第八届“泰迪杯”数据挖掘挑战赛——C 题：“智慧政务”中的文本挖掘应用

摘要：

本文通过对“智慧政务”中的文本挖掘，建立了一套基于留言详情的使用 TextCNN 构建的文本分类模型，由测试数据可以得出其分类模型较为合理。对于文本热点问题挖掘，本文从文本相似度出发，由 word2vec 计算出文本向量，并应用谱聚类方法进行聚类求解出排名前 5 的热点问题。关于主题回复的评价，本文通过从 4 个方面对主题的回复质量进行量化描述,通过权重系数加权求和得到主题回复质量评价模型。结果表明,该模型可以很好地评价一个主题回复的质量。

关键词：智慧政务、文本挖掘、TextCNN、word2vec、热点问题、回复质量

一、问题重述

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，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。

根据收集自互联网公开来源的群众问政留言记录，及相关部门对部分群众留言的 答复意见。请利用自然语言处理和文本挖掘的方法解决下面的问题。

(1)、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，以便后续将群众留言分派至相应的职能部门处理。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。请根据已给数据，建立关于留言内容的一级标签分类模型。

(2)、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。请根据已给数据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。

(3)、针对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。

二、问题分析

2.1 对于问题一的分析

网络问政留言的分析对相关政府部门了解民生、制定政策显得越来越重要。例如，政府机构可能对新闻中涉及自然灾害或者食品安全的相关报道比较关心，同时也想通过网络留言了解社会民众对这些事件的反应。

这项需求涉及到一项非常关键的技术，那就是文本分类，更准确的说是短文本分类。分类是缩小信息规模的一种十分有效的手段，当前依然是吸引国内外众多学者投入精力的研究热点。现在的自动分类系统主要是基于统计学原理，采用机器学习的方法在经过人工标注的数据集上进行训练得到分类器，进而对没有标签的新数据进行分类。

群众网络留言具有如下特点:(1)长度短，短则几个词，长也不过几十个词，信息单元少;(2)在开放域状态下，词语总量大，用词重复率不高;(3)信息更新快，新词出现频繁。这些因素导致当前主流的分类方法在应对留言分类方面略显不足[1]。

因此，我们在针对网络留言的特点上选取 TextCNN 算法，TextCNN 通过一维卷积来获取句子中 n-gram 的特征表示，对文本浅层特征的抽取能力很强，在短文本分类时效果很好，应用广泛，且速度快。

2.2 对于问题二的分析

热点问题反映了某一时间段群众最关心的问题，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。为了更好地解决民生热点问题，我们引入图论中的经典算法谱聚类。把所有的数据看做空间中的点，这些点之间可以用边连接起来。距离较远的两个点之间的边权重值较低，而距离较近的两个点之间的边权重值较高。通过对所有数据点组成的图进行切图，让切图后不同的子图间边权重和尽可能的低，而子图内的边权重和尽可能

的高，从而达到聚类的目的。对高稀疏的文本向量有较好的聚类效果。

2.3 对于问题三的分析

随着互联网的不断发展，中国互联网及其相关配置已逐步成熟。同时互联网政府服务不断完善，中国网民参与政府服务的积极性逐步提升，参与度逐步增. 政府要对广大网民的建设性留言提议及时吸纳，对困难要及时帮助，对不了解情况的要及时宣介。让互联网成为了解群众、贴近群众、为群众排忧解难的新途径，成为发扬人民民主、接受人民监督的新渠道。

因此建立了判定回复信息质量优劣的准则尤为重要,我们将从多个方面对回复文本进行了量化描述.利用 word2vect 算法对政府回复历史数据库进行训练,根据文本进行了量化分析,结果表明,该模型能够很好地评估政务系统的回复质量

三、本文实验环境配置

项目	配置
操作系统	Windows10
开发语言	Python
开发平台	Pycharm
加载程序包	jieba，tensorflow，pandas,等

四、群众留言分类

网络问政留言的分析对相关政府部门了解民生、制定政策显得越来越重要。例如，政府机构可能对新闻中涉及自然灾害或者食品安全的相关报道比较关心，同时也想通过网络留言了解社会民众对这些事件的反应。

其基本流程图如下：

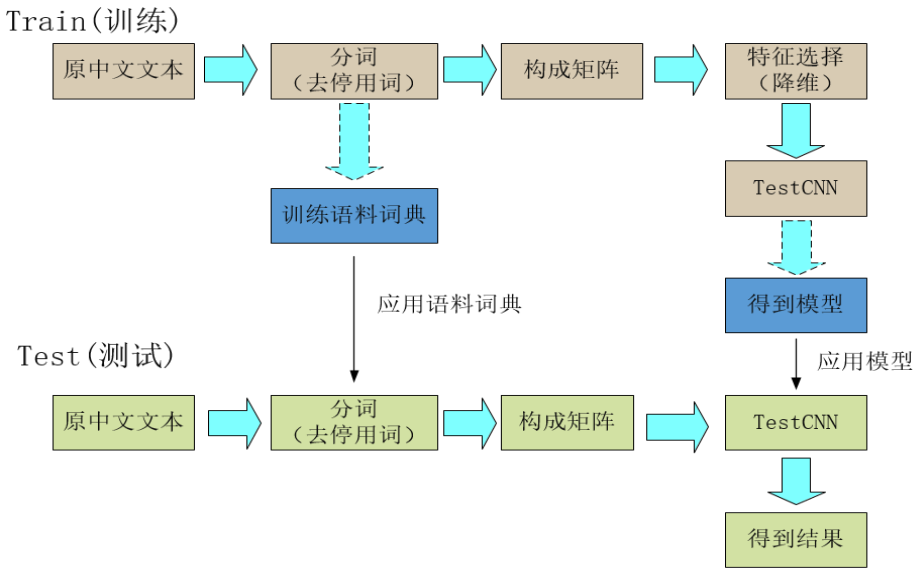

图 1.群众留言分类流程图

4.1 文本的预处理

文本分类处理的数据是以自然语言形式存在的文本集，由于文本数据的无结构化或半结构化特性，其缺乏计算机可理解的语义信息，因此需要将文本转换为能够被计算机处理的结构化形式，文本预处理作为其中的一步，对于最后聚类结果的好坏起着关键性作用。文本预处理主要包括分词、停用词过滤、移除文本中的标点符号等。

- 分词：文本数据由一系列的词语构成。对于中文文本，由于词语之间没有明显的分隔标记，所以需要进行分词处理，根据语法结构将文本中的词语抽取出来。
- 停用词过滤：停用词指的是那些仅仅在文本结构上起着词汇联系的作用，而本身不包含任何语义的词，如“了”、“的”、‘“然后”等。这类词汇在文本中有着较高的出现频率，但对于文本的语义表达没有任何贡献，更多的是语法层面的作用。由于不同类别文本间可能包含相同的停用词，这使得文本间语义界限变得模糊不清。对停用词进行过滤处理能有效减少噪音、降低文本向量的维度，进而提高文本分类的效果。

