智慧政务”中的文本挖掘应用

摘要

近年来，随着科技的发展与进步，政府了解民意、 汇聚民智、凝聚民气的方式产生了很大的变化。政府不仅利用物联网、云计算、移动互联网、人工智能、数据挖掘、知识管理等技术，还强调以用户创新、大众创新、开放创新、共同创新为特征创新 2.0 方法论，提高政府办公、监管、服务、决策的智能化水平。同时，在国家“新基建”的浪潮下，政务服务也将进一步向数字化和智能化的方向发展，政府也会形成高效、敏捷、便民的新型政府。

针对问题一，首先将附件二 9211 条信息中的留言主题中相同的内容进行删除，并将处理之后的文本进行分词以及去除停用词处理，最后将清洗过的内容储存在文件中，完成数据的预处理。其次，借用 python 的 sklearn 库和 pickle 库对清洗过的内容进行分词存储，并将分词好的文本文档转化为持久性的 Bunch 类形式。然后，构建向量空间并初始化向量空间，导入贝叶斯算法包并将数据导入向量空间，应用朴素贝叶斯的方法并成功地根据附件一的一级标签将其分类，建立出一级标签分类模型。

针对问题二，首先对附件三中 4327 条信息中的留言主题的内容进行预处理过程。其次，借用 python 中的 xlwt 库和 openpyxl 库进行 excel 处理，增加热度排名，问题 ID，热度指数的三列表格；接着进行相似性处理，按照时间范围、地点/人群分类，将热度排序并写入目标 excel，最终完成了发现热点问题并对其排名、统计热点问题相对应的留言信息并将其归类，最后将二者存入 excel 表格中。

针对问题三，首先我们分别对附件四中 2817 条信息的留言主题和答复意见进行预处理过程。其次，借用 python 中的 gensim 库和 collections 库将二者通过 doc2bow 转化为稀疏向量，对稀疏向量进行进一步处理，得到新语料库对新语料库通过‘tfidfmodel’进行处理，得到 tfidf 通过 token2id 得到特征数稀疏矩阵相似度，从而建立索引，最终，输出相似度结果并在此基础上对其分析。

关键词：数据预处理；留言分类；向量空间；朴素贝叶斯算法；文本相似度计算；特征提取

Abstact

In recent years, with the development and progress of science and technology, great changes have taken place in the way the government understands public opinion, gathers people's wisdom and gathers people's spirit. The government not only makes use of the Internet of things, cloud computing, mobile Internet, artificial intelligence, data mining, knowledge management and other technologies, but also emphasizes the innovation 2.0 methodology featuring user innovation, mass innovation, open innovation and joint innovation to improve the intelligence level of government office, supervision, service and decision-making. At the same time, under the tide of national "new infrastructure", the government service will be further developed to the digital and intelligent direction, and the government will also form a new type of efficient, agile and convenient government.

For problem 1, first delete the same content in the message subject in the 9211 messages in annex ii, and then divide the processed text into words and remove the stop words. Finally, store the cleaned content in the file to complete the data preprocessing. Secondly, python's sklearn library and pickle library are used to store the cleaned contents in word segmentation, and the text documents with good word segmentation are converted into the persistent form of Bunch class. Then, the vector space is constructed and initialized, the bayesian algorithm package is imported and the data is imported into the vector space, and the naive bayes method is applied to classify it successfully according to the first-level label of attachment 1, and the first-level label classification model is established.

For question 2, the content of the message subject in the 4327 messages in annex 3 is preprocessed. Secondly, the XLWT library and openpyxl library in python are used for excel processing to increase the three-column table of heat ranking, problem ID and heat index. Then, similarity processing was carried out. According to time range, place/population classification, heat was sorted and written into target excel. Finally, hot issues

were found, ranked, and the corresponding messages of hot issues were classified.

