“智慧政务”中的文本挖掘应用

摘要

本文主要基于自然语言处理和文本挖掘的方法，旨在解决群众留言分类、热点问题挖掘以及答复意见评价这三个问题，从而对提升政府的管理水平和施政效率具有极大的推动作用。

针对问题1，基于自然语言处理技术出发，对网络问政平台的群众留言进行深入文本挖掘，利用Python中的Jieba包，实现对群众留言内容的分词，然后对分词结果开展特征选取与降维处理，并利用TF − IDF算法进行词频统计，最后采用余弦相似性进行群众留言分类，从而建立出关于留言内容的一级标签分类模型。这种分类方法的查准率和查全率分别为49.8%和100%，因此这种分类方法能将群众留言分派至相应的职能部门处理。

针对问题2，我们利用题一的分词方法和TF − IDF算法进行切词统计、词频统计、留言内容向量化。同时拟合出三个指标来对留言问题进行热度排序，即留言问题出现次数与留言问题总数的比值、留言问题出现时间的频长与留言出现的时间总频长的比值、留言问题的点赞数与反对数之和与总的点赞数与反对数之和的比值。那么就可以用$\frac{rc}{rn}$、$\frac{rh}{\theta}$、$\frac{dz+df}{d}$三个比值来进行热点问题的筛选。为了解决这三个数值量级的不一致问题，我们便将数值较小的比值进行放大，使之在热点问题排序的过程中呈现出相同的比重。将数据处理之后我们便能够进行热点问题的筛选与确定。

针对问题3，我们将从回复的相关性、完整性和及时性三个方面去解答。相关性则利用留言问题和回复组成向量$(x_1，y_1)$与向量$(x_2，y_2)$，采用余弦相似度计算。完整性则利用留言问题和回复的比值$\frac{m}{n}\times 100\%$来进行评价。及时性则利用留言时间和回复时间的时间差来进行评价。

关键字： Jieba包　TF − IDF算法　余弦相似性　热点问题　拟合

一、问题的背景与重述

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，依靠人工去对留言进行划分和热度处理已经跟不上时代的步伐。同时，随着大数据、云计算、人工智能等技术的崛起，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起到了极大的作用。

问题 1、群众留言分类。在处理网络问政平台的群众留言时，工作人员首先按照一定的划分体系对留言进行分类，分类标准我们可以参考题目给出的附件 1 中提供的内容分类三级标签体系。对留言分类之后便将相关留言内容分派至相应的职能部门进行处理。但在当前社会情况下，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，且差错率高等诸多问题。我们就需要根据题目附件 2 给出的具体数据，建立关于留言内容的一级标签分类模型。模型建立之后使用 $\mathrm{F-Score}$ 对分类方法进行评价:具体公式为$F_1 = \frac{1}{n}\sum_{i=1}^{n}\frac{2P_iR_i}{P_i+R_i}$，其中$P_i$为第 i 类的查准率，R_i为第 i 类的查全率。

问题 2、热点问题挖掘。热点问题的定义是某一时段内群众集中反映的某一问题，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”。而及时有效的发现热点问题，有助于相关部门进行有针对性地处理，提升服务效率。我们需要根据题目给出的附件 3 的具体内容，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。并按照题目中给出表 1 的格式排列出排名前 5 的热点问题，并保存为文件“热点问题表.xls”。按表 2 的格式给出相应热点问题对应的留言信息，并保存为“热点问题留言明细表.xls”。

问题 3、答复意见的评价。需要我们针对题目给出的附件 4 中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，并尝试在数据及应用中具体实现。

二、问题分析

2.1 问题一分析

对于问题一，采用 Python中的Jieba包，实现对群众留言内容的分词。对群众留言内容进行分词后，每个词汇都可以作为标识留言详情的特征。因此，我们要对留言详情的特征进行降维处理，去除对留言详情区分程度很少的特征词汇，主要包含以下三种情形：（1）去除掉具有很高出现频率的词。（2）去除掉经常使用的特殊词，主要含有常见的称谓词、介词、语气助词、量词、常用副词、常用象声词、常用动词。（3）去除一些出现次数很少的特征词汇。再通过对分词结果进行特征选取与降维,实现对区分程度很少的词汇的过滤，留下与留言主题相关的关键词。然后利用TF-IDF算法进行词频统计，最后采用余弦相似性进行群众留言分类，从而建立出关于留言内容的一级标签分类模型。这种分类方法的查准率和查全率分别为49.8%和100%，因此这种分类方法能将群众留言分派至相应的职能部门处理。

