trainReview+testReview
45  cv.fit(wordBag)
46  test_count = cv.transform(testReview)
47  testCount = test_count.toarray()
48  train_count = cv.transform(trainReview)
49  trainCount = train_count.toarray()
```

图 7

⑤ 算法测评函数

```
#评测函数:
def classificate(estimator, trainReview, trainLable, testReview, testLable):
    start = time.time()
    #模型训练,fit通常都是指模型的学习、训练过程
    print('训练:')
    model = estimator
    model.fit(trainReview, trainLable)
    print(model)
    #模型预测:
    print('预测:')
    pred_model = model.predict(testReview)
    print(pred_model)
    #算法评估
    print('评估:')
    score = metrics.accuracy_score(testLable, pred_model)
    matrix = metrics.confusion_matrix(testLable, pred_model)
    report = metrics.classification_report(testLable, pred_model)

    print('>>>准确率\n', score)
    print('\n>>>混淆矩阵\n', matrix)
    print('\n>>>召回率\n', report)
    end = time.time()
    t = end - start
    print('\n>>>算法消耗时间为：', t, '秒\n')
```

图 8

⑥ 算法调用函数

```
#算法调用:
#逻辑回归
log = LogisticRegression()
classificate(log, trainCount, trainLable, testCount, testLable)
#多项式贝叶斯
mul = MultinomialNB()
classificate(mul, trainCount, trainLable, testCount, testLable)
#伯努利贝叶斯
ber = BernoulliNB()
classificate(ber, trainCount, trainLable, testCount, testLable)
#lightGBM
lightGBM = lg.LGBMClassifier()
classificate(lightGBM, trainCount, trainLable, testCount, testLable)
#随机森林
random = RandomForestClassifier()
classificate(random, trainCount, trainLable, testCount, testLable)
#高斯贝叶斯
gau = GaussianNB()
classificate(gau, trainCount, trainLable, testCount, testLable)
#决策树
dec = DecisionTreeClassifier()
classificate(dec, trainCount, trainLable, testCount, testLable)
```

图 9

⑦ 保存模型

```
102  #保存模型
103  from sklearn.externals import joblib
104  joblib.dump(lightGBM,'lightGBM.pkl')
```

图 10

3.2 问题二

3.2.1 文本相似度匹配

关于文本相似度匹配，采用 word2vec 算法。首先对附件 3 中的留言详情进行进行分词操作，将字符文本转为 Gensim 模型所能理解的稀疏矩阵，并利用分词后的文本建立语料库。采用 Gensim 模型对语料库进行模型训练建立适合本题的模型。其次将训练好的模型应用于以分词的留言详情和语料库，将得分大于 0.8 的进行保留视为相似文本。

对于模型的训练，先用 ylk 函数进行留言详情语料库的建立，其中先用 jieba 对留言详情进行分词操作目的是将字符文本转为 Gensim 模型所能理解的稀疏矩阵。之后采用较优的 Gensim 模型对语料库进行模型训练建立适合本题的模型。如图

```
8   def is_chinese(uchar):
9       """判断一个unicode是否是汉字"""
10      if uchar >= u'\u4e00' and uchar <= u'\u9fa5':
11          return True
12      else:
13          return False
14  def is_number(uchar):
15      """判断一个unicode是否是数字"""
16      if uchar >= u'\u0030' and uchar <= u'\u0039':
17          return True
18      else:
19          return False
20  def is_alphabet(uchar):
21      """判断一个unicode是否是英文字母"""
22      if (uchar >= u'\u0041' and uchar <= u'\u005a') or (uchar >= u'\u0061' and uchar <= u'\u007a'):
23          return True
24      else:
25          return False
26  def format_str(content):
27      content_str = ''
28      for i in content:
29          if is_chinese(i):
30              content_str = content_str+i
31          elif is_alphabet(i):
32              content_str = content_str + i
33          elif is_number(i):
34              content_str = content_str + i
35      return content_str
36  captalizer1 = lambda x: format_str(x)
37  df["留言详情"]=df["留言详情"].apply(captalizer1 )
```

图 11

对于模型的训练，先用 ylk 函数进行留言详情语料库的建立，其中先用 jieba 对留言详情进行分词操作目的是将字符文本转为 Gensim 模型所能理解的稀疏矩阵。之后采用较优的 Gensim 模型对语料库进行模型训练建立适合本题的模型。如图