For question 3, we first preprocessed the message subject and reply comments of 2817 pieces of information in annex 4. Second, use the python gensim libraries and the collections of the library will both by doc2bow into sparse vector, the sparse vector for further processing, get new corpus to deal with new corpus by 'tfidfmodel, get tfidf characteristic number of sparse matrix is obtained by token2id similarity, therefore index is set up, in the end, the results and output similarity based on the analysis to it.、

Key words: Data preprocessing; Message classification; Vector space; Naive bayes algorithm; Text similarity calculation; Feature extraction

目录

1.挖掘背景

信息化已经进入新的发展阶段，大数据，人工智能等新兴技术和事物不断涌现,智慧地球、智慧城市等概念不断占据着公共管理场域，通过借助云计算、数据分析和挖掘等也给政府的变革带来契机，智慧政务作为电子政务发展的高阶阶段,积极运用新工具、新技术，最终将改变公众的生活状态。由于各类社情民意相关的文本数据量不断攀升，存在工作量大、效率低，且差错率高等问题，这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。随着大数据、云计算、人工智能等技术的发展，对这些数据进行相应的统计分析挖掘从而进一步得到有价值的信息，已成为我们着重研究方向。为能够提高效率且及时准确的处理市民的相应问题，研究并制定一套基于群众留言反映相关热点问题以及相关部门对留言的答复意见质量的评优的方法，旨在通过对互联网公开来源搜索的系统数据进行研究，开展相关的数据挖掘工作并对结果进行深入分析，建立评价方法，对评论进行统计分析，以提升政府的管理水平和施政效率。

2.挖掘目标

建模的目标是通过相关部门所提供的的各类社情民意的留言内容进行数据挖掘，参考三级分类的标签、群众的留言内容所提及的问题、时间、地点以及所涉及的人群，通过对自然语言处理和文本挖掘的方法，达到以下三个目标：

（1） 由于目前，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。在此基础上，根据考附件 1 提供的内容分类三级标签体系，对留言进行分类，并根据附件 2 给出的数据，建立关于留言内容的一级标签分类模型。
（2） 根据附件 3 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，挖掘热点问题、给出评价结果，并将结果和信息保存在文件中。
（3） 针对附件 4 相关部门对留言的答复意见，从答复的相关性、完整性、可解释性等角度对答复意见的质量给出一套评价方案，并尝试实现。

3.基本假设

（1）假设政府所收集的用户信息输入的准确性，不存在信息中含有错别字、输入窜行等错误。
（2）假设民众所提供的留言内容不存在抄袭、跟风等现象。
（3）假设经过数据预处理后所呈现的数据能够准确的反映市民所遇到的问题。

4.数据预处理

由于政府所统计的民众评论的有关信息的复杂化以及大量化问题，题目附件中所给的信息可能存在者某些异常，例如：大量的冗余信息、不一致信息、非结构化数据和异常的数据。这将会严重的影响数据挖掘和建模的执行效率，甚至可能会导致数据挖掘的结果出现偏差，因此，先对题目中给定的原始文本数据进行预处理操作，以提高数据集的质量，使得数据能够更好地适应建立的挖掘算法和模型。

4.1 数据清洗

对留言主题进行数据清洗：

数据清洗是对数据进行重新审查和校验的过程，目的在于删除重复信息、纠正存在的错误，并提供数据一致性。数据一致性包括检查数据一致性，处理无效值和缺失值等。因为附件 1 中的信息是 9211 条留言的集合，这些数据从政府相关部门收集而来而且包含重复以及无用信息，避免不了信息的错误与冗杂。

主要进行对留言内容的分析和处理：

首先针对附件一提供的留言主题中相同的内容利用 python 中的 jieba 库进行删除，并将处理之后的文本进行分词以及用 stopwords 文本文档进行去除停用词处理，最后将清洗过的内容储存在文件中，完成数据的清洗。

主要代码如下：

```
data=pd.read_excel('D:/大二下/数据挖掘比赛/全部数据/C 题全部数据/附件 2.xlsx',encoding='gbk')
data_new=data['留言主题']
data_dup=data_new.drop_duplicates()       #对相同的短信内容进行删除
data_cut=str(data_dup.apply(lambda  x:  jieba.lcut(x)))          #分词
```

