“智慧政务”中的文本挖掘应用

摘要：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。

文章重点对已给数据预处理、数据清洗、分类等核心环节进行了阐述，并划分出了热门信息，为有关部门优化网络服务、提升工作效率提供了参考。

关键词；贝叶斯模型；数据挖掘 ；相似度分析

1 问题重述

对于问题一：将留言按照一定的划分体系进行分类，以便后续将群众留言分派至相应的职能部门进行处理。本题我们主要对数据进行处理，然后根据朴素贝叶斯分类的原理进行分类。

对于问题二：热点问题挖掘某一时段内群众集中反映的某一问题可称为热点问题，如“XXX 小区多位业主多次反映入夏以来小区楼下烧烤店深夜经营导致噪音和油烟扰民”。及时发现热点问题，有助于相关部门进行有针对性地处理，提升服务效率。该题主要采用文本相似度计算主题间关联度的大小，以此来发现热点问题。

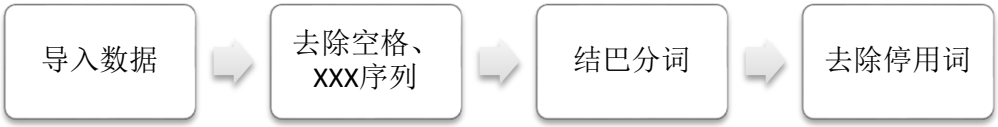
2 分析方法与过程

本文通过文本分词、关键词聚类等手段，对数据进行处理，研究市民对城市建议的模型，采用朴素贝叶斯模型对信箱内容进行分析与分类，从而揭示市民在生活遇到的问题。

2.1 网络访问数据预处理

为了去数据中的异常数据重复和其他无用数据，在进行数据处理之前，首先要对采集到的数据进行清洗，保障数据的质量和有效性。本文中需要过滤的数据主要为重复值和空值，这些信息会影响后续的网络数据建模，因此必须提前进行处理，针对常用的数据异常类型，处理方式如下：根据留言内容筛选出重复值和空值将其删除。

本次预处理流程：



对于问题一：

1.1 首先将数据中留言内容和留言分类提取出来，对留言内容进行结巴分词。本文采用 Python 自带分词——jieba 分词，对留言内容进行中文分词。

1.2 去停用词在文本处理中，停用词是指那些功能极其普遍，与其他词相比没有什么实际含义的词，它们通常是一些单字，符号以及高频的单词，比如中文中的“我、呢、了、地、吗”等，英文中的“many、come、an、a、for”等。对于停用词一般在预处理阶段就将其删除，避免对文本，造成负面影响。本文采取的是网上 1897 个停用词版本。

2.1 将处理过的数据按照留言内容是训练集，留言分类为测试集进行 8:2 分类。然后提取文本特征向量。

2.2 在模型选取的本文选取了 2 种模型朴素贝叶斯模型和贝努力贝叶斯模型。

朴素贝叶斯评分为：0.725

```
In[54]: mnb.score(X_test, y_test)
Out[54]: 0.725
```

贝努力贝叶斯评分为:0.3375

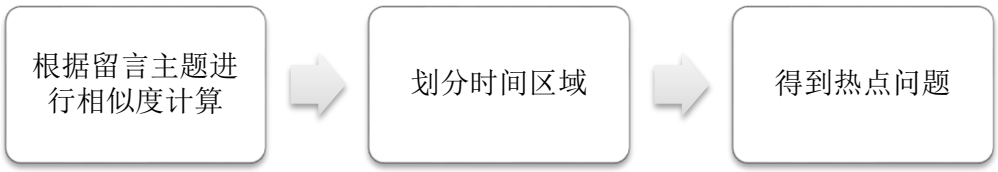
```
In[56]: mnb2.score(X_test, y_test)
Out[56]: 0.3375
```

在进行模型评分时，明显看出朴素贝叶斯模型评分高，所以本文采取的是朴素贝叶斯模型。

预测结果如下

	0	1	2	3	4	5	6	7	8	9
0	城乡建设	城乡建设	交通运输	城乡建设	交通运输	城乡建设	劳动和社会保障	城乡建设	教育文体	劳动和社会保障

对于问题二：



对于该问题主要思路是：讲留言主题中从第一行开始两两进行相似度计算（杰卡德系数），如果大于 0.3 则归为一类，否则另起一类。之后的留言主题再逐一进行相似度计算，以此类推。。

