基于自然语言处理技术的智慧政务系统

摘要

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，对分类数据采集并做出分析处理是必要且有效的手段。

针对任务一：对附件一和附件二中的数据进行简单的分析，对数据进行预处理，并从处理后的数据计算“留言详情”的 TF-IDF 特征值，提取文本特征，使用的是 sklearn 的朴素贝叶斯分类器 MultinomialN 构建分类模型，采取下列四种模型包括 Logistic Regression(逻辑回归)、(Multinomial) Naive Bayes(多项式朴素贝叶斯)、Linear Support Vector Machine(线性支持向量机)、Random Forest(随机森林)评估它们的准确率，选取准确率最高的模型对数据训练，并通过计算 F1 值，评估模型。

针对问题 2，我们首先充分分析附件 3 数据的基础上，对从众多留言中识别出相似的留言进行问题识别，其次，通过过滤一些无意义的词，特殊符号的处理等，对识别的结果进行进行归类，即把相似的留言归为同一问题，最后进行热度评价，先对热度评价指标进行定义，量化评价标准，后进行相似度的计算，计算完后对这个指标进行排序并保存到热点问题留言明细表中。

针对问题 3 答复意见的评价，附件四中有相关部门对留言的答复意见，无论答复意见有没有用，有没有被实现，我们都需要对所有答复内容进行评价，所以需要从答复的相关性、完整性，可解释性等角度对答复意见质量做出一套评价方案.

第一章 问题描述

1.1 问题的描述

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。而我们的任务从采集到的数据建立一个留言分类模型，并对某一时间段内反映的特定地点或特定群问题的留言进行分类，定义合理的热度评价指标，提炼出群众反映的热点问题，并针对政府的回复定义评价指标，反馈给政府。

首先，我们需要针对附件所给的数据，进行数据预处理，包括数据清洗、分词、去停用词、文本特征提取、文本表示、建立相应的多分类模型，并通过计算 F1 值调整模型；之后，我们需要对留言详情中的特定地点和特定人群进行词频统计，进行数据归并，通过热度评价指标筛选出热点问题；最后，通过对大幅意见和问题的匹配度进行分析，对答复意见进行评价。

1.2 论文结构的安排

本文共分为四章，各章内容安排如下：

第一章， 对论文需要解决的问题进行描述，并简单的介绍整篇论文的结构安排。
第二章， 模型假设
第三章， 建立对群众留言内容的一级标签分类模型，对附件一和附件二中的数据进行简单的分析，对数据进行预处理，并从处理后的数据提取文本特征，并构建多分类模型，通过计算 F1 值，评估模型。
第四章，热点问题的挖掘，对附件三中得数据进行数据预处理，对处理后的数据进行文本相似度的计算，提取文本中出现得地点和人群的词汇，并进行词频统计，定义评价指标，对留言主题进行热度排行。
第五章，答复意见的评价，通过对答复意见和留言主题的匹配度，评价答复意见。

第二章 模型假设

为了便于问题的研究，对题目中某些条件进行简化及合理假设。
1.假设所有数据真实可靠。
2.假设网络读取留言数据正常，与人们提交的留言问题一致。

第三章 群众留言分类

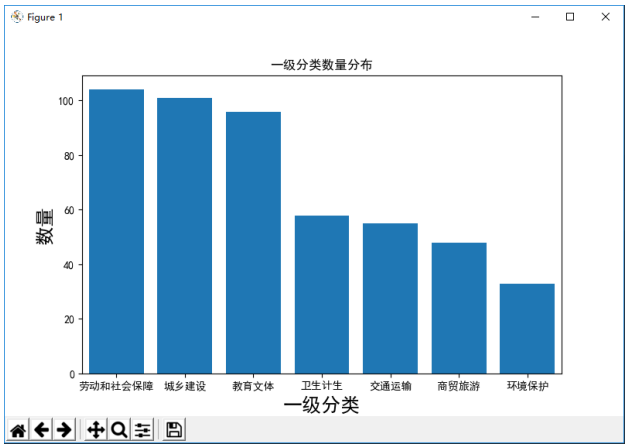
在构建分类模型时，文本特征的提取是必要的，文本特征的提取首先需要对数据进行预处理，包括数据清洗、分词、去停用词，本章就上述几个方面的问题，对附件二的数据进行初步的探索分析。

