第八届“泰迪杯”

全国数据挖掘挑战赛

作品名称：基于“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不断攀升，给以往主要靠人工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。因此，运用自然语言处理和文本挖掘技术对群众留言信息的研究具有重大意义。

对于问题 1，运用 excel 将附件 2 数据表格转化为 csv 格式，利用 python 对附件 2 留言详情进行欠抽样得到不同类别相同数量的文本信息，进而进行文本去重、jieba 分词、去除停用词得到清洗后的数据，词云图展示关键词信息，对抽取的文本所对应的标签进行数字化表示，最后，将文本数据进行向量化表示，以便计算机识别进行朴素贝叶斯算法构造分类器，建立留言分类模型，对模型进行评估。

针对问题 2，运用自然语言处理技术寻找各留言之间的相似度。即通过创建词典和语料库，LSI 通过奇异值分解的方法计算出文本中各个主题的概率分布，把各留言转化为向量矩阵，从而计算相似度，根据相似度的大小对各留言来进行归类。热度指标采用各种相似数量最多的留言即为热点问题。从而筛选出前五个热点问题。在通过命名实体技术，提取各热点问题的时间地点等，从而得到挖掘目标。

针对问题 3，对留言详情和答复意见进行相关性分析。建立相关性，完整性，可解释性的各个量化指标。对答复意见进行评价。

关键词：朴素贝叶斯分类，python，相似度算法，命名实体识别，LST

目录

一．绪论.....4
1.1 文本挖掘研究背景及意义.....4
1.2 挖掘目标.....4
二．文本分类.....5
2.1 流程图.....5
2.2 数据预处理.....5
2.2.1 文件的准备与读取.....5
2.2.2 数据欠抽样与去重、去符.....6
2.2.3 对留言详情进行中文分词与去除停用词.....6
2.2.4 词云图关键词展示.....7
2.2.5 文本向量化表示.....8
2.2.6 朴素贝叶斯算法构造分类器.....9
2.2.7 分类模型评价.....9
图 6：评价.....10
三． 热点问题挖掘.....10
3.1 流程图.....10
3.2 数据预处理.....11
3.2.1 创立词典和语料库.....11
3.2.2 计算相似度.....11
3.2.3 命名实体识别.....12
四．答复意见的评价.....14
4.1 时间角度分析.....14
4.2.1 excel 处理数据.....14
4.2.2 文本相关性分析.....15
五．参考文献.....16
六．代码.....16
6.1 问题 1 代码.....16
6.2 问题 2 代码.....20

一．绪论

1.1 文本挖掘研究背景及意义

随着计算机的高速发展，越来越多的人融入到互联网这个大家庭中来，人们逐步进入信息化时代。虽然网络上的信息形式多种多样，但绝大多数信息却都是以文本形式存在的。因此如何从大量、无结构的文本信息中快速提取出有用的关键信息变的至关重要。

1.2 挖掘目标

本次建模针对网络问政平台的群众留言数据，利用自然语言处理和文本挖掘的方法，达到以下目标：

1） 利用贝叶斯分类算法构造分类器对群众留言进行一级标签分类，对分类模型进行评价。

2） 利用相似度算法，命名实体识别，热度指数算法对某一时段特定人群、地点、问题描述进行热点问题文本挖掘。

3） 利用相关性矩阵对群众留言的答复意见进行评价。

二．文本分类

2.1 流程图

图 1：问题 1 流程图

2.2 数据预处理

2.2.1 文件的准备与读取

用 excel 将附件 2.xlsx 文件另存为以逗号为分隔符号的 csv 格式，以便用 pyhon 软件读取并进行文本分析。读取数据为 9210 条留言信息，6 个字段标题，留言一级标签为 7 类，各类别留言所占条数

如下：

```
(9210, 6)
城乡建设          2009
劳动和社会保障     1969
教育文体          1589
商贸旅游          1215
环境保护           938
卫生计生           877
交通运输           613
Name: 一级标签, dtype: int64
```

2.2.2 数据欠抽样与去重、去符

为保证各个类别所含数据数相同，以免因为类别不平衡影响模型输出，进而对数据进行欠抽样。各类别留言数据保留 600 条，总数据 4200 条。

考虑到一些群众可能会出现留言内容相同的情况，再对进行分类以无意义，进而进行文本去重工作，过滤掉重复留言内容。文本内容中存在大量的换行符、制表符、空格和其他字符等，需去掉。