删减完相同内容后的信息如图所示：

```
0                    A市西湖建筑集团占道施工有安全隐患
1                 A市在水一方大厦人为烂尾多年，安全隐患严重
2                    投诉A市A1区苑物业违规收停车费
3                 A1区蔡锷南路A2区华庭楼顶水箱长年不洗
4                    A1区A2区华庭自来水好大一股霉味
                          ...
9204                   要解决医患矛盾必须装监控！
9205                 两孩子一个是一级脑瘫，能再生育吗？
9206    B市中心医院医生不负责任，做无痛人流手术后结果还是活胚芽
9208                   K8县惊现奇葩证明！
9209              请问J4县卫计委社会抚养费到底该交多少钱？
Name: 留言主题, Length: 8905, dtype: object
```

分词后并去掉 stopwords 的信息如图所示：

```
0                    [A, 市, 西湖, 建筑, 集团, 占, 道, 施工, 有, 安全隐患]
1                 [A, 市, 在水一方, 大厦, 人为, 烂尾, 多年, ，, ，, 安全隐患, 严
2                    [投诉, A, 市, A1, 区苑, 物业, 违规, 收, 停车费]
3                 [A1, 区, 蔡锷, 南路, A2, 区华庭, 楼顶, 水箱, 长年, 不洗]
4                    [A1, 区, A2, 区华庭, 自来水, 好大, 一股, 霉味]
                                ...
9204                   [要, 解决, 医患, 矛盾, 必须, 装, 监控, ！]
9205              [两, 孩子, 一个, 是, 一级, 脑瘫, ，, 能, 再, 生育, 吗, ？]
9206    [B, 市中心, 医院, 医生, 不负责任, ，, 做, 无痛, 人流, 手术, 后, 结
9208                   [K8, 县惊现, 奇葩, 证明, ！]
9209              [请问, J4, 县卫, 计委, 社会, 抚养费, 到底, 该交, 多少, 钱, ?
Name: 留言主题, Length: 8905, dtype: object
```

4.2 数据规约

在对政府收集的用户信息的大数据集上进行复杂的数据分析和挖掘将耗费大量的计算时间，数据规约能够产生更小的但保持原始数据完整性的新数据集，提高对用户信息描述的准确性和完整性，还可以大幅度缩减数据挖掘所需的时间，有效降低存储数据的成本。

在对留言主题的数据进行统计分析时发现每条信息都很长，且都没有按照附件一那样做出很正确的一级分类，二级分类和三级分类，为了后期更方便的进行数据挖掘工作，对每个用户的留言主题以及相关的数据进行数据规约处理。

在规约之后的数据集上继续进行分析和挖掘。

5.分析方法与过程

5.1 问题 1 分析方法与过程

按照一定的划分体系，参考附件一的三级分类标签将留言内容进行分类，建立出一级分类标签模型。

5.1.1 问题 1 模型建立与求解

借用 python 的 sklearn 库和 pickle 库对清洗过的内容进行分词存储，并将分词好的文本文档转化为持久性的 Bunch 类形式。然后，构建向量空间并初始化向量空间，导入贝叶斯算法包并将数据导入向量空间，应用朴素贝叶斯的方法[1]并成功地根据附件一的一级标签将其分类，建立出一级标签分类模型。其部分代码如下（详细代码见附件）：

1.进行训练文本分词存储

```
def segment(corpus_path,seg_path):
 # 获取 corpus_path 下的所有子目录
 cateList = os.listdir(corpus_path)
 for myDir in cateList:
     if not myDir.startswith("."):
         class_path = corpus_path+myDir+"/"
         seg_dir = seg_path+myDir+"/"
         if not os.path.exists(seg_dir):
             os.makedirs(seg_dir)
         file_list = os.listdir(class_path)
         # 遍历类别目录下的所有文件
         for file_path in file_list:
             fullname = class_path + file_path
             print("path:" + fullname)
             content = readfile(fullname, "GBK")
             if content != None:
                 content = content.strip()
                 # 删除换行和多余的空格
                 content = content.replace("\r\n", "").strip()
                 # 为文件的内容分词
                 content_seg = jieba.cut(content)
                 savefile(seg_dir + file_path,
"".join(content_seg))
 print("中文语料分析结束！")
```