将文本进行分词处理后，由于训练集的数量较大，若直接使用此分词结果进行模型训练势必会增加模型的复杂度、时间效率低下等问题。故采用对所有文本计算词向量，而后根据词向量，去除文本中的前 N 个词，表示这 N 个词是比较重要的。由此在保证模型准确率的前提下，提高模型的训练效率。

4.2 Word2vec 模型（计算词向量）

Word2vec 是 Google 的研究员于 2013 年发布的一种基于深度学习的词向量生成工具。目标是要从海量的文档数据中学习高质量的词向量，该词向量在语义和句法上都有很好的行为。通过使用这份词向量可以去完成自然语言处理任务，比如词性标注、命名体识别、短语识别、语义角色标注等等。

Word2vec 核心结构是简化的神经网络。主要用到 Continuous Bag-of-Words Model (CBOW)和 Continuous Skip-gram Model (Skip-gram)两种模型。

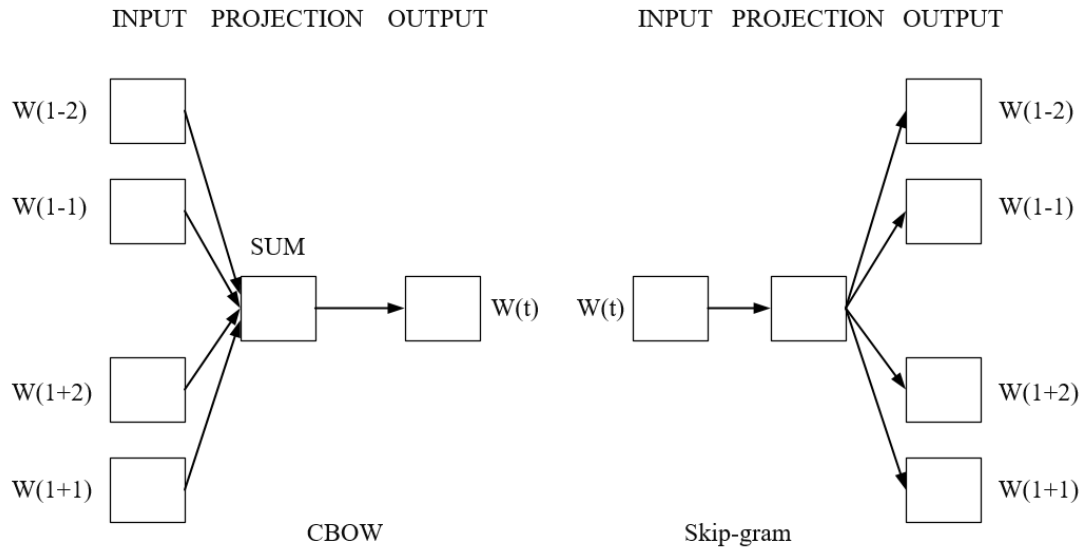

图 2.CBOW 模型和 Skip-gram 模型

Word2vec 的主要结构是一个两层的神经网络，由输入层，投射层和输出层组成，一般地，给定训练单词序列 $w_1, w_2, w_3 ... w_T$ ，目标函数为

$$\frac{1}{T}\sum_{t=k}^{T-k}\log P(w_t \mid w_{t-k},...,w_{t+k})$$

通常预测任务使用多分类器完成，使用 softmax 函数将数值进行归一化。

$$P(w_t \mid w_{t-k},...w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

上式中的 y_i 为预测的每个词语 i 的概率，计算公式如下

$$y = b + Uh(w_{t-k},....w_{t+k};W)$$

这里 U 和 b 是 softmax 参数，h 为词向量的平均值。

虽然上述过程去除了投射层，但其运算量仍然很大。假设预料中有 10000 个不同的词语，若嵌入一个 500 维的词向量，那么 word2vec 的输入一隐层权值矩阵和隐层一输出层的权值矩阵维度都[10000,500]。这样在神经网络学习时将十分缓慢，同时需要大量的训练样本来调整权重，以避免过拟合。因此本文中的 word2vec 用分层 softmax 方法优化训练过程:

分层 softmax:使用 Huffman Tree 来编码输出层的词典。将结果输出转化为树形结构，在每个分叉进一步与测试，将问题转换为二分类预测，且每个预测都有概率，最后将概率想乘就构成了似然函数。在计算过程中，只需要计算路径上所有非叶子节点词向量的贡献即可。

4.3 TextCNN 模型

使用 CNN 模型处理文本分类任务简单、高效、又能获得很高的准确率。算法流程如下:

(1)在输入层：针对传统 one-hot 造成的语义稀疏性，文本特征表示不理想等问题，本文基于词的嵌入表示作为卷积神经网络模型的输入，并融合留言主题信息对特征进行丰富和补充，使得文本的特征表示的过程中能够提取到较多地浅层文本语义信息。

(2)在 CNN 的输入层和卷积层之间引入了注意力机制，通过一种动态分配权重的策略，集中注意力资源在影响文本中的关键特征上，对贡献大的特征赋予更大的注意力概率值。

(3)进行 CNN 特有的卷积操作和池化操作，通过设计不同的卷积核尺寸和数量，充分提取文本深层蕴含的语义信息，从更高层次上学习到文本全面地向量表示，并利用最大池化技术对特征进行采样和降维，完成特征的二次提取。

(4)最后将池化层输出的特征向量在全连接层进行连接，通过 softmax 分类器得到文本所属的类别，并使用了 drop out, batch normalization 和正则化技巧加速模型训练，完成文本分类任务。

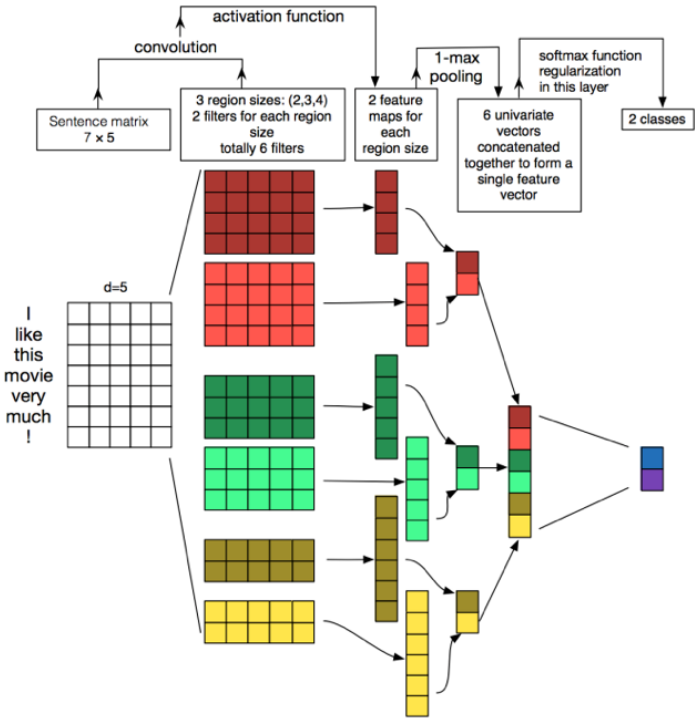

图 3.Text CNN 过程图

4.4 模型的求解

我们根据已给的“附件 2.xsl”统计可得 9210 条留言，留言可分为 7 类，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、商贸旅游和卫生计生。由于各类别留言数相差较大，因此我们采取分层抽样的方法，抽取各类别 90%的留言作为训练集，剩下 10%为测试集。数据如下