得出分类后选出留言内容最多的 5 类提取关键字作为该热点问题的具体标题和人物地点。然后通过 plt 画图计算该热点问题时间区域最集中的部分为该问题的主要时间。

3 结语

本文在深入分析市民日常网络问题基础上，对留言进行了阐述，包括留言数据预处理、数据清洗、文本分类等核心环节，并根据实际需要构建了市民留言分类系统，提出了系列解决方案，为有关部门优化网络服务、提升效率提供了参考。

代码

第一问

```
import pandas as pd
import jieba
import re

def data_process(file='附件 2.csv'):
    data = pd.read_csv(file, encoding='utf8')
    # ----------数据清洗
    # --------去除空格、XXX 序列、
    data_new = data.drop_duplicates('留言详情')  # 去除留言详情中重复的
    # --------结巴分词
    data_zt=data_new['留言主题']
    data_xq=data_new['留言详情']
    data_zt_cut=data_zt.apply(lambda x:jieba.lcut(x))      #分词。返回列表
    data_xq_cut=data_xq.apply(lambda x:jieba.lcut(x))
    # -------  去除停用词
    stopWords = pd.read_csv('停用词表.txt', encoding='utf-8', sep='hahaha', header=None)
    stopWords1 = ['\r\n','\n','\t', '    ', '      ', '    '] + list(stopWords.iloc[:, 0])
    data_zt_after_stop = data_zt_cut.apply(lambda x: [i for i in x if i not in stopWords1])
    data_xq_after_stop = data_xq_cut.apply(lambda x: [i for i in x if i not in stopWords1])
    labels = data_new.loc[data_new.index, '一级分类']  # 找到对应的标签
    adata_zt = data_zt_after_stop.apply(lambda x: ' '.join(x))  # 用空格将每个词连接起来，转
化为字符串
    adata_xq = data_xq_after_stop.apply(lambda x: ' '.join(x))  # 用空格将每个词连接起来，
转化为字符串
    adata=adata_zt + adata_xq
    data_after_stop=data_zt_after_stop + data_xq_after_stop
    return adata, data_after_stop, labels
```

模型

```
from data_process import data_process
from sklearn.model_selection import train_test_split
import pandas as pd
# 读取 txt 文件

adata, data_after_stop, fls = data_process()
from sklearn.datasets import fetch_20newsgroups

X_train, X_test, y_train, y_test = train_test_split(adata, fls, test_size=0.2)
从 sklearn.feature_extraction.test 里导入用于文本特征向量转化模块
from sklearn.feature_extraction.text import CountVectorizer
vec = CountVectorizer()
X_train = vec.fit_transform(X_train)
```

```
X_test = vec.transform(X_test)

# 从 sklearn.naive_bayes 里导入朴素贝叶斯模型
from sklearn.naive_bayes import MultinomialNB
# 使用默认配置初始化朴素贝叶斯模型
mnb = MultinomialNB()
# 利用训练数据对模型参数进行估计
mnb.fit(X_train, y_train)
# 对测试样本进行类别预测,结果存储在变量 y_predict 中
y_predict = mnb.predict(X_test)
print(y_predict)
print(mnb.score(X_test, y_test))   #模型检测
```

第二问相似度计算

```
def jaccard_similarity(s1, s2):
    def add_space(s):
        return ' '.join(list(s))
    # 将字中间加入空格
    s1, s2 = add_space(s1), add_space(s2)
    # 转化为 TF 矩阵
    cv = CountVectorizer(tokenizer=lambda s: s.split())
    corpus = [s1, s2]
    vectors = cv.fit_transform(corpus).toarray()
    # 求交集
    numerator = np.sum(np.min(vectors, axis=0))
    # 求并集
    denominator = np.sum(np.max(vectors, axis=0))
    # 计算杰卡德系数
    return 1.0 * numerator / denominator
```