2.2 问题二分析

对于问题二，我们可以在第一问的基础上进行求解。利用第一题给出的分词方法和 TF − IDF 算法进行切词、词频统计、留言内容向量化；然后根据词频统计将留言分成三个类别；最后利用 Single − Pass 在各类别内部进行聚类。我们拟合三个指标来对留言问题进行热度排序，即留言问题出现次数与留言问题总数的比值、留言问题出现时间的频长与留言出现的时间总频长的比值、留言问题的点赞数与反对数之和与总的点赞数与反对数之和的比值。因为留言问题与总数的比值和时间的比值不在一个量级上，所以我们为了将两者的量级统一 ，我们便将留言问题与总数的比值进行放大。同时对留言问题点赞数和反对数也进行同样的处理，使之与处于同一量级。那么我们就可以用$\frac{rc}{rn}$、$\frac{rh}{\theta}$、$\frac{dz+df}{d}$三个比值来进行热点问题的筛选与确定。

2.3 问题三分析

对于问题三，我们同样在问题一的基础上进行求解。同时我们对回复意见的评价给出了三个指标——相关性、完整性和及时性。考虑到因为问题一已经给出了留言问题的分词和关键词的处理，那么我们可以提取出留言关键词和回复关键词，之后再对他们进行一个整体相似度的比较，利用他们评价的相关度作为回复留言评价的一个相关度指标。而完整性则同样是先在留言和回复中提取出关键词，再根据留言中的关键词在回复关键词中出现的频率进行判定。对于留言的关键词在回复关键词在一个都不出现的情况，我们采用设定一个阈值来计算它是否出现在回复中，即两边的关键词相识度达到阈值之后就可以算作留言关键词出现在回复关键词中。而及时性的评价指标我们则采用回复时间与留言时间的差值进行分析，同样设定一个阈值，当回复时间超过阈值时，我们将此条回复定义为不及时；在阈值内的回复则定义为及时回复。将这三个指标联系起来对回复意见进行评价，从而给出一套具体的数字化评价方案。

三、模型假设

序号	假设	合理性分析
1	假设留言真实准确	留言信息虚假的话就会导致我们对热点问题的分析出现误差，无法得出真正的热点问题
2	假设一个账户在同一时间只能进行一次留言	这样就避免了一些用户恶意留言，以此来提高自己问题的重视程度
3	假设留言内容中都有具体的问题描述	避免用户留言问题未提交，造成留言数据空白的问题
4	假设划分体系不再改变	划分体系的改变会导致数据的分类出现问题，使计算结果出现较大误差
5	假设超过 2000 的高频词汇对分类无影响	全部数据中的高频词都是人们的日常用词，考虑进去会增大分类误差

四、符号说明

符号	符号说明
$f_{i,j}$	某词k_i在所有留言中d_j中出现的次数
N	总留言数
n_i	包含某词汇k_i的留言数
(x_1, y_1)	向量
(x_2, y_2)	向量
$\cos\theta$	向量余弦值
rc	留言问题出现的次数
rn	留言问题的总数
rh	留言问题出现的时间
θ	留言问题的总时间
ω	关注度

五、数学模型的建立与求解

5.1 问题一

通过对问题一的分析，基于自然语言处理技术出发，对网络问政平台的群众留言进行深入文本挖掘，利用Python中的Jieba包的分词技术分析群众留言，然后对所得到的分词结果开展特征选取与降维处理，而文本分词是指使用计算机自动对文本进行词语的切分。

因此，我们通过大数据软件Python中的Jieba包，实现对群众留言内容的分词。

对群众留言内容进行分词后，每个词汇都可以作为标识留言详情的特征。因此，我们要对留言详情的特征进行降维处理，而特征降维就是为了对特征进行识

别剔除，在这里就是去除掉对留言详情区分程度很少的特征词汇，从而降低后续留言内容聚类的算法复杂度，主要包含以下三种情形：[1]