2.2.3 对留言详情进行中文分词与去除停用词

在对群众留言进行分类之前，需先把非结构化的文本信息转化为计算机能够识别的结构化数据。附件 2 数据表中，是以中文文本的方式给出了数据，为了便于转化，需对群众留言详情内容进行中文分词。这里采用 python 中文分词包 jieba 进行分词。

留言文本经过去重、分词之后，并非所有的剩下词语都可以作为特征词，里面存在一些包含信息量很低甚至没有信息量的词语，如：

的、也、呢、吗，需过滤掉，否则会影响下文的分析正确率。停用词是指在信息检索中，为节省存储空间和提高搜索效率，在处理自然语言数据（或文本）之前或之后会自动过滤掉某些字或词，这些字或词即被称为 Stop Words（停用词）。

示例：

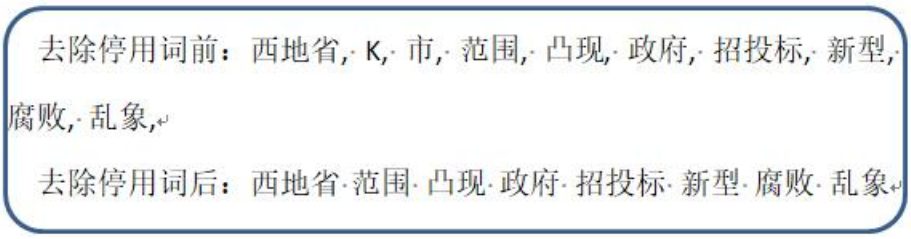


图 2：去停用词前后比较

2.2.4 词云图关键词展示

“词云”就是通过形成“关键词云层”或“关键词渲染”，对网络文本中出现频率较高的“关键词”的视觉上的突出。

代码：

```
# 绘制词云
mask=plt.imread('timg.jpg')
dic={}
for i in data_after_stop[labels==1]:
    for j in i:
        if j not in dic.keys():
            dic[j]=1
        else:
            dic[j]+=1
wc=WordCloud(mask=mask,background_color='white',font_path='simkai.ttf')
wc.fit_words(dic)
plt.imshow(wc)
plt.show()
```

图 3：代码

某次抽取中城乡建设类别词云图展示结果：

图 4：词云

从图中可以看出，开发商、业主、建设、规划、房屋、居民等这些词在城乡建设留言中出现频率极高，极大程度上可以作为文本分类时的关键特征词。

2.2.5 文本向量化表示

CountVectorizer：

只考虑词汇在文本中出现的频率

TfidfVectorizer：

除了考量某词汇在文本出现的频率，还关注包含这个词汇的所有文本的数量。能够削减高频没有意义的词汇出现带来的影响,挖掘更有意义的特征

这里我们采用的是普通统计 CountVectorizer 提取文本特征向量，然后对训练和测试文本分别进行量化处理。

2.2.6 朴素贝叶斯算法构造分类器

这里采用的是多项分布朴素贝叶斯（MultinomialNB），该方法是用于文本分类的两大经典朴素贝叶斯算法之一。其在 sklearn 中的实现函数为 sklearn.naive_bayes.MultinomialNB()。