表 1.群众留言分类表

	留言总数	抽取训练样本数	测试样本数
城乡建设	2009	1800	209
环境保护	938	810	128
交通运输	612	540	72
教育文体	1588	1440	148
劳动和社会保障	1963	1800	163
商贸旅游	1215	1080	135
卫生计生	876	810	66

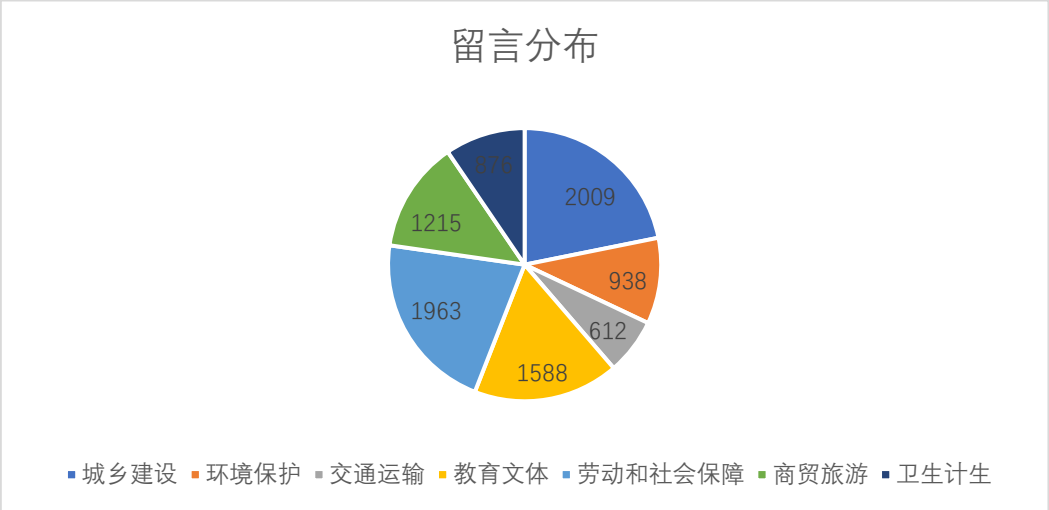

图 4.群众留言分布饼图

对此本文采用 python 的 gensim 程序包，采用 Skip-gram 模型对分类模型中的训练集和测试集中的全部文本进行维度为 100 的词向量训练。并由此得出文本中前 9999 个比较重要的词。TextCNN 中的模型参数如下：

```
embedding_size = 100      #dimension of word embedding
vocab_size = 10000        #number of vocabulary
pre_trianing = None       #use vector_char trained by word2vec

seq_length = 300          #max length of sentence
num_classes = 7          #number of labels

num_filters = 128         #number of convolution kernel
kernel_size = [2, 3, 4]  #size of convolution kernel

keep_prob = 0.5           #droppout
lr= 0.001                 #learning rate
lr_decay= 0.9             #learning rate decay
clip= 5.0                 #gradient clipping threshold

num_epochs = 8            #epochs
batch_size = 8           #batch_size
```

4.5 模型验证与结果评价

4.5.1 F-score

文本分类模型的有精确率(Precision)和召回率(Recall)评估指标。这两项看似没有什么必然的相关性，但是在大规模数据集合中，这两项指标往往是相互制约的。理想情况下做到两个指标都高当然最好，但一般情况下，精确率高，召回率就低，召回率高，精确率就低。所以我们需要综合权衡这 2 个指标，这就引出了一个新的指标 F-score。这是综合考虑精确率和召回率的调和值。

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

其中 P_i 为第 i 类的精确率，R_i 为第 i 类召回率。

表 2.留言分类 Precision 值、Recall 值和 F 值

	PRECISION	RECALL	F-SCORE	SUPPORT
城乡建设	0.87	0.82	0.84	209
环境保护	0.73	0.86	0.79	128
交通运输	0.83	0.79	0.81	72
教育文体	0.94	0.96	0.95	148
劳动和社会保障	0.87	0.94	0.91	163
商贸旅游	0.88	0.82	0.85	135
卫生计生	0.83	0.83	0.83	66
ACCURACY			0.88	

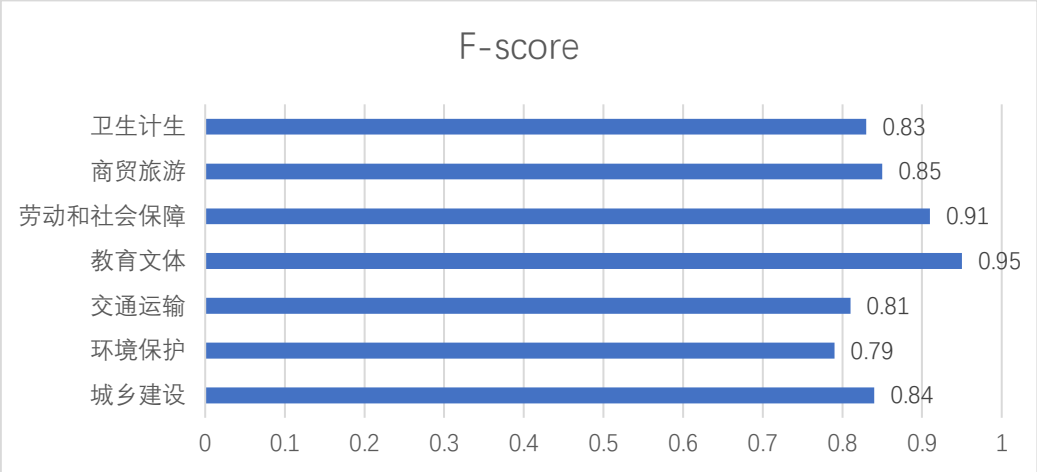

图 5.各类别 F 值的直方图

由以上图表可以看出，TextCNN 模型在教育文体类效果最好可达 0.95，在环境保护类效果效果较差为 0.79。通过 TextCNN 对分类模型进行训练后，其总体分类效果为 0.88。

4.5.2 模型的评价

1) 通过 CNN 的方式捕捉 n-gram 特征，本质上是词袋模型。计算速度快，预性能好；
2) 在文本分词中，涉及到许多地名，添加完整的地名作为停用词，将提高分类效果；
3) 由于每个分类标签所对应的测试集数量不同，因而存在对于某些分类标签其 F1 分数与其他存在较低的表现。
4) 群众留言具有信息更新快，新词出现频繁等问题，对文本分类有一定的干扰。

五、热点问题的挖掘

5.1 谱聚类（spectral clustering）算法

谱聚类是一种基于图论的聚类方法——将带权无向图划分为两个或两个以上的最优子图，使子图内部尽量相似，而子图间距离尽量距离较远。比起传统的 K-Means 算法，谱聚类对数据分布的适应性更强，聚类效果也很优秀，同时聚类的计算量也小很多.