```
def ylk(x):
    seg = jieba.cut(x, cut_all=False)
    # s = ''.join(seg)
    # # m = list(s)
    with open('D://学习//数学建模//泰迪杯//c//word2vec//listTwo.txt', 'a',encoding='utf-8')as f:
        for word in seg:
            f.write(word+" ")
        f.write('\n')
captalizer2 = lambda x: ylk(x)
df["留言详情"]=df["留言详情"].apply(captalizer2)

from gensim.models.word2vec import LineSentence, Word2Vec
sentences = LineSentence("D:/学习/数学建模/泰迪杯/c/word2vec/listTwo.txt")
model = Word2Vec(sentences, min_count=1, iter=1000)
model.save("D:/学习/数学建模/泰迪杯/c/word2vec/w2v.mod")
```

图 12

最终，应用模型将附件 3 中的留言详情与语料库进行相似度计算，将得分大于 0.8 的保留到列表 res 中，将结果写入 excel。如图

```
with open(target, encoding='utf-8')as f:
    for line in f:
        candidates.append(line.strip().split()) #将语料放到列表中便于操作
def xsd(text):
    words = list(jieba.cut(text.strip())) #分词
    flag = False
    res = []
    index = 0
    for candidate in candidates:
        # print("candidate", candidate)
        for c in candidate:
            if c not in model_w2v.wv.vocab:
                print("candidate word %s not in dict. skip this turn" % c)
                flag = True
        if flag:
            break
        score = model_w2v.n_similarity(words, candidate)
        # resultInfo = {'id': index, "score": score, "text": " ".join(candidate)}
        # res.append(resultInfo)
        if score>0.8 :
            res.append(index)
        index += 1
    return (len(res),res)

df["s"]=0
captalizer2 = lambda x: xsd(x)
p=df["留言详情"].apply(captalizer2)
df["s"]=p
df.to_excel("D://学习//数学建模//泰迪杯//c//文本相似度匹配结果.xlsx")
```

图 13

3.2.2 热点问题得分计算

关于热点问题的计算，采用三个指标进行计算分别为时间间隔，点赞数和反对数，相关度 s。通过分配不同权重更合理的选择热点问题。

3.3 问题三

3.3.1 答复质量评价指标计算

针对问题 3 对答复意见的评价，采用 4 个指标进行评价分别为相关性，完整性，可解释性，相隔时间。对相关性的评价，进行留言详情和答复意见进行语义上的比较，判断所答与所问的对应关系。对于完整性的计算，主要通过查看句子结构是否完整给出相应的得分，根据合理的尺度得出答复意见完整性的分数。可解释性的判断，建立回复无关性表找出与留言无关的答复意见将其可解释性设置为 0，其余为 1。相隔时间则采用留言与答复时间的时间差进行计算。

① 相关性：

首先对分词后的留言详情和答复意见建立语料表并利用 Gensim 模型对语料库进行模型训练。其次应用以训练好的模型进行答复意见和语料库中留言详情进行比较，其中建立一个全局变量准确控制答复意见和留言详情的对应性并最终保留得分，关键代码如图：