3.1 数据分析

在数据挖掘中，简单的查看数据、分析数据，可以帮助我们更好的了解数据的分布情况，思考要解决的问题，在后期的建模中着重解决数据带来的影响。数据中包含了 7 个类别，分别是劳动与社会保障、城乡建设、教育文体、卫生计生、交通运输、商贸旅游、环境保护，示例数据中附件二共有 496 条留言数据 首先查看一下我们的数据，我们想要把不同数据分到不同的分类中去,且每条数据只能对应附件一分类体 2 系中的一个类。
对比附件一给出的分类体系中，共有 15 个类，数据中缺少了党务政务、国土资源、纪检监察、经济管理、科技与信息产业、民政、农村农业、政法八个类别的数据，可见附件二不包含于附件一中全部类别的数据，这是需要解决的问题之一。
对附件二中的数据，统计各个类别的数据量，我们看到各个类别的数据量不一致，劳动与社会保障、城乡建设的数据量最多，各有 100 多条；教育文体的数据量有 96 条，与前两项数据量接近；而卫生计生、交通运输、商贸旅游的数据量，相对前面三个类别的数据量只有一半左右；环境保护的数据量最少只有 33 条，只有最高数据量的三分之一，可见数据分布很不均匀，用图形化的方式查看各个类别的分布更直观的看到数据的一个分布情况，因此在建模的过程中，需要考虑数据不平衡带来的对模型训练的影响。

```
      一级分类  count
0  劳动和社会保障    104
1     城乡建设    101
2     教育文体     96
3     卫生计生     58
4     交通运输     55
5     商贸旅游     48
6     环境保护     33

Process finished with exit code 0
```

图一：附件二中的数据各个类别的数据量分布



图二：附件二中的数据各个类别的数据量分布条形图

除此之外，在提取文本特征的时候也需考虑到文本语义带来的词语交叉对分类错误率的增加。

3.2 数据预处理

在数据挖掘过程中，数据处理是第一步，同时也是很重要的一步，数据处理的好坏决定着之后特征提取、分类预测等步骤能否顺利进行。在对数据进行分析之前和分析的过程中，我们逐渐对题目所给的数据的认识逐步加深，并最终得出了一套较为完整的数据预处理流程。

3.21 数据清洗（特殊字符的处理）

由于我们的留言内容都是中文，文本中存在大量的的标点符号，特殊符号，因为这些词和符号对系统分析预测文本的内容没有任何帮助，反而会增加计算的复杂度和增加系统开销，所有在使用这些文本数据之前必须要将它们清理干净。首先定义几个清洗文本的函数：

```python
# 定义删除除字母,数字，汉字以外的所有符号的函数
def remove_punctuation(line):
    line = str(line)
    if line.strip() == '':
        return ''
    rule = re.compile(u"[^a-zA-Z0-9\u4E00-\u9FA5]")
    line = rule.sub('', line)
    return line
```

图三：清洗文本的函数图

3.22 去停用词

中文停用词包含了很多日常使用频率很高的常用词,如 吧，吗，呢，啥等一些感叹词等,这些高频常用词无法反应出文本的主要意思,因此要被过滤掉。

```
# 停用词列表
def stopwordslist(filepath):
    stopwords = [line.strip() for line in open(filepath, 'r', encoding='ISO-8859-1').readlines()]
    return stopwords


# 加载停用词
stopwords = stopwordslist("E:\智慧政务中的文本挖掘应用/stopword.txt")
```

图四：去停用词代码图

3.23 分词

我们过滤掉了留言详情中的标点符号和一些特殊符号,并生成了一个新的字段 clean_review。接下来我们要在 clean_review 的基础上进行分词,把每个评论内容分成由空格隔开的一个一个单独的词语。

```
# 分词，并过滤停用词
data_after_stop = data['cut_review'] = data['clean_review'].apply(lambda x: " ".join([w for w in list(jb.cut(x)) if w not in stopwords]))
#data['cut_review'] = data['clean_review'].apply(lambda x: [w for w in list(jb.cut(x)) if w not in stopwords])
data.head()
print(data)
```

图五：分词代码图

3.24 将一级分类中的类标签转换成 id

将一级分类中的类标签转换成 id，这样便于以后的分类模型的训练。

```
data['一级分类_id'] = data['一级分类'].factorize()[0]
cat_id_data = data[['一级分类', '一级分类_id']].drop_duplicates().sort_values('一级分类_id').reset_index(drop=True)
cat_to_id = dict(cat_id_data.values)
id_to_cat = dict(cat_id_data[['一级分类_id', '一级分类']].values)
data.sample(10)
print(cat_id_data)
```