（1）去除掉具有很高出现频率的词，如“领导”、“公司”、“学校”、“政府”等在群众留言中出现次数高于2000次的高频无用词汇。

（2）去除掉经常使用的特殊词，主要含有常见的称谓词、介词、语气助词、量词、常用副词、常用象声词、常用动词，如“但愿”、“但是”、“具体”、“一些”等与留言主题无关的词汇。

（3）去除一些出现次数很少的特征词汇，如“收工”、“全科”、“图像”、“摄像机”等在群众留言文本数据中出现次数低于5的低频词汇。

```
    ...:     print("去停用词: %s"%text)
分词: ['路段', 'tA3', '未管', '文明城市', '车流', '安全隐患', '西行', '加油站', '上下班', '人行道', '路灯', '整改', '人流', '围墙',
'燕子', '西湖', '路口', '安置', '大道', '施工']
去停用词: ['路段', 'tA3', '未管', '文明城市', '车流', '安全隐患', '西行', '加油站', '上下班', '人行道', '路灯', '整改', '人流', '围
墙', '燕子', '西湖', '路口', '安置', '大道', '施工']
分词: ['护栏', '在水一方', '电等', '过往行人', '锈迹斑斑', '烂尾', '四楼', '主干道', '一楼', '人行道', '牵头', '倒塌', '拆除', '占
用', '书院', '围着', '大厦', '请求', '人为', '设施']
去停用词: ['护栏', '在水一方', '电等', '过往行人', '锈迹斑斑', '烂尾', '四楼', '主干道', '一楼', '人行道', '牵头', '倒塌', '拆除',
'占用', '书院', '围着', '大厦', '请求', '人为', '设施']
分词: ['业主', '物业', '车位', '收费', '小区', '停车', '停车费', '属于', 'A1', '公摊', '程明', '西地省', '小区业主', '相关', '社
区', '征得', '物业管理', '业主大会', '服务', '地下']
去停用词: ['业主', '物业', '车位', '收费', '小区', '停车', '停车费', '公摊', '程明', '西地省', '小区业主', '相关', '社区', '征得',
'物业管理', '业主大会', '服务', '地下']
分词: ['不洗', 'tA1', 'A2', '区华庭', '水是', '健康', '雾味', '致癌物', '水箱', '我们', '环保部门', '基本', '楼顶', '长年', '自来
水', '必不可少', '供水', '用品', '日常生活', '小区']
去停用词: ['不洗', 'tA1', '区华庭', '水是', '健康', '雾味', '致癌物', '水箱', '环保部门', '楼顶', '长年', '自来水', '必不可少', '供
```

图1　剔除停用词图

再通过对得到的分词结果进行特征选取与降维，实现对区分程度很少的词汇的过滤，留下与留言主题相关的关键词。结合这道题的具体问题，对现有关键词进行特征化，变为算法可处理的向量。

由于TF − IDF模型[2]是一种比较常见的文本关键词提取方法，并且TF − IDF能够过滤掉一些非关键词，保留体现群众留言内容特征的词汇，因此我们采用TF − IDF（词频-逆文档频率）算法计算关键词重要性权重值。

TF − IDF（词频-逆文档频率）算法计算公式如下：

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{1}$$

$$idf_i = \log \frac{|D|}{|\{j:t_i \in d_j\}|} \tag{2}$$

其中，某词k_i在所有留言中d_j中出现的次数为$f_{i,j}$，总留言数为N，以及包含某词汇k_i的留言数为n_i。

为了保证分母不为0，即某词汇没有出现在某一留言中，因此IDF公式中的分

母改为$1+\{j:t_i \in d_j\}$，得到某词k_i相对于留言d_j的$\mathrm{TF}-\mathrm{IDF}$权重计算公式如下：

$$tfidf_{i,j}=\frac{f_{i,j}}{\sum_k f_{k,j}}\times \log\frac{N}{n_i+1} \tag{3}$$

通过$\mathrm{TF}-\mathrm{IDF}$(词频-逆文档频率)算法计算出关键词,然后采用余弦相似性进行群众留言分类，余弦相似度可以计算两个向量间的夹角，通过夹角的余弦值表示两向量的接近程度，如果两向量余弦值越接近1，即两向量越相近，在留言特征向量中，表示两个留言在同一类别中的可能性大。