2.将分好词的文本文件转换并持久化为 Bunch 类形式

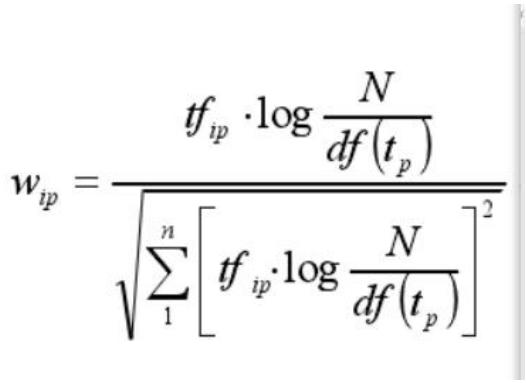
```
def readbunchobj(path):
    file_obj = open(path, "rb")
    bunch = pickle.load(file_obj)
    file_obj.close()
    return bunch
def writebunchobj(path,bunchobj):
    file_obj = open(path, "wb")
    pickle.dump(bunchobj, file_obj)
    file_obj.close()
def bunchObj(wordbag_path,seg_path):
    bunch = Bunch(target_name=[], label=[], filename=[], contents=[])
    catelist = os.listdir(seg_path)
    bunch.target_name.extend(catelist)
    for myDir in catelist:
        if not myDir.startswith("."):
            class_path = seg_path + myDir + "/"
            file_list = os.listdir(class_path)
            for file_path in file_list:
                fullname = class_path + file_path
                print(fullname)
                bunch.label.append(myDir)
                bunch.filename.append(fullname)

bunch.contents.append(readfile(fullname, "GBK").strip())
    file_obj = open(wordbag_path, "wb")
    pickle.dump(bunch, file_obj)
    file_obj.close()
    print("构建文本对象结束！")
```

3. 训练模型

词频（TF）：是一种把特征项[2]在文档中出现的频率作为权重的计算方法。

词频-反文档频率（TF-IDF）[3]：是目前被大家广泛应用的权重计算方法。其中 TF 是指特征项对文本内容的一种描述力，IDF 是指特征项对文本的一种区分力。它的基本思想是：如果某一特征在一文本中出现次数很多，在其它文本中出现次数很少，那么预示着该特征对这一文本的代表性越强并且能够很好的对类别进行区分。计算公式如下：



```
def startTrain(stopword_path, wordbag_path, space_path):
    stpwrdlst = readfile(stopword_path,"utf-8").splitlines()
    bunch = readbunchobj('C:/Users/ASUS/Desktop/分词语料 bunch 对象持久
化文件.dat')

    tfidfspace = Bunch(target_name=bunch.target_name,
label=bunch.label, filename=bunch.filename, tdm=[], vocabulary={})
    vectorizer = TfidfVectorizer(stop_words=stpwrdlst,
sublinear_tf=True, max_df=0.5)
    transform = TfidfTransformer()
    tfidfspace.tdm = vectorizer.fit_transform(bunch.contents)
    tfidfspace.vocabulary = vectorizer.vocabulary_

    writebunchobj(space_path, tfidfspace)
    print("文本分类模型训练完成")

# 未分词分类语料库路径
corpus_path = "C:/Users/ASUS/Desktop/未分词分类语料库/"
# 分词后的分类语料库路径
segment_path = "C:/Users/ASUS/Desktop/分词后的语料库/"
# 分词语料 Bunch 对象持久化文件路径
wordbag_path = "C:/Users/ASUS/Desktop/分词语料 bunch 对象持久化文
件.dat"
# 停用词路径
stop_words_path = "C:/Users/ASUS/Desktop/呆萌的停用词表.txt"
# 创建词袋的持久化路径
space_path = "C:/Users/ASUS/Desktop/创建词袋的持久化路径.dat"


startTrain('C:/Users/ASUS/Desktop/呆萌的停用词表.txt',
'C:/Users/ASUS/Desktop/分词语料 bunch 对象持久化文件.dat',
'C:/Users/ASUS/Desktop/创建词袋的持久化路径.dat')

def readbunchobj(path):
```

```
    file_obj = open(path,"rb")
    bunch    = pickle.load(file_obj)
    file_obj.close()
    return bunch

trainpath = "/Users/FengZhen/Desktop/accumulate/机器学习
/classify_test/train_word_bag/tfidfspace.dat"
train_set = readbunchobj(trainpath)

testpath  = "/Users/FengZhen/Desktop/accumulate/机器学习
/classify_test/test_word_bag/testspace.dat"
test_set = readbunchobj(testpath)
```