某次抽取测试集分类结果：

```
[4 1 2 6 5 7 7 1 4 3 7 4 6 6 2 2 7 5 5 1 5 4 4 3 6 2 3 7 1 3 4 4 7 7 3 4 2
 3 4 1 7 2 5 4 6 1 4 1 4 2 7 6 3 4 2 7 6 2 2 6 6 4 1 3 6 2 4 1 2 7 3 1 7 7
 3 5 7 2 7 7 2 7 7 7 6 6 5 6 6 1 4 1 5 4 5 2 6 4 5 1 5 5 7 1 5 6 3 2 5 1 4
 7 1 1 2 7 4 1 3 3 5 6 7 6 1 3 2 5 2 6 7 2 4 7 1 6 5 7 3 6 4 3 1 3 6 1 4 2
 5 3 4 6 3 6 3 1 5 3 6 5 7 6 2 7 6 1 1 4 5 6 5 7 5 7 7 1 4 4 2 2 3 1 1 5 4
 5 6 5 2 2 6 6 7 2 2 6 5 4 6 2 5 5 5 1 6 3 7 3 4 2 1 1 6 6 1 2 7 7 6 4 6 2
 2 2 7 6 6 3 2 2 7 7 5 6 7 7 2 2 2 3 1 3 6 2 7 5 1 3 3 4 4 5 1 6 5 2 2 4 4
 3 7 3 4 2 6 2 6 5 4 5 5 7 6 7 3 3 2 2 1 6 7 4 1 4 5 3 6 6 7 4 2 4 4 6 6 4
 5 2 4 2 1 6 6 5 5 2 2 5 1 2 5 3 3 4 7 6 1 6 1 3 1 4 4 6 4 6 2 6 2 7 4 4 4
 6 4 2 7 1 5 2 3 7 3 7 2 7 1 1 2 1 6 5 1 2 7 7 6 1 7 7 2 3 7 5 3 4 7 6 4 5
 6 2 6 7 5 4 6 1 1 7 1 7 2 6 1 6 4 1 4 2 7 2 7 6 1 3 4 3 6 4 6 4 6 5 2 7 5
 6 6 3 7 3 6 3 6 2 7 2 5 4 5 6 1 7 4 4 7 1 7 2 4 5 6 7 3 2 1 5 7 7 7 7 6 6
 3 5 3 2 7 6 4 6 1 1 3 1 7 4 3 2 4 6 1 1 6 3 2 2 1 5 4 2 4 4 3 7 3 6 5 7 6
 6 6 5 6 7 3 5 6 7 2 4 6 1 1 7 7 3 5 5 7 1 2 3 5 1 3 7 1 6 7 5 5 2 1 3 6 7
 1 7 3 7 6 7 5 6 1 5 2 3 1 3 4 1 7 3 6 2 6 2 5 5 3 3 7 2 5 2 5 6 2 7 3 5 1
 1 3 3 7 3 5 7 7 3 7 2 4 1 6 3 5 6 7 2 4 7 4 5 6 6 6 4 6 6 2 6 2 5 6 6 5 4
 4 4 1 3 6 5 2 7 7 7 6 4 1 5 5 4 7 7 2 6 1 1 3 5 1 1 6 4 3 3 6 7 3 6 7 1 4
 6 7 5 3 5 7 5 3 7 3 3 5 1 4 7 2 2 5 2 5 5 4 7 6 2 2 7 4 4 5 3 5 6 4 6 1 6
 7 5 2 6 1 6 5 2 7 6 3 2 4 1 6 5 2 2 2 6 2 2 1 2 2 4 7 4 6 7 7 5 7 2 4 1 6
 6 3 7 5 1 3 2 5 1 7 6 2 3 6 3 1 5 7 3 4 7 3 1 4 7 5 1 1 1 6 3 2 5 3 6 7 4
 6 1 1 6 6 4 7 2 7 1 2 5 7 4 4 2 4 2 4 5 1 1 2 6 6 4 6 3 3 1 3 2 4 6 4 1 2
 3 4 5 6 2 5 3 5 4 3 3 2 3 4 1 3 4 5 7 2 5 3 7 1 4 4 7 1 6 1 5 5 2 4 1 4 4
 5 5 3 6 1 1 2 3 1 3 6 7 2 4 7 6]
```

图 5：分类结果

2.2.7 分类模型评价

精确率是一个二分类指标，而准确率能应用于多分类，其计算公式为：

$$准确率(accuracy) = \frac{1}{n}\sum_{i=1}^{n} I(f(x_i) = y_i)$$

通过选取样本数据的 80%作为训练数据集，20%的数据作为测试

数据集。在训练集上训练分类器，在测试集上预测对比，图 6 显示测试集上 7 个类别的平均精确率约为 0.86，召回率约为 0.86，F1 值约为 0.85，整个测试集上准确率为 0.853012048927711，效果还是不错的。