图六：一级分类中的类标签转换成 id 代码图

	一级分类	一级分类_id
0	NaN	-1
1	城乡建设	0
2	环境保护	1
3	交通运输	2
4	教育文体	3
5	劳动和社会保障	4
6	商贸旅游	5
7	卫生计生	6

图七：一级分类中的类标签转换成 id 结果图

数据预处理之后的结果如下：

	留言详情	...	cut_review
0	\n\t\t\t\t\t\n\t\t\t\t\tA3区大道西行便道，未管所路口至加油站路段，...	...	A3 区 大道 西行 便道 未管 所 路口 至 加油站 路段 人行道 包括 路灯 杆 被 圈...
1	\n\t\t\t\t\t\n\t\t\t\t\t位于书院路主干道的在水一方大厦一楼至四楼人为...	...	位于 书院 路 主干道 的 在水一方 大厦 一楼 至 四楼 人为 拆除 水电 等 设施 后 ...
2	\n\t\t\t\t\t\n\t\t\t\t\tA市政府、市交警支队、市安监局、市环保局、A...	...	市政府 市 交警支队 市 安监局 市 环保局 A3 区政府 我们 是 市 A3 区 杜鹃 文...
3	\n\t\t\t\t\t\n\t\t\t\t\t胡书记，您好，感谢您百忙之中查看这份留言。我...	...	胡书记 您好 感谢您 百忙之中 查看 这份 留言 我 的 父亲 51 在 A6 区 金星 北...
4	\n\t\t\t\t\t\n\t\t\t\t\t K8县丁字街的商户乱摆摊，前段时间丁字街...	...	K8 县 丁字街 的 商户 乱 摆摊 前段时间 丁字街 的 交通 好 了 几天 最近 那些 ...
...	...	...	...
65197	NaN	...	nan
65198	NaN	...	nan
65199	NaN	...	nan
65200	NaN	...	nan
65201	NaN	...	nan

[65202 rows x 4 columns]