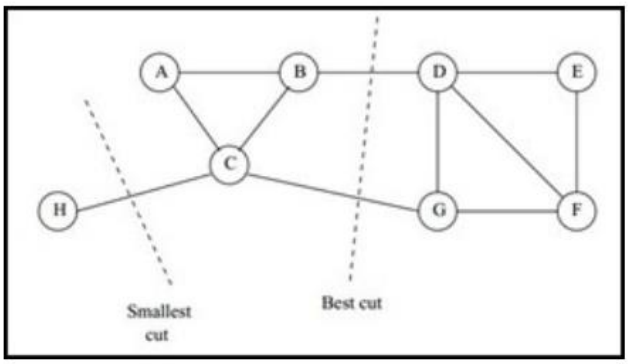

图 6.图 G 的聚类分割

5.1.1 留言详情无向权重图

对于给定图 G(V，E)，其中$V=\{v_1,v_2,...v_n\}$是图 G 的顶点集合，$E=\{e_1,e_2,...e_n\}$是图 G 的边集。对于图 G 中的任意两个点，可以有边连接，也可以没有边连接。我们定义权重w_{ij}为点v_i和点v_j之间的权重。由于本文中的留言详情之间不存在指向关系，所以有$w_{ij}=w_{ji}$，即$W=[w_{ij}]=W^T$。

对于有边连接的两个点v_i和v_j，$w_{ij}>0$。对于没有边连接的两个点$w_{ij}=0$。对于图中的任意一个点v_i，它的度d_i定义为和它相连的所有边的权重之和，即

$$d_i=\sum_{j=1}^{n}w_{ij}$$

利用每个点度的定义，我们可以得到一个 nxn 的度矩阵 *D* 它是一个对角矩阵，只有主对角线有值，对应第 i 行的第 i 个点的度数，定义如下

$$D=\begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{pmatrix}$$

构造“非规范化”拉普拉斯矩阵$L=D-W$。

我们将“附件 3.xsl”的留言详情通过 word2vec 提取词向量，生成文本向量$X=[x_1,x_2,...x_n]$.由于本文数据为文本类型，具有一定语义内涵，所以在构造相似矩阵$S=[s_{ij}]$时，令

$$s_{ij}=\begin{cases}\cos<x_i,x_j> & i\neq j\\ 0 & i=j\end{cases}$$

计算权重矩阵 W 时，令

$$w_{ij}=\begin{cases}e^{-\frac{\|x_i-x_j\|_2}{2\sigma^2}} & i\neq j\\ 0 & i=j\end{cases}$$

5.2 谱聚类算法流程

输入：相似矩阵 S，聚类个数 k
输出：k 个聚类结果

step1. 输入 S，计算 $L=I-D^{-\frac{1}{2}}SD^{-\frac{1}{2}}$

step2. 计算 L 的 k 个最小特征值对应的特征向量 $U_k=[u_1,u_2,...u_k]\in R^{n\times k}$，$U_k=\begin{bmatrix}v_1^T\\ v_2^T\\ \vdots\\ v_n^T\end{bmatrix}$，

其中 $v_i\in R^k, i=1,2,...n$

step3. 用 k-means 方法将 $\{v_i^T\}_{i=1}^n$ 进行聚类，得到 k 类，输出聚类结果 C。

5.3 模型的求解与评价

5.3.1 热点问题分类结果

“附件 3.xsl”共计留言 4326 条，其中有 118 条留言重复，将其删去后共计 4208 条有效留言，文本向量化后带入谱聚类算法，将留言分成 400 类，其中前 5 项热点问题如下：

表 3.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	35	2019/1/28-2019/12/30	A7 县泉塘街	扰民
2	2	32	2019/2/12-2020/1/7	A7 县星沙镇	政务建议
3	3	30	2019/1/6-2020/1/6	A5 区	房屋问题
4	4	27	2019/1/4-2019/12/11	A 市鑫天鑫城小区	小区供水水质
5	5	27	2019/1/21-2019/12/2	A 市	押金不退问题

表 4.前 5 项热点问题具体留言顺序号

热度排名	具体留言顺序号
1	35,148,538,741,1012,1171,1180,1222,1436,1525,1538,1876,1901,2053,2091,2150,2162,2405,2660,2685,2720,2757,3016,3113,3283,3394,3402,3580,3792,3930,3962,3979,4009,4054,4140
2	162,252,270,617,766,803,961,962,1161,1309,1345,1346,2023,2065,2105,2107,2391,2511,2700,2827,2858,3134,3149,3176,3308,3524,3850,3892,3986,4007,4125,4166
3	16,165,183,206,238,339,750,843,847,969,1154,1448,1487,1816,1874,1959,2097,2176,2237,2372,2472,2563,2629,2947,3329,3373,3501,3602,4059,4124
4	479,576,812,822,914,1050,1099,1166,1236,1265,1267,1387,1672,1754,1990,2111,2566,2635,2824,3081,3294,3386,3464,3559,3731,3988,4144
5	177,305,506,690,887,947,1389,1415,1649,1708,1753,1919,1937,1942,1943,2156,2181,2354,2402,2465,2481,2649,2750,3096,3852,4082,4165

表 4.热点留言问题明细部分表

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	188820	A00028138	A 市星沙城区旧城区棚户改造项目范围是什么?	2019/2/21 11:38:34	领导好：目前星沙城区旧城区棚户改造项目正如火如荼的进行当中…	0	0
1	191794	A0005420	A7 县泉塘社区东七路与漓楚路交汇处斑马线如何安全通过	2019/3/11 17:32:46	A7 县泉塘社区东七路与漓楚路交汇处，孩子们每天过的马路斑马线…	0	2
1	201022	A000108567	咨询 A7 县东十一路的建设进展及泉塘片区发展规划问题	2019/5/15 12:55:58	本人打算在 A7 县泉塘片区的华远海蓝郡置业居住，目前该片区多为工厂区域…	0	1
1	206171	A00050328	A7 县早安星城周边工厂废气扰民问题求解决	2019/1/28 19:12:12	早安星城业主难忍周边工厂废气污染，纷纷挂出抗议白旗…	0	1
			……				
5	289012	A0007408	在 A 市江山帝景买房遇到消费陷阱	2019/2/26 16:01:22	我于 2018 年 12 月 6 日在 A3 区江山帝景售楼中心交付二万元购买…	0	1