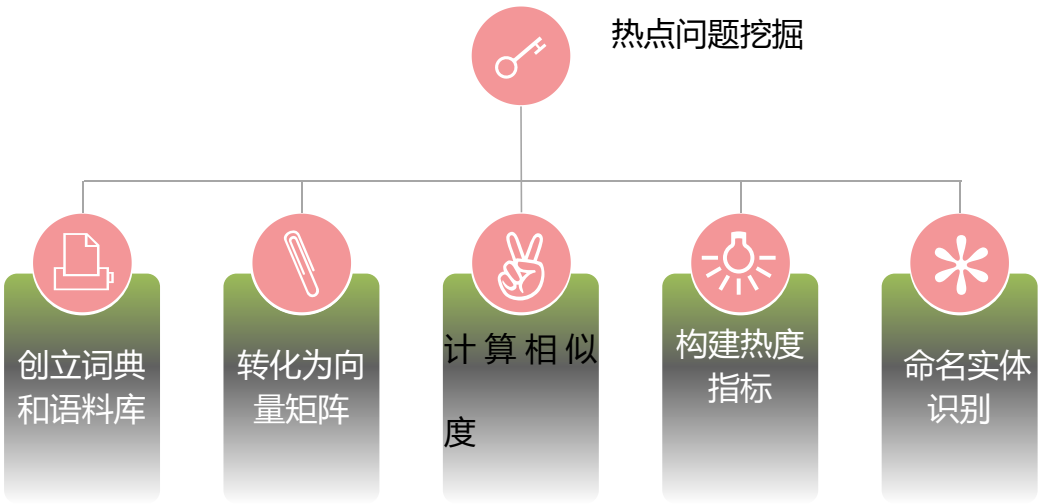
Accurancy of Naive Bayes Classfier is: 0.8530120481927711

	precision	recall	f1-score	support
1	0.77	0.74	0.75	117
2	0.85	0.91	0.88	113
3	0.95	0.84	0.89	116
4	0.89	0.83	0.86	122
5	0.87	0.74	0.80	129
6	0.84	0.99	0.91	120
7	0.81	0.94	0.87	113
accuracy			0.85	830
macro avg	0.86	0.86	0.85	830
weighted avg	0.86	0.85	0.85	830

图 6：评价

三．热点问题挖掘

3.1 流程图



3.2 数据预处理

3.2.1 创立词典和语料库

通过各留言的关键字和关键字参数创立词典。可以用 dict 函数通过映射（比如其他字典）或者（键，值）这样的序列建立字典。再继续建立语料库就是我们要分析文件的合计。

```
Prefix dict has been built succesfully.
C:/Users/有你/PycharmProjects/untitled2/day4.py:15: FutureWarning: The signature of `Series.to_csv` was aligned to that of `DataFrame.to_csv`, and
  data_cut.to_csv('data2.csv',encoding='utf-8')
0.005*"问题" + 0.005*"A市" + 0.004*"A7县" + 0.004*"小区" + 0.004*"咨询" + 0.003*"建议" + 0.003*"A2区" + 0.003*"噪音" + 0.003*"不" + 0.003*"A3区"
0.004*"A市" + 0.004*"附近" + 0.004*"小区" + 0.004*"A7县" + 0.004*"扰民" + 0.004*"A3区" + 0.004*"A2区" + 0.004*"新城" + 0.004*"严重" + 0.003*"噪音"
0.006*"魅力之城小区" + 0.005*"A市" + 0.005*"劳动" + 0.004*"滨河" + 0.004*"苑" + 0.004*"A3区" + 0.004*"车位" + 0.004*"咨询" + 0.004*"景园" + 0.004*"伊"
0.004*"经济学院" + 0.004*"A7县" + 0.003*"A市" + 0.003*"区" + 0.003*"A1" + 0.003*"夜宵摊" + 0.003*"没有" + 0.003*"咨询" + 0.003*"居民" + 0.003*"经营"
0.005*"A市" + 0.005*"A7县" + 0.005*"问题" + 0.004*"反映" + 0.004*"建设" + 0.003*"城市" + 0.003*"工程" + 0.003*"建议" + 0.003*"国家" + 0.003*"市"
```

3.2.2 计算相似度

每篇文本中有多个概率分布不同的主题；每个主题中都包含所有已知词，但是这些词在不同主题中的概率分布不同。LSI 通过奇异值分解的方法计算出文本中各个主题的概率分布。假设有 5 个主题，那么通过 LSI 模型，文本向量就可以降到 5 维，每个分量表示对应主题的权重。分词上使用了 结巴分词，词袋模型、LSI 模型的实现使用 gensim 库。

奇异值分解可以将网页文本通过向量转换后的非完全正交的多维特征投影到较少 的一个潜在语义空间中，同时保持原空间的语义特征，从而可以实现对特征空间的降噪 和降维处理。在线性代数中，奇异值分解是一类矩阵分解，是正规矩阵酉对角化的一种推广。结果