Process finished with exit code 0

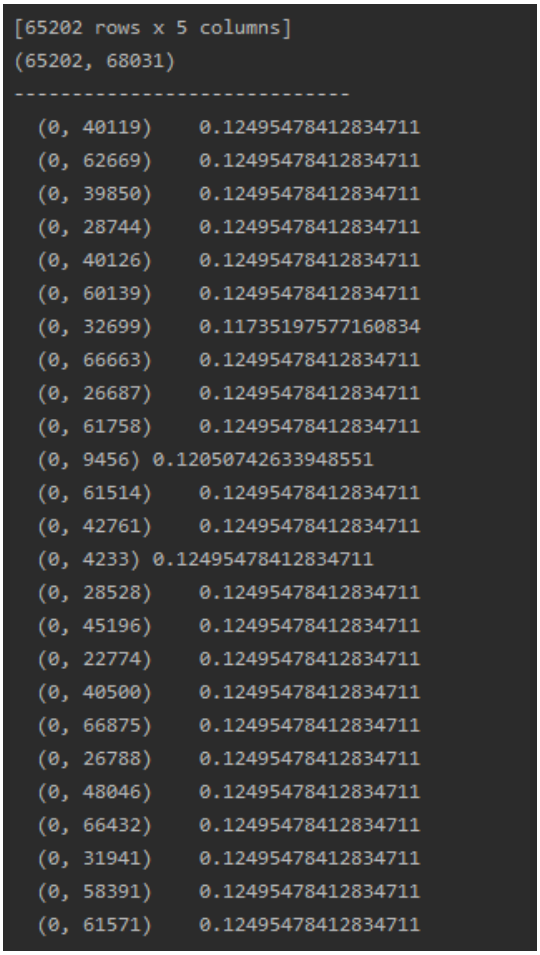
图八：数据预处理之后的结果图

3.3 文本特征的提取

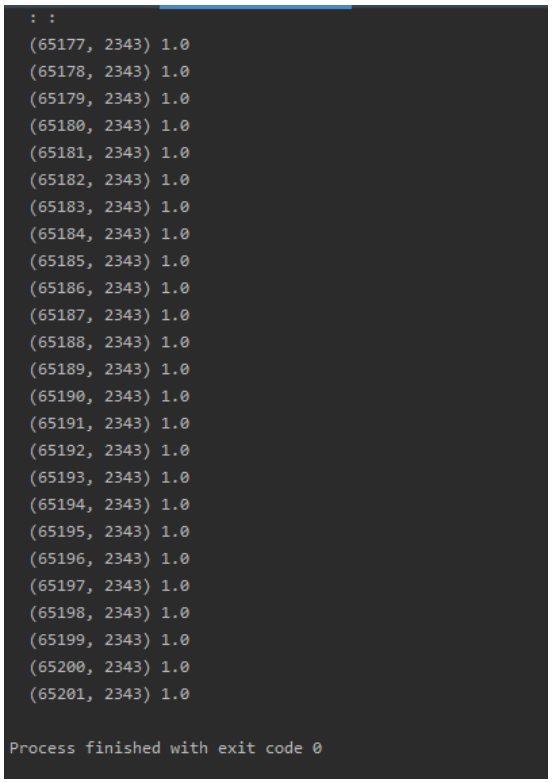
TF-IDF 是在单词计数的基础上，降低了常用高频词的权重,增加罕见词的权重。因为罕见词更能表达文章的主题思想，因此选取此方法计 cut_review 的 TF-IDF 的特征值。使用 sklearn.feature_extraction.text.TfidfVectorizer 方法来抽取文本的 TF-IDF 的特征值。这里我们使用了参数 ngram_range=(1,2),这表示我们除了抽取评论中的每个词语外,还要抽取每个词相邻的词并组成一个“词语对”,如：词 1，词 2，词 3，词 4，(词 1，词 2)，(词 2,词 3)，(词 3，词 4)。这样就扩展了我们特征集的数量,有了丰富的特征集才有可能提高我们分类文本的准确度。

```
#计算cut_review的 TF-IDF的特征值
tfidf = TfidfVectorizer(norm='l2', ngram_range=(1, 2))
features = tfidf.fit_transform(data.cut_review)
labels = data.一级分类_id
print(features.shape)
print('----------------------------')
print(features)
```

图九：计算分词后的留言详情 TF-IDF 特征值代码图



图十：计算分词后的留言详情 TF-IDF 特征值部分结果图



图十一：计算分词后的留言详情 TF-IDF 特征值部分结果图

3.4 分类器的选择

为了训练监督学习的分类器，我们首先将“留言详情”转变为包含数字的词向量。例如我们前面已经转换好的 tf-idf 的 features。这里选取朴素贝叶斯分类器。

朴素贝叶斯分类器最适合用于基于词频的高维数据分类器，最典型的应用如垃圾邮件分类器等，准确率可以高达 95%以上。这里我们使用的是 sklearn 的朴素贝叶斯分类器 MultinomialNB，我们首先将 review 转换成词频向量，然后将词频向量再转换成 TF-IDF 向量，还有一种简化的方式是直接使用 TfidfVectorizer 来生成 TF-IDF 向量(正如前面生成 features 的过程)，按照一般的方式将生成 TF-IDF 向量分成两个步骤：

①生成词频向量

②生成 TF-IDF 向量

③训练 MultinomialNB 分类器。

```
#创建特征集
X_train, X_test, y_train, y_test = train_test_split(data['cut_review'], data['一级分类_id'], random_state=0)
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)

tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)

clf = MultinomialNB().fit(X_train_tfidf, y_train)
```

图十二：创建特征集代码图

3.5 模型的选择

对模型训练完毕之后，采取下列四种模型评估它们的准确率

Logistic Regression(逻辑回归)

(Multinomial) Naive Bayes(多项式朴素贝叶斯)

Linear Support Vector Machine(线性支持向量机)

Random Forest(随机森林)

```
#模型选择
models = [
    RandomForestClassifier(n_estimators=200, max_depth=3, random_state=0),
    LinearSVC(),
    MultinomialNB(),
    LogisticRegression(random_state=0),
]
CV = 5
cv_data = pd.DataFrame(index=range(CV * len(models)))
entries = []
for model in models:
    model_name = model.__class__.__name__
    accuracies = cross_val_score(model, features, labels, scoring='accuracy', cv=CV)
    for fold_idx, accuracy in enumerate(accuracies):
        entries.append((model_name, fold_idx, accuracy))
cv_data = pd.DataFrame(entries, columns=['model_name', 'fold_idx', 'accuracy'])

sns.boxplot(x='model_name', y='accuracy', data=cv_data)
sns.stripplot(x='model_name', y='accuracy', data=cv_data,
              size=8, jitter=True, edgecolor="gray", linewidth=2)
plt.xticks(rotation=360)
plt.show()
print(cv_data.groupby('model_name').accuracy.mean())
```