5.3.2 模型的评价

1) 我们将文本数据转化为图论问题解决，利用谱聚类处理稀疏数据比传统聚类方法效率高，准确度高。
2) 在留言问题中，大部分留言地点模糊，只写小区名或者街道名，这对热点问题聚类造成一定的影响，在今后的留言中，群众要写清楚 X 市 X 区（X 县）将会对按地点的热点问题挖掘提供便利。

六、答复意见的评价

留言板是沟通政府与百姓的桥梁,是政府了解民意和百姓诉求,解决百姓实际困难的一个窗口，因此留言答复意见显得尤为重要。答复内容是否能切实解决群众问题,是否详尽，是否依法依规依程序办事,答复是否及时都是政府为人民服务中需要关注的重要问题

。

根据已给的留言回复数据,我们从回复的相关性，完整性，可解释性等进行归纳总结得到以下优劣评判特征。

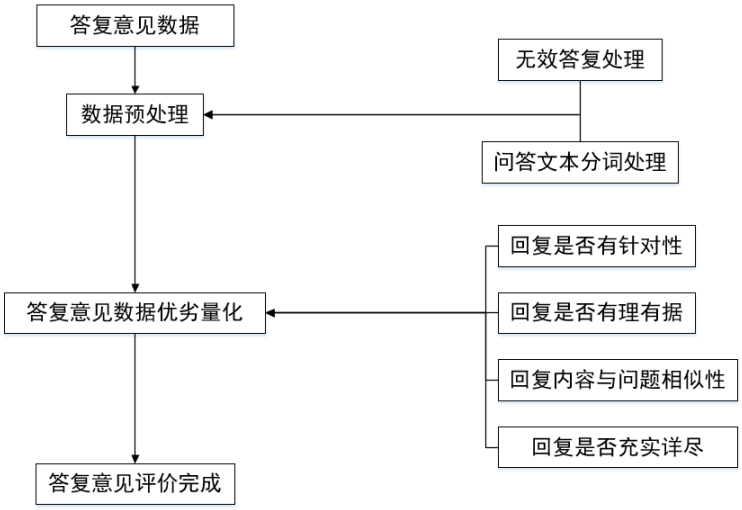

图 7.留言答复评价流程图

6.1 评价指标

6.1.1 回复是否有针对性

在政务网站回复过程中,要避免提供过多无效信息而影响评判,字数虽多但信息量少，没有解决问题。我们在 2816 条留言回复中筛选出 70 条如以下所示回复的无效回复，并将这 70 条回复剔除。

留言详情： 我投诉 J 市邦万装修公司拖欠农民工工资，我和一些工友于 2018 年 3 月份在 K10 县搬迁项目工地上刮胶做了 2 个多月，工程当年 5 月份已完工但工程款还欠 2 万多至今还没付。

留言回复： 您的留言已收悉！我们已将您反映的问题转相关部门进行处理，敬请关注后续回复，谢谢！

6.1.2 回复是否有理有据

政府回复的内容需要有严密的逻辑和准确的表达,体现依法依规按程序办事.假设剔除部分坏数据后的剩余文本数 据中共有 N_0 条比较规范的问答数据(一问一答作为一条数据).将引用法律条文或者政府相关文件通知时出现的关键词（如“根据《XX》法”或关于《XX 通知》）和回复中的文本进行语义匹配.若在留言回复的文本中匹配到相应的关键字,则认为该条回复引用了法律条文或政府通知进行解答[3]。统计回复信息有引用的回答数目,计算出现频率,即 N_l 与 N_0 的比值.该频率值可以作为此评价项,记为

$$F_1 = \frac{N_l}{N_0}$$

6.1.3 回复内容已问题主题相似性

政府回复要切实解决群众问题，为此，计算出所有留言详情和留言回复组成的文本的词向量，同时为减少计算量并使得文本的向量表示的维度相同。分别对留言详情和留言回复提取关键字，由关键字在词向量中所对应的向量代表文本的向量。两文本相似度采用余铉相似度进行计算：

$$F_2 = \cos < a, b >$$

其中 $a = \{a_1, a_2, ...a_n\}$ 为留言详情对应的文本向量， $b = \{b_1, b_2, ...b_n\}$ 为留言回复对应的文本向量 n 表示用 word2vec 训练词向量时设定的词向量维度。

6.1.4 回复是否充实详尽

留言回复内容的详细程度与回复的文本长度有直接的关系.简短内容的回复信息量一般不够,评分应该较低;同时，较长文本的回复评分不应该过高.因此,可以考虑使用对数函数来量化回复的文本长度与评分的关系,建立“回复内容是否充实”的评价项

$$F_3 = log_{100} L_i$$

其中，L_i 为针对第 i 个问题回复的文本长度。

6.1.5 回复是否及时

留言回复的及时性体现了政府的工作效率，回复时间间隔越短越好，时间间隔长则该项评分较低，因此可以考虑指数模型来量化及时回复和回复评分的关系，建立“回复是否及时的评价项”设回复时间间隔为 t_0 ，则该项指标可量化为

$$F_4 = e^{-t_0}$$

6.2 留言回复评价模型

为了给出每个回复信息的质量评分,根据回复质量评价函数得到模型的具体步骤如下:

(1)剔除无效数据;

(2)根据本文的评价指标，计算第 k 个回复的各项得分,表示为向量 $M_k=(F_1,F_2,F_3,F_4)$;

(3)设置权重向量为 $\lambda=(\lambda_1,\lambda_2,\lambda_3,\lambda_4)$,计算作为第 k 条的回复质量评价函数 $F(k)=M_k\lambda^T$。

6.3 模型的求解与评价

本文将“附录 4.xsl”中剔除 70 条无效回复后的 2746 条留言回复数据带入 $F(k)$ 模型，并把“回复是否有理有据”评价项赋予较高的权重 λ_2 =0.4，剩下 3 个指标别赋予 0.2 的权重，即

$$\lambda=(\lambda_1,\lambda_2,\lambda_3,\lambda_4)=(0.4,0.2,0.2,0.2)$$

可得

$$F(k)=M_k\lambda^T=0.4\frac{N_{lk}}{N_{0k}}+0.2\cos<a_k,b_k>+0.2log_{100}L_{ik}+0.2e^{-t_{0k}}$$。

带入数据求解后可得如下数据分布图:

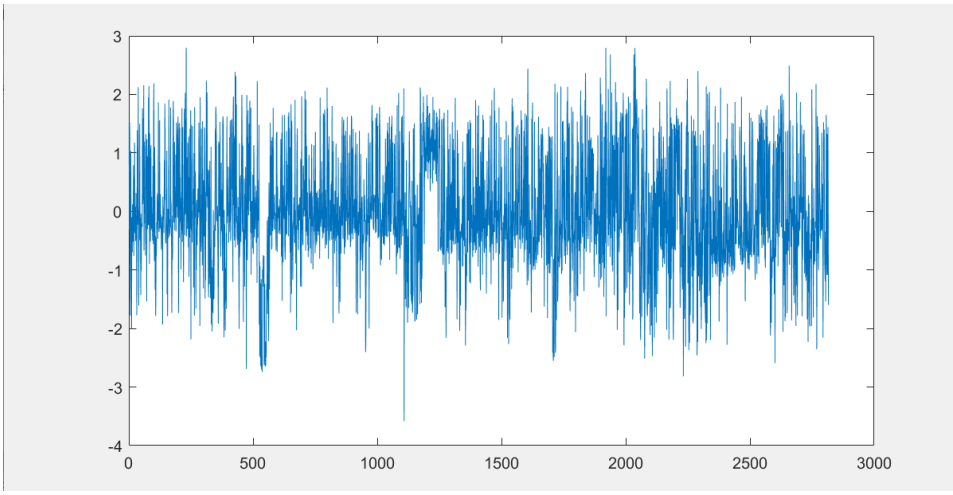

图 8.F(k)折线图

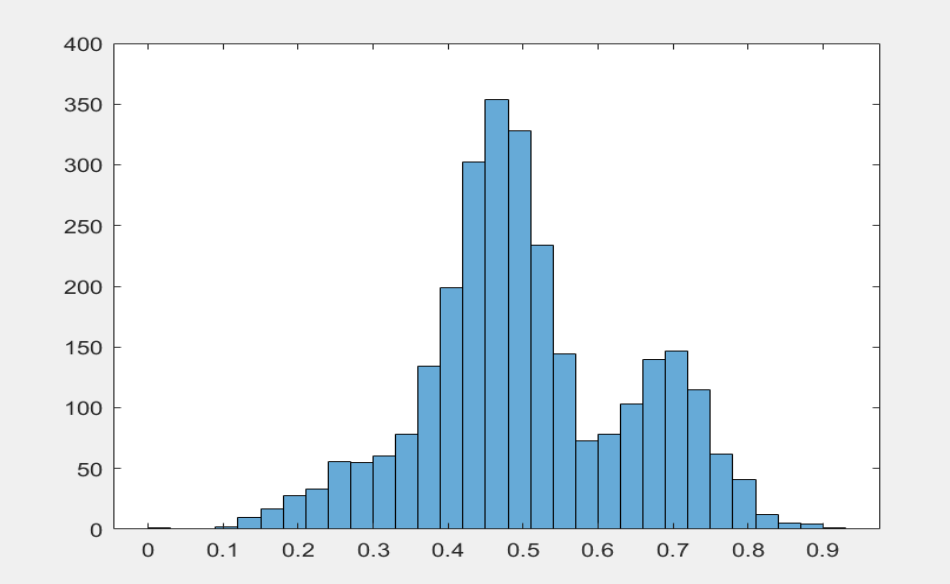

图 9.F(k)直方图

上图中，横坐标表示回复质量函数值区间，纵坐标表示回复质量的评价函数值在各个区间的数量。从图中可以看出，评分总体服从正态分布的形态，体现了本文关于主题回复评价方法的合理性。

七、总结

本文通过对“智慧政务”中的文本挖掘，建立了一套基于留言详情的文本分类模型，由测试数据可以得出其分类模型较为合理。对于文本热点问题挖掘，本文从文本相似度出发，由 word2vec 计算出文本向量，通过对文本相似度矩阵进行计算并应用谱聚类方法进行聚类并筛选出其聚类中包含文本个数的前 5 个类别作为其排名前 5 的热点问题。本文通过从 4 个方面对主题的回复质量进行量化描述,通过权重系数加权求和得到主题回复质量评价模型。结果表明,该模型可以很好地评价一个主题回复的质量。也有助于建设智能问答的网络平台，促进优质和高效的问题回复，帮助政务服务网站提供更好的用户体验。

.

八、参考文献

[1]江大鹏. 基于词向量的短文本分类方法研究[D].浙江大学,2015.
[2]贾威. 基于武汉城市留言板的舆情热点监控研究[D].华中师范大学,2019.
[3]杨开平,李明奇,覃思义.基于网络回复的律师评价方法[J].计算机科学,2018,45(09):237-242.

附录 1：

留言分类中的部分代码：

```
#encoding:utf-8
from Text_Cnn import *
from data_processing import *
from Parameters import Parameters as pm
import os
def train():
    tensorboard_dir = './tensorboard/Text_cnn'
    save_dir = './checkpoints/Text_cnn'
    if not os.path.exists(tensorboard_dir):
        os.makedirs(tensorboard_dir)
    if not os.path.exists(save_dir):
        os.makedirs(save_dir)
    save_path = os.path.join(save_dir, 'best_validation')
    tf.summary.scalar("loss", model.loss)
    tf.summary.scalar("accuracy", model.accuracy)
    merged_summary = tf.summary.merge_all()
    writer = tf.summary.FileWriter(tensorboard_dir)
    saver = tf.train.Saver()
    session = tf.Session()
    session.run(tf.global_variables_initializer())
    writer.add_graph(session.graph)

    print("Loading Training data...")
    x_train, y_train = process(pm.train_filename, wordid, cat_to_id, pm.seq_length)
    x_val, y_val = process(pm.val_filename, wordid, cat_to_id, pm.seq_length)
    for epoch in range(pm.num_epochs):
        print('Epoch:', epoch + 1)
        num_batchs = int((len(x_train) - 1) / pm.batch_size) + 1
        batch_train = batch_iter(x_train, y_train, pm.batch_size)
        for x_batch, y_batch in batch_train:
            feed_dict = model.feed_data(x_batch, y_batch, pm.keep_prob)
            _, global_step, train_summary, train_loss, train_accuracy = session.run([model.optimizer,
model.global_step,
                                                                                     merged_summary,
model.loss, model.accuracy], feed_dict=feed_dict )
            if global_step % 100 == 0:
                val_loss, val_accuracy = model.evaluate(session, x_val, y_val)
                print(global_step, train_loss, train_accuracy, val_loss, val_accuracy)

            if (global_step + 1) % num_batchs == 0:
```

```
                print("Saving model...")
                saver.save(session, save_path, global_step=global_step)

        pm.lr *= pm.lr_decay


if __name__ == '__main__':
    pm = pm
    filenames = [pm.train_filename, pm.test_filename, pm.val_filename]
    categories, cat_to_id = read_category()
    wordid = get_wordid(pm.vocab_filename)
    pm.vocab_size = len(wordid)
    pm.pre_trianing = get_word2vec(pm.vector_word_npz)
    model = TextCnn()
    train()
    # pre_label, label = val()
    #
    # correct = np.equal(pre_label, np.argmax(label, 1))
    # accuracy = np.mean(np.cast['float32'](correct))
    # print('accuracy:', accuracy)
    # print(pre_label[:10])
     # print(np.argmax(label, 1)[:10])
```

Word2vec 部分代码：

```
# coding:utf-8
# 计算全部文本的词向量
import codecs
import logging
import re
import gensim
import jieba

from gensim.models import word2vec
from gensim.models.keyedvectors import KeyedVectors
    # file_handle.write('\n')  # 有时放在循环里面需要自动转行，不然会覆盖上一条数据
# # # 设置输出日志
# # logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
# #
# # 直接用 gemsim 提供的 API 去读取 txt 文件，读取文件的 API 有 LineSentence 和 Text8Corpus,
```