如下：

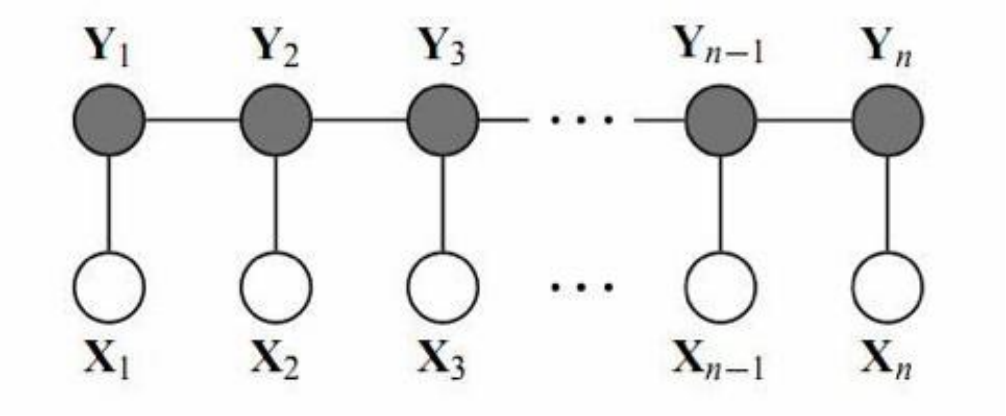
```
→ [0. ..... 0. ..... 0. ..... ... 0.86044705 1. ..... 0.5428475]
[(4324, 1.0), (4323, 0.86044705), (4322, 0.55307525), (1912, 0.54615086), (4325, 0.54
(2841, 0.4295824), (303, 0.32780302), (3141, 0.25002235), (4321, 0.24126221),
0.23685311), (3826, 0.2161834), (268, 0.21146408), (2204, 0.20089409), (983, 0.189
(2263, 0.18614058), (966, 0.1846972), (605, 0.18403766), (231, 0.18256387), (3209, 0.17
(1011, 0.1726151), (1834, 0.13964826), (1887, 0.13319175), (2608, 0.13095014)
0.1308495), (3245, 0.12957528), (4109, 0.12948246), (730, 0.12672518), (2550, 0.1239
(531, 0.12370136), (2097, 0.12135864), (1160, 0.11876798), (1452, 0.11854203)
0.11634626), (3255, 0.11479662), (409, 0.113395646), (364, 0.11209249), (287, 0.11
(1121, 0.110691145), (2157, 0.1106382), (3186, 0.110522084), (1454, 0.10492376),
0.103743576), (348, 0.103537), (2562, 0.0973533), (735, 0.097167954), (2539, 0.096
(3037, 0.0961541), (1707, 0.095533386), (2588, 0.093805365), (497, 0.0923272),
0.09195094)]
```

```
→ [0. ..... 0. ..... 0. ..... ... 0.86044705 1. ..... 0.5428475]
[(4324, 1.0), (4323, 0.86044705), (4322, 0.55307525), (1912, 0.54615086), (4325, 0.5428475),
(2841, 0.4295824), (303, 0.32780302), (3141, 0.25002235), (4321, 0.24126221), (2865,
0.23685311), (3826, 0.2161834), (268, 0.21146408), (2204, 0.20089409), (983, 0.18955854),
(2263, 0.18614058), (966, 0.1846972), (605, 0.18403766), (231, 0.18256387), (3209, 0.17700139),
(1011, 0.1726151), (1834, 0.13964826), (1887, 0.13319175), (2608, 0.13095014), (182,
0.1308495), (3245, 0.12957528), (4109, 0.12948246), (730, 0.12672518), (2550, 0.123993196),
(531, 0.12370136), (2097, 0.12135864), (1160, 0.11876798), (1452, 0.11854203), (902,
0.11634626), (3255, 0.11479662), (409, 0.113395646), (364, 0.11209249), (287, 0.1110407),
(1121, 0.110691145), (2157, 0.1106382), (3186, 0.110522084), (1454, 0.10492376), (1357,
0.103743576), (348, 0.103537), (2562, 0.0973533), (735, 0.097167954), (2539, 0.09616774),
(3037, 0.0961541), (1707, 0.095533386), (2588, 0.093805365), (497, 0.0923272), (2212,
0.09195094)]
```