```
#训练模型
from gensim.models.word2vec import LineSentence, Word2Vec
sentences = LineSentence("D:/学习/数学建模/泰迪杯/c/word2vec/listTree.txt")
model = Word2Vec(sentences, min_count=1, iter=1000)
model.save("D:/学习/数学建模/泰迪杯/c/word2vec/w3v.mod")

target = "D:/学习/数学建模/泰迪杯/c/word2vec/listTree.txt"
model ="D:/学习/数学建模/泰迪杯/c/word2vec/w3v.mod"
model_w2v = Word2Vec.load(model)
candidates = []

with open(target, encoding='utf-8')as f:
    for line in f:
        candidates.append(line.strip().split()) #将语料放到列表中便于操作
URL=0
def xsd(text):
    global URL
    words = list(jieba.cut(text.strip())) #分词
    flag = False
    res = []
    index = 0
    candidate=candidates[URL]
    score = model_w2v.n_similarity(words, candidate)
    URL=URL+1
    return score
```

图 14

② 完整性：

关于答复意见结构完整性的判断通过建立完整性表 spacial_chinese 查看答复意见中是否存在必要的称呼如问候答复等。通过 is_contain 方法进行关键词筛选并给出得分，关键代码如图：

```
def is_contain(x):
    a=0
    j=0
    k=0
    m=0
    l=[]
    special_chinese = ["您好","你好","谢谢","感谢","您","谢谢","网友","处理","尊敬","首先","第一"," (一) ","一、","1、","一是","1."]
    for i in special_chinese:
        if i in x:
            a=a+0.1
        if "答复" in x and j==0:
            j=j+1
            a=a+0.2
        if "回答" in x and k==0:
            k=k+1
            a=a+0.2
        if "建议" in x and m==0:
            m=m+1
            a=a+0.2
    return a
```

图 15

③ 可解释性：

可解释性的评判即找出对留言无意义的回答，通过 is_contain 函数找出对留言的答复无意义的答复信息，并标记为 0，其他标记为 1，如图

```
def is_contain(x):
    special_chinese = ["网友您好留言已收悉","已收悉","您好你所反映的问题已转交相关单位调查处置","2016年6月12日","2019年1月14日",
    for i in special_chinese:
        if i == x:
            return 0
    return 1
```

图 16

④ 时间间隔：

对于时间间隔的计算，首先利用 map 中的 lambda 函数分割字符串，保留日期，其次调用 str_time1 函数进行字符串到时间格式的转化，最终算出差值。如图

```
def str_time1(x):
    try:
        x=datetime.datetime.strptime(x, "%Y/%m/%d")
    except ValueError:
        x = datetime.datetime.strptime(x, "%Y-%m-%d")
    return x

df["答复时间"] = df["答复时间"].map(lambda x:x.split()[0]
df["留言时间"]=df["留言时间"].map(lambda x:str(x))
df["留言时间"]=df["留言时间"].map(lambda x:x.split()[0])