4. 应用朴素贝叶斯

贝叶斯分类是一类分类算法的总称，这类算法均已贝叶斯定理为基础，因此统称为贝叶斯分类。事件 A 在事件 B（发生）的条件下的概率，与事件 B 在事件 A（发生）的条件下的概率是不一样的，但他们有确定的关系，贝叶斯定理就是对在这种关系的陈述。公式如下：

$$P(x|y_i)P(y_i) = P(a_1|y_i)P(a_2|y_i)\cdots P(a_m|y_i)P(y_i)。$$

```
#alpha:0.001 alpha 越小，迭代次数越多，精度越高
clf = MultinomialNB(alpha = 0.001).fit(train_set.tdm,train_set.label)

#预测分类结果
predicted = clf.predict(test_set.tdm)
total     = len(predicted)
rate      = 0
for flabel,file_name,expct_cate in
zip(test_set.label,test_set.filenames,predicted):
    print(file_name, u":实际类别:", flabel, u"-->预测类别:",
expct_cate)
    if flabel != expct_cate:
        rate  += 1
        print(file_name,u":实际类别:",flabel,u"-->预测类
别:",expct_cate)
#精度
print("error rate:",float(rate)*100/float(total),"%")
```

5.2 问题 2 分析方法与过程

附件三中所给的所示数据，首先找出热点问题，并依据时间范围，地点/人群等分类，将问题描述分成不同的类别并将其定义合理的热度评价指标，做出热度排名并给出评价结果。

5.2.1 问题 2 模型建立及求解

借用 python 中的 xlwt 库和 openpyxl 库进行 excel 处理，增加了热度排名，问题 ID，热度指数的表格；接着进行相似性处理[4]、按照热度高低排序以及写入目标 excel，最终完成了发现热点问题并对其排名、统计热点问题相对应的留言信息并将其归类[5]、最后将二者存入 excel 表格中。

主要代码如下：

1.excel 处理

通过 excel 处理，增加了热度排名，问题 ID，热度指数的表格

```
wbk = xlwt.Workbook(encoding='ascii')
sheet = wbk.add_sheet("热度排名")  # sheet 名称
sheet.write(0, 0, '热度排名')
sheet.write(0, 1, 'Id')
sheet.write(0, 2, '热度指数')
sheet.write(0, 3, '时间范围')
sheet.write(0, 4, '地点人群')
sheet.write(0, 5, '问题描述')
```

```
wb = openpyxl.load_workbook('D:/大二下/数据挖掘比赛/全部数据/C 题全部数据/附件 3.xlsx')
ws = wb.active
col = ws['B']
```