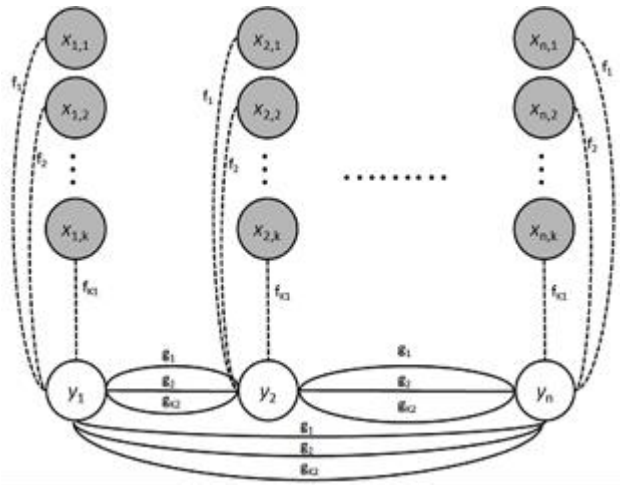
3.2.3 命名实体识别

命名实体识别（Named Entity Recognition, NER）是 NLP 的基础任务，指从文本中识别出命名性指称项，为关系抽取等任务做铺垫。狭义上，是识别出人名、地名和组织机构名这三类命名实体（时间、货币名称等构成规律明显的实体类型可以用正则等方式识别）。当然，在特定领域中，会相应地定义领域内的各种实体类型。通过热度指标我们筛选出了前五个热点问题。然后提取命名性指标项。

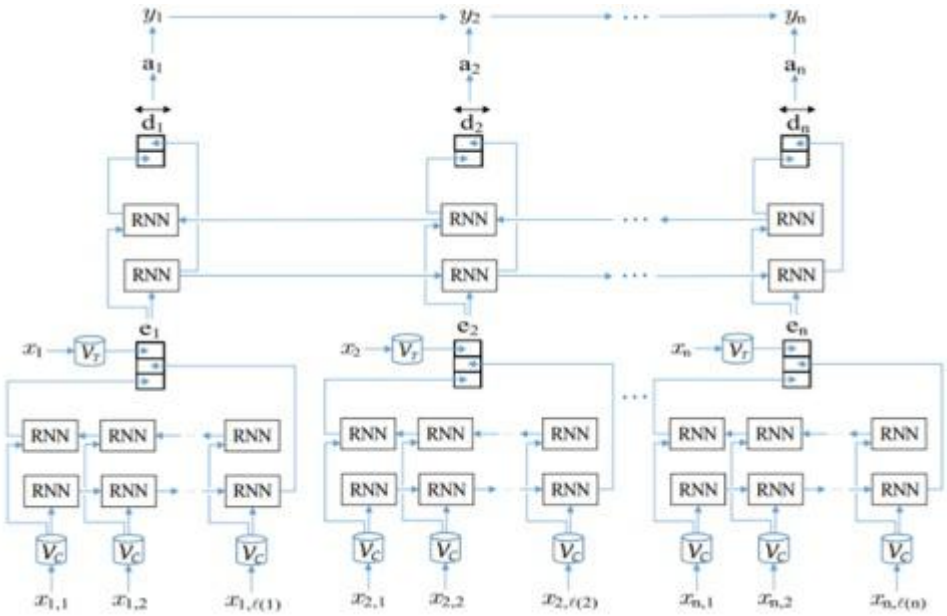
我们利用的处理算法是条件随机场（CRF），它是一种判别式概率模型，是随机场的一种，常用于标注或分析序列资料，如自然语言文字或是生物序列。简单是说在 NER 中应用是，给定一系列的特征去预测每个词的标签。如下图：



X 我们可以看做成一句话的每个单词对应的特征，Y 可以看做成单词对应的标签。这里的标签就是对应场景下的人名、地名等等。但是这些特征需要我们根据不同的场景去人工的抽取，比如抽取人名的特征我们往往可能看看单词的第一个字是不是百家姓等等。所以更多严谨的 CRF 的图应该如下：



然后利用 word2vec 就是将相似的词的“距离”拉的很近这样可以一定程度上减少未出现词的影响。如下:



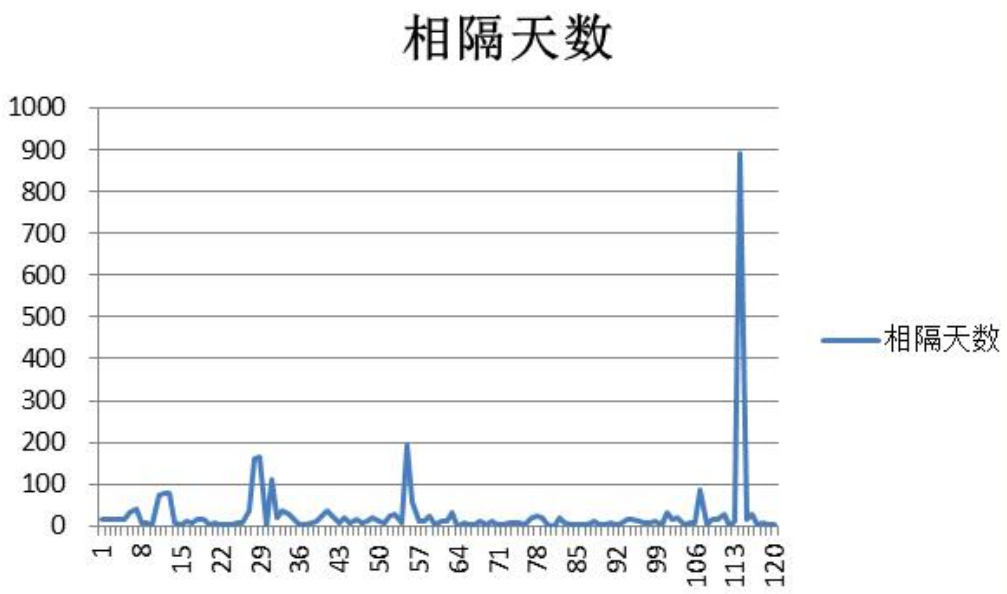
四．答复意见的评价

4.1 时间角度分析

4.2.1 excel 处理数据

运用 excel 计算出答复时间和留言时间之间相隔天数，结果保留整数格式。

绘制折线图如下：



从计算结果可以看出，相关部门对留言答复得时间多在 15 天左右，答复是比较快捷的。

4.2.2 文本相关性分析

对留言详情和答复意见进行相关性分析，对两组数据分别进行数据处理：

第一步：对数据使用 DataFrame 化，并进行数组化。

第二步：对数据进行分词，并去除停用词，使用’’.join 连接列表。

第三步：对文本数据进行 jieba 分词和停用词的去除。

第四步：使用 TF-idf 词袋模型，对特征进行向量化。

第五步：使用 from sklearn.metrics.pairwise import cosine_similarity， 对两两样本之间做相关性矩阵，使用的是余弦相似度计算公式。

五．参考文献

[1] Python 机器学习实战—使用朴素贝叶斯进行垃圾短信识别

[2] 王大阳.Python 文本数据分析：新闻分类任务（贝叶斯，TF-IDF 词向量），2019

[3]张振亚，王进，程红梅,等. 基于余弦相似度的文本空间索引方法研究[J]. 计算 机科学，2005，32(9):160-163.

[4]黄章益，刘怀亮. 一种基于语义的中文文本特征降维技术研究[J]. 情报杂志，2011(S2):123-125.