captalizer1 = lambda x: str_time1(x)
df["答复时间"]=df["答复时间"].apply(captalizer1)
df["留言时间"]=df["留言时间"].apply(captalizer1)
```

图 17

3.3.2 答复质量评分

给相关性、完整性、可解释性、时间得分分配权重，令相关性权重为 1、完整性权重为 3、可解释性权重为 5、时间得分权重为 3。计算质量得分。

四、实验结果

4.1 问题一

4.1.1 文本切割处理

通过分词处理，将用户留言详情处理后的部分结果如表所示。

表 5

留言详情	一级分类
区 大道 西行 便道 未管 路口 加油站 路段 人行道 路灯 杆 圈 西湖 建筑 集团 燕子 山 安置 房 项目 施工 围墙 上下 …	城乡建设
书院 路 主干道 在水一方 大厦 一楼 四楼 人为 拆除 水电 设施 烂尾 多年 护栏 围着 占用 人行道 护栏 锈迹斑斑 倒塌 …	城乡建设
区苑 小区 区 火炬 路 小区 物业 市程明 物业管理 有限公司 未经 小区业主 同意 利用 业主 公摊 公共 面积 业主 公摊 …	城乡建设
区区 华庭 小区 高层 二次 供水 楼顶 水箱 长年 不洗 水 霉味 水是 日常生活 自来水 龙头 必不可少 用品 霉 一种 …	城乡建设
小区 洗澡 汗液 睡觉 天气 转凉 家里 多岁 老人 洗个 热水澡 停水 水压 小到 打不亮 燃气 一个多月 物业公司 水压 小 …	城乡建设
购买 盛世 耀凯 小区 栋 楼楼 两层 共计 平方 足额 缴纳 物业费 费用 小区 入住 成立 小区 业委会 物业公司 小区 …	城乡建设
西地省 地区 常年 阴冷 潮湿 气候 近年 气候 恶劣 地处 月亮岛 片区 近年 规划 楚江 供暖 不知 规划 供暖 具体 近年 …	城乡建设
胡书记 市区 桐梓 坡 西路 可可 小城 居民 停水 小区业主 业委会 找 物业 开发商 居委会 自来水厂 单位 寻求 找 物业 …	城乡建设
梅家田 社区 辖区 小区 居民 依法 依规 小区 物业公司 交纳 城市 垃圾处理 费 环卫局 收取 城市 垃圾处理 费是 环卫 …	城乡建设
您们好 市区 魏家坡巷 业主 多年 小区 脏乱差 社区 得不到解决 小区 情况 小区 停车 情况 不容乐观 地方 停车 情况 …	城乡建设

4.1.2 一级标签分类模型训练结果

分别运行各个模型的调用代码，各模型训练结果如下表所示。

表 6

算法名称	python 模型	准确率	耗时/s
逻辑回归	LogisticRegression	85.70%	46.3
多项式贝叶斯	MultinomialNB	85.40%	30.4
伯努利贝叶斯	BernoulliNB	56.90%	33.6
lightGBM	lg.LGBMClassifier	86.60%	6.6
随机森林	RandomForestClassifier	82.70%	108.1
高斯贝叶斯	GaussianNB	66.30%	21.6
决策树	DecisionTreeClassifier	72.50%	85.1

由表可知，lightGBM 算法与本次问题契合度最高，该模型的混淆矩阵及召回率如图所示。

>>>混淆矩阵