2.相似性处理

首先将所有评论的关键词集合在一起， 计算每一条评论的关键词向量与最优关键词向量的余弦相似度。

余弦相似度：在相似度计算中，一般会将文本表示为由词组成的向量，具体的取值为上述的 TF-IDF 权重，因此两条文本的相似性就用两个向量的相似度[6]来表示，通常采用余弦相似度来计算，夹角越小越相似。应用公式如下：

$$\cos(\theta) = \frac{\sum_{i=1}^{n} X_i \times Y_i}{\sqrt{\sum_{i=1}^{n}(X_i)^2} \times \sqrt{\sum_{i=1}^{n}(Y_i)^2}}$$

```
  rcount = 1
    texts = []
    orig_txt = []
    key_list = []
    name_list = []
    sheet_list = []

    for cell in col:
        if cell.value is None:
            continue
        if not isinstance(cell.value, str):
            continue
        item = cell.value.strip('\n\r').split('\t')
        string = item[0]
        if string is None or len(string) == 0:
            continue
        else:
            textstr = seg_sentence(string, stop_words)
            texts.append(textstr)
            orig_txt.append(string)
    dictionary = corpora.Dictionary(texts)
    feature_cnt = len(dictionary.token2id.keys())
    corpus = [dictionary.doc2bow(text) for text in texts]
    tfidf = models.LsiModel(corpus)
    index = similarities.SparseMatrixSimilarity(tfidf[corpus],
num_features=feature_cnt)
    result_lt = []
    word_dict = {}
    count =0
    for keyword in orig_txt:
        count = count+1
        if keyword in result_lt or keyword is None or len(keyword) == 0:
            continue
        kw_vector = dictionary.doc2bow(seg_sentence(keyword,
stop_words))
        sim = index[tfidf[kw_vector]]
        result_list = []
        for i in range(len(sim)):
            if sim[i] > 0.5:
                if orig_txt[i] in result_lt and orig_txt[i] not in
result_list:
                    continue
                result_list.append(orig_txt[i])
                result_lt.append(orig_txt[i])
```

```
        if len(result_list) >0:
            word_dict[keyword] = len(result_list)
        if len(result_list) >= 1:
            sname =
re.sub(u"([^\u4e00-\u9fa5\u0030-\u0039\u0041-\u005a\u0061-\u007a])",
"", keyword[0:10])+ '_'\
                    + str(len(result_list)+ len(str_to_hex(keyword)))
+ str_to_hex(keyword)[-5:]
            sheet_t = wbk.add_sheet(sname)  # Excel 单元格名字
            for i in range(len(result_list)):
                sheet_t.write(i, 0, label=result_list[i])
```

3.按照热度排序

按照相似度从高到低进行排序并基于词向量相似度的排序模型构建，计算每一条评论的关键词向量与最优关键词向量[7]的距离，根据距离进行评论排序。

```
     with open('C:/Users/ASUS/Desktop/附件三.txt', 'w',
encoding='utf-8') as wf2:
            orderList = list(word_dict.values())
            orderList.sort(reverse=True)
            count = len(orderList)
            for i in range(count):
                for key in word_dict:
                    if word_dict[key] == orderList[i]:
                        key_list.append(key)
                        word_dict[key] = 0
            wf2.truncate()
```

4.写入目标 excel

将热点问题排名以及有关时间范围、地点/人群等分类的信息写入新的 excel 表格中，完成。

```
    for i in range(len(key_list)):
        sheet.write(i+rcount, 0, label=key_list[i])
        sheet.write(i+rcount, 1, label=orderList[i])
        if orderList[i] >= 1:
            shname =
re.sub(u"([^\u4e00-\u9fa5\u0030-\u0039\u0041-\u005a\u0061-\u007a])",
"", key_list[i][0:10]) \
                     + '_'+ str(orderList[i]+
len(str_to_hex(key_list[i])))+ str_to_hex(key_list[i])[-5:]
            link = 'HYPERLINK("#%s!A1";"%s")' % (shname, shname)
            sheet.write(i+rcount, 2, xlwt.Formula(link))
    rcount = rcount + len(key_list)
    key_list = []
    orderList = []
```

```
texts = []
orig_txt = []
wbk.save('C:/Users/ASUS/Desktop/热点表.xls')
```

热度排名	问题ID	热度指	围	地点/人群	问题描述
1	1	92	/2至2019/	A市	市周围环境的改造
2	2	17	1/10至202	A市	通以及车辆的问题
3	3	17	/1/6至2020	A3区	关街道的建设
4	4	17	/5/7至2020	A市	天桥和地铁站的建设
5	5	12	4/22至2019	青山新村社	关社区的改造