六．代码

6.1 问题 1 代码

```
import pandas as pd
import re
import jieba
data = pd.read_csv('附件 2.csv',encoding='GBK')
y=data['一级标签'].value_counts()
#print(y)
```

```
#欠抽样
n=600
a=data[data['一级标签']=='城乡建设'].sample(n)
b=data[data['一级标签'] == '教育文体'].sample(n)
c=data[data['一级标签'] == '卫生计生'].sample(n)
d=data[data['一级标签'] == '交通运输'].sample(n)
e=data[data['一级标签'] == '商贸旅游'].sample(n)
f=data[data['一级标签'] == '环境保护'].sample(n)
g=data[data['一级标签'] == '劳动和社会保障'].sample(n)
data_new=pd.concat([a,b,c,d,e,f,g],axis=0)
data_new.to_csv('data0.txt',encoding='utf-8')
print(data_new)
#  文本去重
data1=data_new['留言详情'].drop_duplicates()
# print(data_dup)
data2=data1.apply(lambda x:re.sub('\t','',x)).apply(lambda x:
re.sub('\n','',x)).apply(lambda x: re.sub('\r','',x)).apply(lambda x:
re.sub('  ','',x)).apply(lambda x: re.sub('  ','',x))
data3=data2.apply(lambda x: re.sub('      ','',x)).apply(lambda x:
re.sub('    ','',x)).apply(lambda x: re.sub('      ','',x)).apply(lambda x:
re.sub('                                          ','',x))
data4=data3.apply(lambda x: re.sub('  ，  ','',x)).apply(lambda x:
```

```
re.sub(' :','',x)).apply(lambda x: re.sub(', ','',x)).apply(lambda x: re.sub('
','',x)).apply(lambda x: re.sub('','',x))
data_cut=data4.apply(lambda x: jieba.lcut(x))
stopwords=pd.read_csv('1.txt',header=None)
stopwords=['{','}']+list(stopwords.iloc[:,0])
# print(stopwords)
data_after_stop=data_cut.apply(lambda x:[i for i in x if i not in
stopwords])
# data_after_stop.to_csv('data1.csv',encoding='utf-8')
# print(data_after_stop)
# print(data_new.一级分类.unique())
label_mapping={'城乡建设':1,'教育文体':2,'卫生计生':3,'交通运输':4,'
商贸旅游':5,'环境保护':6,'劳动和社会保障':7}
data_new['一级标签']=data_new['一级标签'].map(label_mapping)
# label=data_new['一级标签']
# print(label)
labels=data_new.loc[data_after_stop.index,'一级标签']
# print(labels)
adata=data_after_stop.apply(lambda x:' '.join(x))
# print(labels)
# print(adata)
```

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
# 绘制词云
mask=plt.imread('timg.jpg')
dic={}
for i in data_after_stop[labels==1]:
    for j in i:
        if j not in dic.keys():
            dic[j]=1
        else:
            dic[j]+=1
wc=WordCloud(mask=mask,background_color='white',font_path='simkai.
ttf')
wc.fit_words(dic)
plt.imshow(wc)
plt.show()
#模型分类器构建
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report#导入评价性能评估模
块
```

```
# 模型构建
data_train,data_test,labels_train,labels_test=train_test_split(adata,label
s,test_size=0.2)
# 朴素贝叶斯
vec=CountVectorizer()
data_train=vec.fit_transform(data_train)
data_test=vec.transform(data_test)
#模型训练
mnb=MultinomialNB()
mnb.fit(data_train,labels_train)
y_predict=mnb.predict(data_test)
#打印准确性,平均精确率，召回率，F1 指标
print('Accurancy of Naive Bayes Classfier
is:',mnb.score(data_test,labels_test))
print (classification_report(labels_test,y_predict))
```

6.2 问题 2 代码

```
# 创建词典
# dictionary = corpora.Dictionary(data1)
# # print(dictionary)
# #获取语料库
```

```
# corpus = [dictionary.doc2bow(i) for i in data1]

# tfidf = models.TfidfModel(corpus)

# # print(corpus)

# #特征数

# featureNUM = len(dictionary.token2id.keys())

# # print(featureNUM)

# #通过 TfIdf 对整个语料库进行转换并将其编入索引，以准备相似性查询

# index = similarities.SparseMatrixSimilarity(tfidf[corpus],num_features=featureNUM)

# #稀疏向量.dictionary.doc2bow(doc)是把文档 doc 变成一个稀疏向量，[(0, 1), (1, 1)]，表明 id 为 0,1 的词汇出现了 1 次，至于其他词汇，没有出现。

# string=data1[4324]

# new_vec = dictionary.doc2bow(string)

# sim = index[tfidf[new_vec]]
```

print(new_vec)

#计算向量相似度

a=sorted(enumerate(sim), key=lambda item: -item[1])

print(a)

命名实体识别：

```
# -*- coding: utf-8 -*-
import sys
reload(sys)
sys.setdefaultencoding('utf8')    #让 cmd 识别正确的编码
import nltk
newfile = open('news.txt')
text = newfile.read()  #读取文件
tokens = nltk.word_tokenize(text)  #分词
tagged = nltk.pos_tag(tokens)  #词性标注
entities = nltk.chunk.ne_chunk(tagged)  #命名实体识别
a1=str(entities) #将文件转换为字符串
file_object = open('out.txt', 'w')
file_object.write(a1)   #写入到文件中
file_object.close( )
print entities
```