PathLineSentences 等。

```
sentences = word2vec.LineSentence("/Users/take/Desktop/python/problem1/data/aa.txt")
# # #
# # # # 训练模型，词向量的长度设置为 200， 迭代次数为 8，采用 skip-gram 模型，模型保存为 bin 格式
model = gensim.models.Word2Vec(sentences, size=100, sg=1, iter=8)
model.wv.save_word2vec_format("/Users/take/Desktop/python/problem1/data/vector_word" + ".bin", binary=True)
model = KeyedVectors.load_word2vec_format('/Users/take/Desktop/python/problem1/data/vector_word.bin', binary=True)
model.save_word2vec_format('/Users/take/Desktop/python/problem1/data/vector_word.txt', binary=False)
```

求 F1 值代码（部分）：

```
#encoding:utf-8
import numpy as np
from Text_Cnn import *
from data_processing import *  # read_category, get_wordid, get_word2vec, process， batch_iter, sequence
from Parameters import Parameters as pm
from sklearn import metrics

def val():
    pre_label = []
    label = []
    session = tf.Session()
    session.run(tf.global_variables_initializer())
    save_path = tf.train.latest_checkpoint('./checkpoints/Text_cnn')
    saver = tf.train.Saver()
    saver.restore(sess=session, save_path=save_path)

    val_x, val_y = process(pm.val_filename, wordid, cat_to_id, max_length=300)
    batch_val = batch_iter(val_x, val_y, batch_size=8)
    for x_batch, y_batch in batch_val:
        pre_lab = session.run(model.predicitions, feed_dict={model.input_x: x_batch,
                                                             model.keep_pro: 1.0})
        pre_label.extend(pre_lab)
        label.extend(y_batch)
    return pre_label, label
if __name__ == '__main__':
    pm = pm
    sentences = []
    label2 = []
    categories, cat_to_id = read_category()
    wordid = get_wordid(pm.vocab_filename)
```

```
        pm.vocab_size = len(wordid)
        pm.pre_trianing = get_word2vec(pm.vector_word_npz)
        model = TextCnn()
        pre_label, label = val()
        correct = np.equal(pre_label, np.argmax(label, 1))
        accuracy = np.mean(np.cast['float32'](correct))
        print('accuracy:', accuracy)
        print(pre_label[:10])
        print(np.argmax(label, 1)[:10])
        print("Precision, Recall and F1-Score...")
print(metrics.classification_report(np.argmax(label, 1), pre_label, target_names=categories))
```

附录 2：

问题 2 部分代码：

```
# coding:utf-8
# coding:utf-8
# coding:utf-8
import re
import numpy as np
import xlrd
import openpyxl
from pandas import DataFrame

datefile = xlrd.open_workbook("/Users/take/Downloads/删除重复值.xlsx")  #  读取 Excel 文件

date = datefile.sheet_by_name("Sheet1")  #  选择 Excel 文件中的 Sheet1 表
datefile.rows = date.col_values(4, 1)  #  读取 Sheet1 文件中的第 5 列除第一行的全部文件
# date.col  对列操作
# date.roe  对行操作
wb = openpyxl.Workbook()
sheet = wb.active
# a = [35,148,538,741,1012,1171,1180,1222,1436,1525,1538,1876,1901,2053,2091]
a = [35,148,538,741,1012,1171,1180,1222,1436,1525,1538,1876,1901,2053,2091,2150,2162,2405,2660,2685,2720,2757,3016,3113,3283,3394,3402,3580,3792,3930,3962,3979,4009,4054,4140]
b = [162,252,270,617,766,803,961,962,1161,1309,1345,1346,2023,2065,2105,2107,2391,2511,2700,2827,2858,3134,3149,3176,3308,3524,3850,3892,3986,4007,4125,4166]
c = [16, 165, 183,206,238,339,750,843,847,969,1154,1448,1487,1816,1874,1959,2097,2176,2237,2372,2472,2563,2629,2947,3329,3373,3501,3602,4059,4124]
d =
```

```
[479,576,812,822,914,1050,1099,1166,1236,1265,1267,1387,1672,1754,1990,2111,2566,2635,2824,3081,3294,3386,3464,3559,3731,3988,4144]
e = [177,305,506,690,887,947,1389,1415,1649,1708,1753,1919,1937,1942,1943,2156,2181,2354,2402,2465,2481,2649,2750,3096,3852,4082,4165]
temp = date.row_values(0)
# print(temp)
sheet.cell(row=1, column=1, value='问题 ID')
for i in range(0, 7):
    sheet.cell(row=1, column=2 + i, value=temp[i])
j = 2
m = 1
for i in a:
    temp = date.row_values(i)
    sheet.cell(row=j, column=1, value=m)
    for i in range(0, 7):
        sheet.cell(row=j, column=2 + i, value=temp[i])
    j = j + 1
    # print(temp)
# j = 2
m = m + 1
for i in b:
    temp = date.row_values(i)
    sheet.cell(row=j, column=1, value=m)
    for i in range(0, 7):
        sheet.cell(row=j, column=2 + i, value=temp[i])
    j = j + 1
    # print(temp)

m = m + 1
for i in c:
    temp = date.row_values(i)
    sheet.cell(row=j, column=1, value=m)
    for i in range(0, 7):
        sheet.cell(row=j, column=2 + i, value=temp[i])
    j = j + 1
    # print(temp)

m = m + 1
for i in d:
    temp = date.row_values(i)
    sheet.cell(row=j, column=1, value=m)
    for i in range(0, 7):
        sheet.cell(row=j, column=2 + i, value=temp[i])
```

```
        j = j + 1
        # print(temp)

m = m + 1
for i in e:
        temp = date.row_values(i)
        sheet.cell(row=j, column=1, value=m)
        for i in range(0, 7):
                sheet.cell(row=j, column=2 + i, value=temp[i])
        j = j + 1
        # print(temp)
wb.save('表 2-热点问题留言明细表.xlsx')
```