向量$(x_1，\ y_1)$与向量$(x_2，\ y_2)$的余弦相似度计算公式如下：

$$c^2=a^2+b^2-2ab\cos\theta \tag{4}$$

其中：

$$a^2=x_1^2+y_1^2$$

$$b^2=x_2^2+y_2^2$$

$$c^2=(x_1-x_2)^2+(y_1-y_2)^2 \tag{5}$$

于是，可以得到下列公式：

$$\begin{aligned}\cos\theta&=\frac{a^2+b^2-c^2}{2ab}\\&=\frac{2(x_1x_2+y_1y_2)}{2\sqrt{x_1^2+y_1^2}\sqrt{x_2^2+y_2^2}}\\&=\frac{A\cdot B}{\|A\|\cdot\|B\|}\end{aligned} \tag{6}$$

最后把留言详情和分类标准中相似度最大的归为一类，以此类推，就可以得到群众留言的分类。由此，可以得到，模型的查全率为100%,遗漏率为0%,对群众留言内容能够全部识别,但从查准率指标来看，查准率为49.8%,表明分类精确不够高。

使用$\mathrm{F}-\mathrm{Score}$对分类方法进行评价:

$$F_1=\frac{1}{n}\sum_{i=1}^{n}\frac{2P_iR_i}{P_i+R_i} \tag{7}$$

其中P_i为第i类的查准率，R_i为第i类的查全率。

由 Python 程序运行可以得到$F_1=0.39$。

5.2 问题二

对于问题二，我们将采用问题一分词方法和 TF − IDF 算法进行切词、词频统计、留言内容向量化；然后根据词频统计将留言分成三个类别；最后利用 Single − Pass 在各类别内部进行聚类，从而提取出热点问题。分词后得到的部分数据如下图 2 所示：

图 2 分词后留言中关键词出现次数

留言热度评价指标主要考虑留言时间跨度长短、留言内容出现次数、点赞数和反对数等。

设 rc 为留言问题出现的次数，rn 为留言问题的总数，rh 为留言问题出现的时间（即出现了多少天），θ 为留言问题的总时间，ω 为关注度。

因为留言问题出现次数和留言问题出现时间都与问题的关注度成正比，所以一定时间内群众留言问题的关注度可以用下述公式来进行定量描述：

$$\omega = \frac{rc}{rn} \times 10 + \frac{rh}{\theta} \tag{8}$$

rc 与 rn 的比值可以反映此问题的出现频率，而θ 则是这段时间的具体数值，可以根据我们所期望的时间来进行具体设置。又因为留言的问题出现频率较全部留言问题来说频率一般不会大于 0.01，而 rh 与 θ 的比值则完全可以大于 0.01，所以我们为了使之在一个量级进行合并运算，所以我们将 $\frac{rc}{rn}$ 的值进行了处理再使之与 $\frac{rh}{\theta}$ 相加。最后我们便可以根据运算结果给出排名前 5 的热点问题如表 1 所示，热点问题留言明细表见附件。

表 1-热点问题表

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	278	2019/7/28 至 2020/1/7	A 市利保壹号公馆	夜间街道上噪音扰民
2	2	230	2019/6/19 至 2020/1/6	A 市桔园小区	社区物业不负责
3	3	113	2018/10/27 至 2020/1/7	A1 区地铁五号线	地铁超时作业，影响住户
4	4	110	2019/04/28 至 2019/9/9	A 市教师学生	学校有多方面违规
5	5	61	2019/5/30 至 2020/1/6	A 市西湖公园	A 市多处地点存在安全隐患