图十三：评估四种模型准确率的代码图

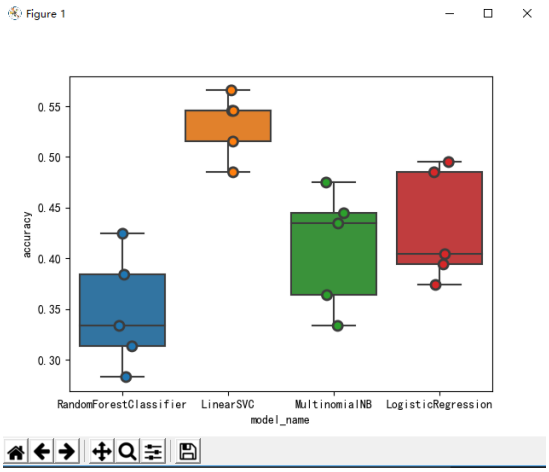
从打印的结果可以看出四个模型的准确率均比较低，线性支持向量机的准确最高，其余三个分类器的平均准确率都在 50%以下，随机森林分类器的准确率最低，只有 34%，这是因为随机森林属于集成分类器，一般来说集成分类器不适合处理高维数据，如文本数据，因为文本数据有太多的特征值，使得集成分类器难以

应付。

从箱体图上可以更直观的看出四种分类器的准确率之间的差别。

```
[495 rows x 5 columns]
model_name
LinearSVC                0.531313
LogisticRegression       0.430303
MultinomialNB            0.410101
RandomForestClassifier   0.347475
Name: accuracy, dtype: float64
```

图十四：四种模型准确率的结果图



图十五：四种模型准确率的箱体图

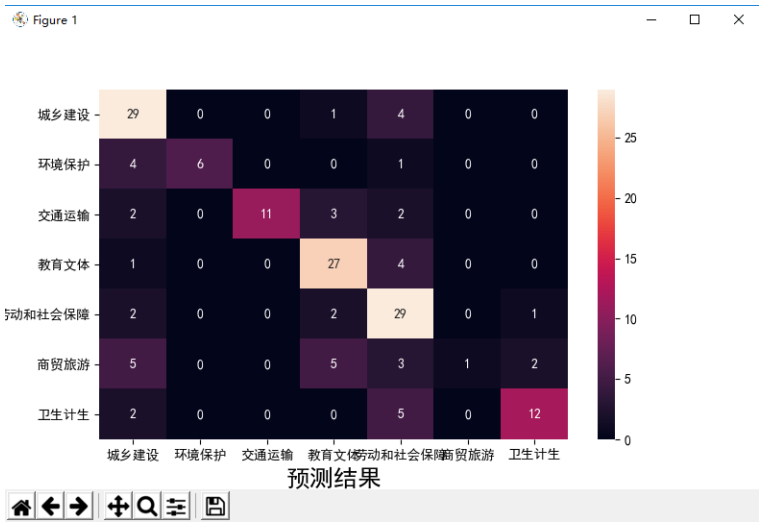
3.6 模型的评估

下面我们就针对从上一步得出的平均准确率最高的 LinearSVC 模型，查看混淆矩阵，并显示预测标签和实际标签之间的差异。

```
# 训练模型
model = LinearSVC()
X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(features, labels, data.index,
                                                                                 test_size=0.33, stratify=labels,
                                                                                 random_state=0)

model.fit(X_train, y_train)
y_pred = model.predict(X_test)



#生成混淆矩阵
conf_mat = confusion_matrix(y_test, y_pred)
fig, ax = plt.subplots(figsize=(10, 8))
sns.heatmap(conf_mat, annot=True, fmt='d',
            xticklabels=cat_id_data.一级分类.values, yticklabels=cat_id_data.一级分类.values)
plt.rcParams['font.sans-serif'] = 'SimHei'
plt.ylabel('实际结果', fontsize=18)
plt.xlabel('预测结果', fontsize=18)
pl.xticks(rotation=360)
plt.show()
```

图十六：LinearSVC 模型训练、混淆矩阵代码图

混淆矩阵的主对角线表示预测正确的数量,除主对角线外其余都是预测错误的数量，从下面的混淆矩阵可以看出“交通运输”、“环境保护”、“商贸旅游”类预测最准确,没有预测错误。“城乡建设”和“劳动和社会保障”预测的错误数量教多。