```
[[135   7   1   7  29   1   4]
 [  0 549  16   5  15   5   0]
 [  2  13 208  18  19   2   1]
 [  4  17   4 270  51  16   3]
 [ 13  16   1  31 527   9   5]
 [  0   9   0   5  15 447   1]
 [  2   2   1   2  16   3 255]]
```

>>>召回率

	precision	recall	f1-score	support
0	0.87	0.73	0.79	184
1	0.90	0.93	0.91	590
2	0.90	0.79	0.84	263
3	0.80	0.74	0.77	365
4	0.78	0.88	0.83	602
5	0.93	0.94	0.93	477
6	0.95	0.91	0.93	281
accuracy			0.87	2762
macro avg	0.87	0.84	0.86	2762
weighted avg	0.87	0.87	0.87	2762

图 18

4.2 问题二

4.2.1 文本相似度匹配结果

通过文本相似度匹配，计算出的相似留言数及留言编号如下表所示。

表 7

留言编号	留言详情	相似留言个数	相似留言序号
233281	尊敬的领导我是一名 9…	99	26, 53, 146, 156, 176…
234868	A 市恒大溪上桃花源楼…	99	19, 69, 78, 145, 176…
225106	我是一名 90 后 F 市人没…	97	32, 53, 69, 107, 114…
212231	尊敬的领导你们好我是…	97	25, 57, 69, 70, 78, 96…
189278	领导您好我是一名肢体…	97	16, 26, 53, 69, 156 …
250517	我孩子今年被 A 市一个…	94	26, 53, 56, 156, 186…
254977	2019624 的时候 A 市搜…	94	19, 26, 39, 53, 69, 13…
201694	以网络方式做零食站点…	92	26, 53, 176, 186, 187…
233281	我们是 A 市 A4 区新河…	89	26, 82, 121, 143, 186…
234868	本人是 A 市原国安驾校…	89	19, 26, 53, 176, 186 …

4.2.2 热点问题得分

根据权重分析算法，计算得分，求得排名前 5 的热点问题如表所示。

表 8

热度排名	问题 ID	热度指数	时间范围	地点/人群	问题描述
1	1	95.3	2019/9/17 4:25:00 2019/11/22 16:54:42	魅力之城市民	小区楼下烧烤油烟打而且到深夜
2	2	70.6	2019/8/26 8:33:03 2019/9/25 0:31:33	伊景园市民	伊景园房子盖好后还不让住户进入
3	3	65.8	2019/12/6 10:22:50 2019/12/20 10:22:50	小区住户	外小区的和随便停入本小区
4	4	51.6	2019/12/1 10:22:50 2019/12/6 10:22:50	伊景园	购车捆绑销售
5	5	48.6	2019/1/6 10:22:50 2019/11/16 10:22:50	西地省学生	本学校强令学生到无关单位实习

参照相似留言编号信息，找出热度前 5 主题的详细信息，部分留言如表所示。

表 9

问题 ID	留言编号	留言用户	留言主题	留言时间	留言详情	点赞数	反对数
1	360103	A0012425	A5 区劳…	2019/9/25 0:31:33	A 区劳动…	0	0
2	360108	A0283523	A1 区劳…	2019/8/1 16:20:02	局长你好…	6	0
3	360101	A324156	B6 区劳…	2019/7/28 12:49:18	沪昆高速…	4	0

4.3 问题三

将原数据带入程序，计算出相关性、完整性、可解释性、时间差（天）四个指标的值如下表所示。

表 10

留言详情	答复意见	答复时间	相关性	完整性	可解释性	时间差
2019 年 4…	现将网友在…	2019/5/10 14:56	1	1	15	1
潇楚南路…	网友 A0002...	2019/5/9 9:49	0.5	1	15	0.5
地处省会…	市民同志你…	2019/5/9 9:49	0.5	1	15	0.5
尊敬的书…	网友 A0001…	2019/5/9 9:49	0.5	1	15	0.5
建议将白…	网友 A0009…	2019/5/9 9:51	0.8	1	16	0.8
欢迎领导…	网友 A0007…	2019/5/9 10:02	0.4	1	31	0.4
尊敬的胡…	网友 A0001…	2019/5/9 10:18	0.5	1	41	0.5
我做为一…	网友 UU008…	2019/1/29 10:53	0.6	1	29	0.6
我是美麓…	网友 UU008…	2019/1/16 15:29	0.7	1	16	0.7
胡书记好…	网友 UU008…	2019/5/10 14:56	1	1	15	1

将求得的值带入评价模型中，求得的得分如下表所示。

表 11

留言编号	相关性	完整性	可解释性	时间差	时间得分	得分
32836	0.753531	1.2	1	1	4	21.35353
137172	0.698329	0.9	1	3	0	20.39833
90356	0.581453	0.8	1	3	4	19.98145
32979	0.18421	0.9	1	2	4	19.88421
32751	0.168419	0.9	1	1	0	19.86842

五、结论及展望

5.1 实验结果

成功编写文本分词算法及一级标签分类算法模型，通过实验，发现 lightGBM 算法与本问题匹配度较高，运行准确度为 86.3%，耗时 6.6 秒，远高于其他文本分类算法，训练的模型保存为“lightGBM.pkl”文件。

问题二、三通过文本相似度等指标，进行权重分析，成功构建评价指标。

附录、参考文献

[1]权军,易萍.浅议人工智能在人社系统电子政务公共服务领域应用的意义[J].劳动保障世界,2019(34):56-57.

[2]王冬梅.大数据背景下人工智能如何服务于政府经济决策[J].中国统计,2017(04):8-10.

[3]郭腾州,孙宝山.深度学习在文本生成中的应用研究[J].仪器仪表用户,2020,27(02):110-112+42.

[4]韩万渠.政民互动平台推动公众有效参与的运行机制研究——基于平台赋权和议题匹配的比较案例分析[J].探索,2020(02):149-160.

附录一 文本分类算法运行结果

lightGBM：