5.3 问题三

对于问题三，我们可以利用问题一的结论继续进行求解。在解答回复相关性时我们可以利用留言问题和回复组成向量$(x_1，y_1)$与向量$(x_2，y_2)$，那么余弦相似度计算公式如下：

$$c^2 = a^2 + b^2 - 2ab\cos\theta$$

其中：

$$a^2 = x_1^2 + y_1^2$$

$$b^2 = x_2^2 + y_2^2$$

$$c^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$$

于是，可以得到下列公式：

$$\cos\theta = \quad = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

那么我们便可以根据留言问题和回复的相似度的出相关度的评价标准。而完整性的评价标准，我们则可以先提取留言关键词 m 和回复关键词 n，再计算m在n中出现的比例。其中当m与n的相似度$\csc\theta \geq \vartheta$，$\vartheta$为阈值，则

$$m = n$$

$$\frac{m}{n} \times 100\% = \delta \tag{9}$$

那么δ的值就可以作为回复完整性的一个评价标准。最后再计算回复的及时性有，设留言时间为τ_1，回复时间为τ_2，同时设定一个阈值τ，当

$$\tau_2 - \tau_1 > \tau \tag{10}$$

我们可以评价这条回复在及时性方面为不及时。

$$\tau_2 - \tau_1 < \tau \tag{11}$$

我们评价这条回复在及时性方面为及时。

这样我们就得出相关性、完整性和及时性三个方面的评价标准，可以对回复意见的质量给出一套评价方案并得以实现。

六、模型的优缺点

6.1 模型的优点

- 该模型运用的知识结构较为简单，容易被更多人掌握理解；
- 该模型对于数据的精确度把握要求较高，使得数据的整体精确度增加；
- 该模型是在已给出数据的情况下进行运算的，在实际使用时可以将实时数据进行导入，第一时间得出结果，实用性和可塑性较高。