问题ID	留言编号	留言用户	留言主题	留言时间	留言详情	反对数	点赞数
1	244993	A00031618	城市建设全	2019/11/1	套落后随处	0	1
1	188409	A0003274	道站地铁出	2019/6/19	星沙四区	0	4
1	188820	A00028138	区棚户改造	2019/2/21	7县星沙街	0	0
1	189278	A00048293	疾人有什么	2019/7/29	我现在房子	0	0
1	190213	A00031618	来水深度净	2019/1/16	业污染导致	0	0
2	191872	A00031618	快轨道交通	2019/3/1	网络和数十	2	9
2	193678	A00010407	市交通存在	2019/3/24	为何轨道	0	8
2	194663	A00084955	口黄金大道口	2019/11/5	，特别是上	0	0
2	197527	A00010625	段与交通安全	2019/3/29	滞不前，车	2	9
2	197747	A00010215	交通管理所	2019/9/4	，感到不可	0	1
3	189739	A00051608	道茶场村五	2019/9/12	周边都拆迁	0	0
3	231507	A00096989	锦绣小区有十	2019/5/27	正常休息。本	0	0
3	289772	A00041363	雅盛园安置小	2019/3/19	些费用支出	0	5
3	192027	A00020229	安置小区惟乡	9/2/20 18:04	等质量问题	0	0
3	201930	A0006149	落雨棚超外地	2019/9/20	窗户的外墙	0	0
4	192521	A00083527	加智能红绿	2019/5/7	方向都能够	0	0
4	191993	A00083527	增加智能红绿	2019/5/7	间，增加路	0	1
4	197437	A00013625	电学院段有	2019/4/16	修地道是	0	0
4	192897	A00011163	站晚上工地	2019/3/13	窗户会更加	0	0
4	203070	A00017571	地铁站公交	2019/8/19	得从A6区坐	0	2
5	188059	A00028571	期与四期中	2019/11/22	开业主，态	0	0
5	188073	A909164	麓谷明珠	2019/3/11	府调规、改	0	0
5	190033	A00010696	厦旁新规划	2019/11/3	启用，泉星	0	5
5	191327	A00073692	区空地夜间	2019/11/18	工作业，但	0	0
5	197442	A00080321	星社区拆除	2019/9/12	。 你们敢	0	15

5.3 问题 3 分析方法与过程

针对附件 4 相关部门对留言的答复意见，我们要利用代码分析答复意见与留言主题的二者之间的关联性，从答复的相关性、完整性、可解释性等角度对答复意见的质量给出一套评价方案，并尝试实现。

5.3.1 问题 2 模型建立及求解

借用 python 中的 gensim 库和 collections 库将二者通过 doc2bow 转化为稀疏向量对稀疏向量进行进一步处理,得到新语料库对新语料库通过‘tfidfmodel’进行处理，得到 tfidf 通过 token2id 得到特征数稀疏矩阵相似度，从而建立索引，最终，输出相似度结果并在此基础上对其分析。部分代码如下：

1.首先对留言主题和答复意见进行预处理过程,并将留言主题的每个词语后面都加上空格，有利于我们做进一步的处理。

```
data=pd.read_excel('D:/大二下/数据挖掘比赛/全部数据/C 题全部数据/附件 4.xlsx',encoding='gbk')
d1=data['留言主题']
d2=data['答复意见']
data1=d1.apply(lambda x: jieba.lcut(x))
data2=d2.apply(lambda x: jieba.lcut(x))
data11=data1.apply(lambda x: ' '.join(x))
data22=data2
stopWords=pd.read_csv('D:\大二下\泰迪杯工作室\基于文本内容的垃圾短信分类\基于文本内容的垃圾短信识别-数据&代码\stopword.txt',

encoding='GB18030',engine='python',sep='hahaha',header=None)
data_afrter_stop1=data11.apply(lambda x: [i for i in  x if i not in stopWords])
data_afrter_stop2=data22.apply(lambda x: [i for i in  x if i not in stopWords])
texts= list(data11.str.split(' '))
new_doc=str(data22)
```