```
训练:
LGBMClassifier(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
               importance_type='split', learning_rate=0.1, max_depth=-1,
               min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
               n_estimators=100, n_jobs=-1, num_leaves=31, objective=None,
               random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
               subsample=1.0, subsample_for_bin=200000, subsample_freq=0)
预测:
[4 4 4 ... 2 2 2]
评估:
>>>准确率
 0.8656770456191166

>>>混淆矩阵
 [[135   7   1   7  29   1   4]
 [  0 549  16   5  15   5   0]
 [  2  13 208  18  19   2   1]
 [  4  17   4 270  51  16   3]
 [ 13  16   1  31 527   9   5]
 [  0   9   0   5  15 447   1]
 [  2   2   1   2  16   3 255]]

>>>召回率
              precision    recall  f1-score   support

           0       0.87      0.73      0.79       184
           1       0.90      0.93      0.91       590
           2       0.90      0.79      0.84       263
           3       0.80      0.74      0.77       365
           4       0.78      0.88      0.83       602
           5       0.93      0.94      0.93       477
           6       0.95      0.91      0.93       281

    accuracy                           0.87      2762
   macro avg       0.87      0.84      0.86      2762
weighted avg       0.87      0.87      0.87      2762


>>>算法消耗时间为: 6.640444755554199 秒


Process finished with exit code 0
```

伯努利贝叶斯：

```
训练:
BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
预测:
[4 4 4 ... 4 1 1]
评估:
>>>准确率
 0.5687907313540912

>>>混淆矩阵
 [[  1  22   0   7 153   0   1]
 [  0 546   3   0  40   1   0]
 [  0 119  32   4 104   4   0]
 [  2  19   2  52 283   7   0]
 [  1  25   0   5 565   5   1]
 [  4  56   0   3 108 306   0]
 [  1   2   0   0 209   0  69]]

>>>召回率
              precision    recall  f1-score   support

           0       0.11      0.01      0.01       184
           1       0.69      0.93      0.79       590
           2       0.86      0.12      0.21       263
           3       0.73      0.14      0.24       365
           4       0.39      0.94      0.55       602
           5       0.95      0.64      0.76       477
           6       0.97      0.25      0.39       281

    accuracy                           0.57      2762
   macro avg       0.67      0.43      0.42      2762
weighted avg       0.68      0.57      0.51      2762


>>>算法消耗时间为： 33.61093091964722 秒


Process finished with exit code 0
```

多项式贝叶斯

```
训练:
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
预测:
[4 4 1 ... 2 2 2]
评估:
>>>准确率
 0.8540912382331644

>>>混淆矩阵
 [[ 92  16   1  19  47   2   7]
 [  0 550  13   2   9  15   1]
 [  0  20 221  12   3   4   3]
 [  2   6   5 260  70  15   7]
 [  3  32   0  16 521  11  19]
 [  0  17   0   9   8 443   0]
 [  0   1   0   0   7   1 272]]

>>>召回率
              precision    recall  f1-score   support

           0       0.95      0.50      0.65       184
           1       0.86      0.93      0.89       590
           2       0.92      0.84      0.88       263
           3       0.82      0.71      0.76       365
           4       0.78      0.87      0.82       602
           5       0.90      0.93      0.92       477
           6       0.88      0.97      0.92       281

    accuracy                           0.85      2762
   macro avg       0.87      0.82      0.84      2762
weighted avg       0.86      0.85      0.85      2762


>>>算法消耗时间为:  30.390491008758545 秒
```

高斯贝叶斯：

```
训练:
GaussianNB(priors=None, var_smoothing=1e-09)
预测:
[4 4 4 ... 2 2 5]
评估:
>>>准确率
 0.663287472845764

>>>混淆矩阵
 [[ 43  13   3  39  65  16   5]
 [  1 452  35  14  18  69   1]
 [  0  77 140  17   6  20   3]
 [  5  19   5 209  77  35  15]
 [  9  51   5  48 418  39  32]
 [  0  52   0  11  43 369   2]
 [  2   5   0  11  55   7 201]]