图十七：混淆矩阵图

3.7 F1 值评估

多分类模型一般不使用准确率(accuracy)来评估模型的质量,因为 accuracy 不能反应出每一个分类的准确性,当训练数据不平衡，准确率不能反映出模型的实际预测精度,这时候我们就需要借助于 F1 分数来评估模型。

下面我们将查看各个类的 F1 分数。

```
print('accuracy %s' % accuracy_score(y_pred, y_test))
print(classification_report(y_test, y_pred, target_names=cat_id_data['一级分类'].values))
#print('Testing accuracy %s' % accuracy_score(y_test, y_pred))
print('Testing F1 score: {}'.format(f1_score(y_test, y_pred, average='weighted')))
```

图十八：计算各类标签及模型 F1 值的代码图

从以下 F1 分数上看，“教育文体”类的 F1 分数最大,(从混淆矩阵也可看出没有预测错误)，“商贸旅游”类 F1 分数最差只有 12%，究其原因可能是因为“商贸旅游”分类的训练数据最少,使得模型学习的不够充分,导致预测失误较多,其余类的值均在 70%以上，较为接近。总体的 F1 分数为 67%，准确率在 70%，总体效果较差。模型还有待优化。

```
accuracy 0.7012195121951219
              precision    recall  f1-score   support

       城乡建设       0.64      0.85      0.73        34
       环境保护       1.00      0.55      0.71        11
       交通运输       1.00      0.61      0.76        18
       教育文体       0.71      0.84      0.77        32
     劳动和社会保障       0.60      0.85      0.71        34
       商贸旅游       1.00      0.06      0.12        16
       卫生计生       0.80      0.63      0.71        19

    accuracy                           0.70       164
   macro avg       0.82      0.63      0.64       164
weighted avg       0.76      0.70      0.67       164

Testing F1 score: 0.6732348538608519
```

图十九：各类标签及模型的 F1 值结果图

第四章　热便问题挖掘

定义点赞的数目，涉及人群的范围，出现的次数，关注的人数，关注度，反应的人数等相关的量化评价标准，及相应的计算方法，确定热门问题评价相关指标。利用数学方法选取最具分类信息的特征，使用 word2vec 工具，对文本相似度进行计算，得出评价结果，按表 1 的格式给出排名前 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”。

4.1 问题识别和问题归类

对所有问题的留言的具体内容进行识别，并提取关键字词。从而从每句问题留言中提取关键词，其次，通过解决问题中存在的地名和时间有简写的问题，过滤一些无意义的词，特殊符号的处理等，对识别的结果进归类，把特定地点或人群的数据归并，即把相似的留言归为同一问题，结果对应表 2 即把相似的留言归为同一问题。

4.2 热度评价

基于字面匹配其中字面距离是衡量的标准，字面距离：字符串有字符构成，只要比较两个字符串中每一个字符是否相等便知道两个字符串是否相等，或者更简单一点将每一个字符串通过哈希函数映射为一个哈希值，然后进行比较。用到的是 TF-IDF 的方法。

一个词的权重由 TF*IDF 表示，其中 TF 表示词频，及一个词在这篇文本中出现的频率；IDF 表示逆文的频率，即一个词在所有文本中出现的频率的倒数。其中他们的计算公式分别是：$TF=\frac{\text{某个词在文章中出现次数}}{\text{文章的总词数}}$；$IDF=\log\left(\frac{\text{语料库的文档总数}}{\text{包含该次的文档数}+1}\right)$.

例如提取两句文本，首先进行分词，统计所有词组将 S1 和 S2 中出现的所有不同词组融合起来，并得到一个词向量超集，然后获取 TF 词频，并乘以 IDF 权重，分别得到 S1,S2 这两句话的 TF*IDF 表示。$Similarity=\cos(\ominus)=\frac{AXB}{\|A\|\|B\|}=$

$\frac{\sum_{i=1}^{n}AiXBi}{\sqrt{\sum_{i=1}^{n}(Ai)^2}\times\sqrt{\sum_{i=1}^{n}(Bi)^2}}$根据此公式计算得到文本相似度。得到相似度后就可知道那条是热点问题，从而排列出来。

第五章　答复意见的评价

答复的相关性是指答复意见的内容是否与问题相关，完整性是指是否满足某种规范，可解释性是指答复意见中的相关解释。那我们如何将相关性、完整性、可解释性等描述量化？构建什么指标来计算和评价呢？这是问题中存在的难点。