6.2 模型的缺点

- 对于用户留言格式要求较高；
- 建立模型与处理数据的过程中存在误差，操作不够严谨；
- 数据的预处理上误差较大；
- 对于大量数据的处理效果不算太好。

七、参考文献

[1] 吴刚勇，张千斌，吴恒超，顾冰；基于自然语言处理技术的电力客户投诉工单文本挖掘分析[J] ；2018 年

[2] 郭英杰；基于主题的文本挖掘及可视化系统研究与实现[D] ；2018 年

八、附录

问题一程序：

```
import pandas as pda
import numpy as np
import jieba.analyse
import jieba
from jieba import lcut
from gensim.similarities import SparseMatrixSimilarity
from gensim.corpora import Dictionary
from gensim.models import TfidfModel

fname01="D:/C 题全部数据/附件 1.xlsx"
fname02="D:/C 题全部数据/附件 2.xlsx"
data01=pda.read_excel(fname01)
d01=data01.loc[:,['一级分类']]
getd=np.array(d01).astype(str)  #将数据转化为数组
data1=getd.tolist() #将数组转换为列表
#列表去重
newdata1 = []
for id in data1:
    if id not in newdata1:
        newdata1.append(id)

#获取分类列表并去重
cl_list = list(data01['一级分类'].drop_duplicates())
#按照类别分文件存放数据
j=1
for i in cl_list:
    iris1 = data01[data01['一级分类']==i]
```

```
    iris1.to_excel('D:/C 题全部数据/work/%s.xlsx'%(j))
    j=j+1

#读取三级分类
flist=[]
for ty in range(1,16):
    dt=pda.read_excel('D:/C 题全部数据/work/%s.xlsx'%(ty))
    gda=np.array(dt.loc[:,['三级分类']]).astype(str)  #将数据转化为数
组
    dd1=gda.tolist() #将数组转换为列表
    newd = []
    for id in dd1:
        if id not in newd:
            newd.append(id)
    flist.append(newd)

data02=pda.read_excel(fname02)
d03=data02.loc[:,['留言详情']]
getd3=np.array(d03).astype(str)
data13=getd3.tolist()
data4=data13[0:10]

mfl=[]
fenlei=[]
#将文本集生成分词列表
for ke in data13:
    texts = [lcut(str(text)) for text in flist]
    # 基于文本集建立词典，并获得词典特征数
    dictionary = Dictionary(texts)
```

```
    num_features = len(dictionary.token2id)
    #基于词典，将分词列表集转换成稀疏向量集，称作语料库
    corpus = [dictionary.doc2bow(text) for text in texts]
    #同理，用词典把搜索词也转换为稀疏向量
    tag=jieba.analyse.extract_tags(str(ke))
    kw_vector = dictionary.doc2bow(lcut(str(tag)))
    #创建 TF-IDF 模型，传入语料库训练
    tfidf = TfidfModel(corpus)
    #用训练好的 TF-IDF 模型处理被检索文本和搜索词
    tf_texts = tfidf[corpus]  # 此处将语料库用作被检索文本
    tf_kw = tfidf[kw_vector]
    #相似度计算
    sparse_matrix = SparseMatrixSimilarity(tf_texts, num_features)
    similarities = sparse_matrix.get_similarities(tf_kw)
    ss=list(similarities)
    mfl.append(max(ss))
    fenlei.append(ss.index(max(ss)))

fenshj=[]
for fen in fenlei:
    fenshj.append(newdatal[fen])

d04=data02.loc[:,['一级标签']]
getd4=np.array(d04).astype(str)
datal4=getd4.tolist()
sum=0
for xx in range(len(fenshj)):
    if fenshj[xx]==datal4[xx]:
        sum=sum+1
```

```
#总体分类准确率 zql
zql=sum / (len(data14))
print("分类的整体准确率：%f"%zql)

yjbq=data02.loc[:,['一级标签']]
gety=np.array(yjbq).astype(str)
datayi=gety.tolist()
yiji = []
for id in datayi:
    if id not in yiji:
        yiji.append(id)

su1=0
su2=0
su3=0
su4=0
su5=0
su6=0
su7=0
for zq in range(0,len(fenshj)):
    if zq<=2009 and fenshj[zq]==yiji[0]:
        su1=su1+1
    if zq>=2010 and zq<=2947 and fenshj[zq]==yiji[1]:
        su2=su2+1
    if zq>=2948 and zq<=3560 and fenshj[zq]==yiji[2]:
        su3=su3+1
    if zq>=3561 and zq<=5149 and fenshj[zq]==yiji[3]:
        su4=su4+1
    if zq>=5150 and zq<=7118 and fenshj[zq]==yiji[4]:
```

```
            su5=su5+1
        if zq>=7119 and zq<=8333 and fenshj[zq]==yiji[5]:
            su6=su6+1
        if zq>=8334 and fenshj[zq]==yiji[6]:
            su7=su7+1

zhunq=[su1,su2,su3,su4,su5,su6,su7]
shul=[2009,937,612,1588,1968,1214,876]
#计算每一类的准确率
leizql=[]
for fen in range(0,len(zhunq)):
    zql=zhunq[fen]/shul[fen]
    print("%s 的准确率：%f"%(yiji[fen][0],zql))
    leizql.append(zql)

#计算 F
fs=0
for it in range(0,len(zhunq)):
    fs=fs+(2*leizql[it]*1)/(leizql[it]+1)
fs1=fs*(1/len(yiji))
print("F 的值：%f"%fs1)
```

问题二代码

```
import jieba
import pandas as pda
import numpy as np
import jieba.analyse

fname03="D:/C 题全部数据/附件 3.xlsx"
data033=pda.read_excel(fname03)
dd3=data033.loc[:,['留言主题']]
getdata=np.array(dd3).astype(str)  #将数据转化为数组
dlist=getdata.tolist() #将数组转换为列表
ddata=dlist[0:5]
#分词去除停用词
lrow2 = str(dlist).replace('\\n','').replace('\\t','')
jb2=jieba.lcut(lrow2,cut_all=False)
with  open(r"D:\CNKI 论 文 \ 停 用 词 表 .txt","r",encoding='utf-8')  as
stopfile:  # 读取停用词，用 utf-8 的编码格式
    txt = stopfile.readlines()  # 一次性将所有的词按行读进来
    stopword = set(word.strip('\n') for word in txt)  # 去除每个词后的
换行符放入 tuple 中
text = [word for word in jb2 if word not in stopword]  # 去除停用词
#去除字数为 1 的词
textt=[]
for x in text:
    if len(x)>1:
        textt.append(x)
#统计词频
dicc = {}
for stt in textt:
```

```
    if stt in dicc.keys():
        dicc[stt] = dicc[stt] + 1
    else:
        dicc[stt] = 1

#排序
dicl= sorted(dicc.items(),key=lambda d:d[1],reverse= True)
#print(dicl)

f=open(r'D:\cipintj.txt','w',encoding='utf-8')
#fp=open(r'D:\cptj.txt','w',encoding='utf-8')
f.write(str(dicl))
f.close()
```