2.计算词语的频率，对词语整理

```
  frequency=defaultdict(int)
for text in texts:
    for token in text:
        frequency[token]+=1
```

3.通过语料库建立词典

```
  dictionary=corpora.Dictionary(texts)
dictionary.save('C:/Users/ASUS/Desktop/数据/分词后的语料库/分词附件四.txt')
```

4.将要对比的文档通过 doc2bow 转化为稀疏向量

```
  new_vec=dictionary.doc2bow(new_doc.split())
```

5.对稀疏向量进一步处理，得到新语料库

```
  corpus=[dictionary.doc2bow(text) for text in texts]
```

6.通过 tf-idf 模型处理新语料库，得到 tfidf 值

```
  tfidf=models.TfidfModel(corpus)
```

7.通过 token2id 得到特征数（字典里面的键的个数）

```
  feature_Num=len(dictionary.token2id.keys())
```

8.计算稀疏矩阵相似度，建立一个索引，根据索引得到最终相似度

```
index=similarities.SparseMatrixSimilarity(tfidf[corpus],num_features=
feature_Num)
sim=index[tfidf[new_vec]]
print(sim)
```

```
1  留言主题和答复意见相似度为：73.77%
2  留言主题和答复意见相似度为：77.05%
3  留言主题和答复意见相似度为：88.52%
4  留言主题和答复意见相似度为：98.21%
5  留言主题和答复意见相似度为：89.83%
6  留言主题和答复意见相似度为：76.27%
7  留言主题和答复意见相似度为：73.17%
8  留言主题和答复意见相似度为：80.70%
9  留言主题和答复意见相似度为：97.22%
10 留言主题和答复意见相似度为：72.22%
11 留言主题和答复意见相似度为：85.29%
12 留言主题和答复意见相似度为：91.67%
13 留言主题和答复意见相似度为：92.59%
14 留言主题和答复意见相似度为：72.73%
15 留言主题和答复意见相似度为：80.00%
```

5.3.2 答复意见的评价

首先，从留言答复的分词结果可知，留言答复的内容覆盖面极广，包括社区建设、交通建设、餐饮管理和车辆管理等。可见广大市民在生活中发现的问题以及政府提出的解决方案很全面，由此得出答复意见的完整性。

其次，由留言主题与留言答复较高的相似度可知，留言答复的内容并没有答非所问，而是按照留言主题所提出的问题，尽力的做出与之相关的答复，提出合理的解决方案。由此可见，政府能够虚心听取人们群众的问题以及政府解决问题的高效性，所以答复意见有较高的相关性。

然后，通过对留言答复进行预处理、统计词频、转化为稀疏变量和计算相似度等操作，我们可以得到足够多且准确的信息。这些信息有助于我们更好的理解留言答复的内容，体现了留言答复的可解释性。

最后，通过对留言答复的分析，我们可以得出政府一心为人民服务的态度，通过“智慧政务”的实行，极大幅度的提升了政府的管理水平和施政效率，政府也最终会形成高效、敏捷、便民的新型政府。

参考文献

[1]张航.基于朴素贝叶斯的中文文本分类及实现[D].山东师范大学，2018.
[2]郭永辉.面向短文本分类的特征扩展方法[D].哈尔滨工业大学,2013.
[3]黄承慧,印鉴,侯昉.一种结合词项语义信息和 TF-IDF 方法的文本相似度量方法[J].计算机学报,2011,34(5):856-864.

[4]王小林,王东,杨思春,等.基于《知网》的词语语义相似度算法[J].计算机工程,2014,
[5]王东,熊世桓.基于同义词词林扩展的短文本分类[J].兰州理工大学学报,2015,04:104-108.
[6]刘铭,吴冲,刘远超,等.基于特征权重量化的相似度计算方法[J].计算机学报,2015(07):1420-1433.
[7]江大鹏.基于词向量的短文本分类方法研究[D].浙江大学,2015.