附录 3：

问题 3 部分代码：

```
# coding:utf-8
import codecs
import re
from math import log
import gensim
import numpy as np
from gensim.models.keyedvectors import KeyedVectors
import jieba.analyse
import xlrd

import mypro3_time
from word_sim import word_sim

#  读取词向量
f = codecs.open('./data/temp/vector_word.txt', 'r', encoding='utf-8')
dd = []
arr = []
for ealine in f:
        item = ealine.split(' ')    #  输出全部词向量
        key = item[0]    #  输出每个词向量对应的词
        dd.append(key)
        # print(key)
        vec = np.array(item[1:], dtype='float32')    #  每个词向量矩阵
        embeddings = np.array(vec)
        arr.append(embeddings)

pro3_mod = ()    #  用元组存储留言回复的评价指标
```

```
mod_fianll = []
# print(mypro3_time.pro3_time)
#   pro3_sim 留言内容和留言回复的相似性
#   pro3_profession 回复内容是否有理有据，有则返回 1，没有则为 0
#   pro_Enrich 回复内容是否充实
datefile = xlrd.open_workbook("/Users/take/Desktop/数据挖掘/第三题/C 题全部数据/附件 4.xlsx")  # 读取
Excel 文件
text_word = open('./data/stopwords.txt', 'r', encoding='utf-8')
date = datefile.sheet_by_name("Sheet1")  # 选择 Excel 文件中的 Sheet1 表
datefile.rows = date.col_values(5, 1)  # 读取 Sheet1 文件中的第 5 列除第一行的全部文件

pro3_sim = []
pro3_profession = []
pro_Enrich = []
dat = []  # 求最大文本长度
for i in range(1, len(datefile.rows) + 1):
    # print(i)
    # o = 1
    oo1 = []
    oo2 = []
    temp_1 = []
    temp_2 = []

# if o == 1:
    # i = 1
    # 留言回复:
    temp = date.col_values(5, i, i + 1)
    datefile.result = [x.strip() for x in temp if x.strip() != '']
    # datefile.result = temp
    datefile.result = str(datefile.result)
    # result = str(temp)
    pattern = re.compile(r'xa0')  # 正则去除数据中的\xa0
    datefile.result = re.sub(pattern, '', datefile.result)
    pattern = re.compile(r'u3000')  # 正则去除数据中的\u3000
    datefile.result = re.sub(pattern, '', datefile.result)
    datefile.result = (str(datefile.result)).replace('\\', '').strip()  # 删除\
    datefile.result = (str(datefile.result)).replace('\'', '').strip()  # 删除'
    out = datefile.result.replace('[', '').replace(']', '')  # 去掉列表中的所有中括号
    # 去掉标点符号
    punc = ':  ，  !  、  。  (  “”:;  -?  .*":. )'
    datefile.result1 = re.sub(r"[%s]+" % punc, '', out)
    # 去掉文本中的留言用户编号
    out_list = re.sub(r"(?<=UU)[0-9]{7}", '', datefile.result1)
    out_list = re.sub(r"(?<=UU)[0-9]{7}", '', out_list)
```

```
        out_list = re.sub(r"(?<=UU)[0-9a-z]{6}", '', out_list)
        out_list = re.sub(r"(?<=UU)[0-9a-z]{5}", '', out_list)
        out_list = re.sub(r"(?<=A0)[0-9a-z]{8}", '', out_list)
        out_list = re.sub(r"(?<=A0)[0-9a-z]{7}", '', out_list)
        out_list = re.sub(r"(?<=A0)[0-9a-z]{6}", '', out_list)
        out_list = re.sub('UU', '', out_list)
        out_list = re.sub('A0', '', out_list)
        out_list = re.sub('网友', '', out_list)

    # 回复内容是否有理有据
        index = out_list.find('《')
        num_1 = 0
        if index != -1:
            num_1 = 1
        pro3_profession.append(num_1)
        # print(num_1)

    # 回复内容是否充实(文本长度)
        index_2 = len(out_list)
        dat.append(index_2)
        if index_2 == 0:
            num_2 = 0
        else:
            num_2 = log(index_2, 100)
        pro_Enrich.append(num_2)
        item_str = jieba.lcut(out_list)

        # print(item_str)
        # out_list = []
        # for word in item_str:
        #     if word not in stop:
        #         # if not remove_digits(word):
        #         #     continue
        #         if word != '\t':
        #             out_list.append(word)
        out_list = (str(item_str)).replace('\'', '').strip()  # 删除'
        # out_list = re.sub('UU', '', out_list)
        out_list = out_list.replace('[', '').replace(']', '')  # 去掉列表中的所有中括号
        out_list = (str(out_list)).replace(',', ' ').strip()  # 删除,
        # print(out_list)

        tages1 = jieba.analyse.extract_tags(out_list, topK=20)
        for word in tages1:
            temp_1.append(word)
```

```
 # print('TF-IDF 关键词结果:' + '、'.join(tages1))
for word in temp_1:
        if word in dd:
                if len(oo1) == 5:
                        continue
                oo1.append(word)
# print(oo1)


#  留言内容:
temp2 = date.col_values(4, i, i + 1)
datefile.result2 = [x.strip() for x in temp2 if x.strip() != '']
# datefile.result = temp
datefile.result2 = str(datefile.result2)
# result = str(temp)
pattern2 = re.compile(r'xa0')   #  正则去除数据中的\xa0
datefile.result2 = re.sub(pattern2, '', datefile.result2)
pattern2 = re.compile(r'u3000')   #  正则去除数据中的\u3000
datefile.result2 = re.sub(pattern2, '', datefile.result2)
datefile.result2 = (str(datefile.result2)).replace('\\', '').strip()   #  删除\
datefile.result2 = (str(datefile.result2)).replace('\'', '').strip()   #  删除'
out2 = datefile.result2.replace('[', '').replace(']', '')   #  去掉列表中的所有中括号
#  去掉标点符号
punc = ':  ,  !  、。  (  )  “”:;  .?  .*':. '
datefile.result2 = re.sub(r"[%s]+" % punc, '', out2)
# print(datefile.result2)

item_str2 = jieba.lcut(datefile.result2)

# print(item_str)
# out_list = []
# for word in item_str:
#         if word not in stop:
#                 # if not remove_digits(word):
#                 #         continue
#                 if word != '\t':
#                         out_list.append(word)
out_list2 = (str(item_str2)).replace('\'', '').strip()   #  删除'
# out_list = re.sub('UU', '', out_list)
out_list2 = out_list2.replace('[', '').replace(']', '')   #  去掉列表中的所有中括号
out_list2 = (str(out_list2)).replace(',', ' ').strip()   #  删除,
# print(out_list2)

tages2 = jieba.analyse.extract_tags(out_list2, topK=20)
```

```
        # 记录关键字
        for word in tages2:
            temp_2.append(word)
        for word in temp_2:
            # if len(oo2) == 5:
            #         continue
            if word in dd:
                if len(oo2) == 5:
                    continue
                oo2.append(word)
        # print(oo2)
        # print('TF-IDF 关键词结果:' + '、'.join(tages2))
        lol = word_sim(oo1, oo2, dd, arr)
        # print(lol)
        pro3_sim.append(lol)   # 留言内容和留言回复的相似性
# max_len = max(dat)
# print(max_len)
pro3_mod = (pro3_sim, pro3_profession, pro_Enrich, mypro3_time.pro3_time)
for i in range(0, len(pro3_mod[0])):
        indx1 = 0.4*pro3_mod[0][i]
        indx2 = 0.2*pro3_mod[1][i]
        indx3 = 0.2*pro3_mod[2][i]
        indx4 = 0.2*pro3_mod[3][i]
        ll = indx1+indx2+indx3+indx4
        mod_fianll.append(ll)
# print(pro3_sim)
print(mod_fianll)
```