>>>召回率
              precision    recall  f1-score   support

           0       0.72      0.23      0.35       184
           1       0.68      0.77      0.72       590
           2       0.74      0.53      0.62       263
           3       0.60      0.57      0.59       365
           4       0.61      0.69      0.65       602
           5       0.66      0.77      0.72       477
           6       0.78      0.72      0.74       281

    accuracy                           0.66      2762
   macro avg       0.68      0.61      0.63      2762
weighted avg       0.67      0.66      0.65      2762


>>>算法消耗时间为:  21.606224060058594 秒
```

决策树：

```
训练:
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
预测:
[4 4 4 ... 1 2 2]
评估:
>>>准确率
 0.725199131064446

>>>混淆矩阵
 [[105  11   6  15  35   7   5]
 [  4 488  35   8  39  15   1]
 [  4  38 168  18  25   8   2]
 [ 15  46  11 184  82  20   7]
 [ 14  55   7  59 417  22  28]
 [  2  14   1  12  17 429   2]
 [  1   5   3   3  47  10 212]]

>>>召回率
              precision    recall  f1-score   support

           0       0.72      0.57      0.64       184
           1       0.74      0.83      0.78       590
           2       0.73      0.64      0.68       263
           3       0.62      0.50      0.55       365
           4       0.63      0.69      0.66       602
           5       0.84      0.90      0.87       477
           6       0.82      0.75      0.79       281

    accuracy                           0.73      2762
   macro avg       0.73      0.70      0.71      2762
weighted avg       0.72      0.73      0.72      2762


>>>算法消耗时间为:  85.09719967842102 秒


Process finished with exit code 0
```

逻辑回归：

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
预测:
[4 4 4 ... 2 2 2]
评估:
>>>准确率
 0.8573497465604635

>>>混淆矩阵
 [[135   9   1   3  31   1   4]
 [  0 550  13   5  17   5   0]
 [  0  23 209  15  14   1   1]
 [  4  26   6 264  54   7   4]
 [ 11  29   0  28 515   9  10]
 [  0  16   0   7  17 436   1]
 [  1   3   1   2  13   2 259]]

>>>召回率
              precision    recall  f1-score   support

           0       0.89      0.73      0.81       184
           1       0.84      0.93      0.88       590
           2       0.91      0.79      0.85       263
           3       0.81      0.72      0.77       365
           4       0.78      0.86      0.82       602
           5       0.95      0.91      0.93       477
           6       0.93      0.92      0.92       281

    accuracy                           0.86      2762
   macro avg       0.87      0.84      0.85      2762
weighted avg       0.86      0.86      0.86      2762


>>>算法消耗时间为:  46.25247764587402 秒


Process finished with exit code 0
```

随机森林：

```
训练:
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
预测:
[4 4 4 ... 2 2 2]
评估:
>>>准确率
 0.8272990586531499

>>>混淆矩阵
 [[100  16   1   5  54   4   4]
 [  0 567  10   1   6   6   0]
 [  1  38 198  10  12   2   2]
 [  4  44   8 203  87  12   7]
 [  6  40   0  14 527  10   5]
 [  0  16   0   4  13 444   0]
 [  0   2   0   2  24   7 246]]

>>>召回率
              precision    recall  f1-score   support

           0       0.90      0.54      0.68       184
           1       0.78      0.96      0.86       590
           2       0.91      0.75      0.82       263
           3       0.85      0.56      0.67       365
           4       0.73      0.88      0.80       602
           5       0.92      0.93      0.92       477
           6       0.93      0.88      0.90       281

    accuracy                           0.83      2762
   macro avg       0.86      0.79      0.81      2762
weighted avg       0.84      0.83      0.82      2762


>>>算法消耗时间为： 108.12015080451965 秒


Process finished with exit code 0
```