**实 验 报 告**

|  |  |
| --- | --- |
| **课程名称：** | **数据挖掘** |
| **学生姓名：** |  |
| **学生学号：** |  |
| **学生专业：** | **软件工程** |
| **开课学期：** | **2019-2020学年第一学期** |

**软件学院**

**2019年11月**

**一、实验概述**

|  |  |
| --- | --- |
| 题目 | O2O商铺食品安全相关评论发现 |
| 内容 | 对餐饮评论有无安全问题进行打标签 |
| 提交次数 | 25 |
| 最终成绩 | 0.94258374 |
| 最终排名 | 26 |

**二、数据处理**

1.1 数据概览

本次数据由train.csv 和test.csv两部分组成，train.csv共由1万条数据，其中数据已经标有类别，test\_new.csv共有2千行数据，没有类别。

1.1.1 类别情况

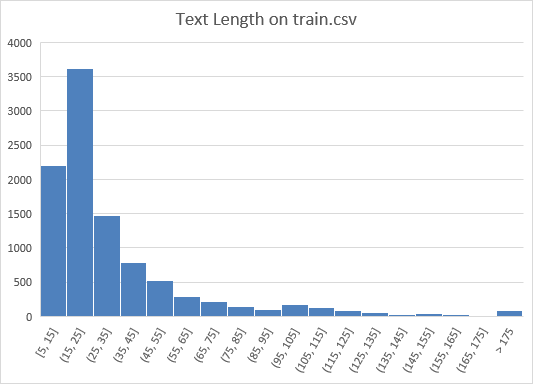
train.csv，数据统计如下：

|  |  |
| --- | --- |
| Not safe | 1511 |
| safe | 8489 |
| total | 10000 |

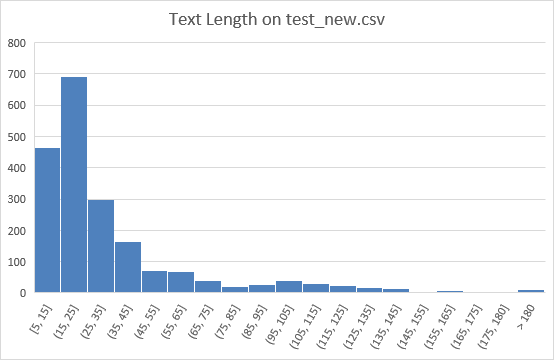
两个标签的比例为not safe : safe = 5.667，不算非常失衡的类型，如有必要可以进行一些处理使得两者比例更为靠近。

1.1.2 句子长度情况

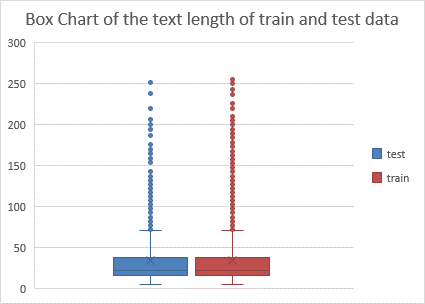
train.csv，数据统计如下：



test\_new.csv，数据统计如下：



test.csv 与 train.csv 的句子长度分布：



结论：从上面的质量分布图中，我们得出，原始数据中，大多数文本长度在5-30之间，而且数据中还包含着不少的非法符号，所以有效的数据长度会更短，所以我们的模型应该选择那些适合短文本分类的。而训练集和测试集的句长分布基本一致，减少了一些工作量。

1.2 处理非法字符

我们不难在评论中发现，评论中混杂着不少的非法符号，如emoji表情、# 等，我们利用python对此进行了删除，只留下中文、字母、数字、中文标点、空格。

*def* is\_chinese\_char(*uchar*):

    return uchar >= *u*'\u4e00' and uchar <= *u*'\u9fa5'

*def* is\_chinese\_punctuation(*uchar*):

    chinese\_punctuations = [*u*'。', *u*'，', *u*'、', *u*'？', *u*'：', *u*'、', *u*'“', *u*'”', *u*'！']

    return uchar in chinese\_punctuations

*def* is\_digist(*uchar*):

    return '0' <= uchar and uchar <= '9'

*def* is\_alpha(*uchar*):

    return ('a' <= uchar and uchar <= 'z') or ('A' <= uchar and uchar <= 'Z')

*def* is\_space(*uchar*):

    return uchar == ' '

*def* remove(*data*, *pattern*):

    """

    remove the char of string in the data, which makes pattern(char) True \\

    return the new data

    """

    return [''.join(filter(pattern, *str*)) for *str* in data]

1.3 处理错别字

一些错别字的存在可能会影响到我们模型对数据的理解，我们利用百度智能云平台的api接口，处理了数据中的错别字。

from aip import AipNlp

app\_id      =   'XXXX'

api\_key     =   'XXXX'

secret\_key  =   'XXXX'

client = AipNlp(app\_id, api\_key, secret\_key)

*def* correct(*data*, *text\_out\_path*, *bin\_out\_path*):

    responses = []

    texts = []

    i = 0

    for line in data:

        if i % 100 == 0:

            print(i)

        i += 1

        response = client.ecnet(line)

        responses.append(response)

        texts.append(response['item']['correct\_query'])

    write\_data(texts, text\_out\_path)

    save\_model(bin\_out\_path, responses)

但是从处理完的结果上看，数据集错别字并没有出现得很多，并且大多数的错别字是生活化的语言，百度提供的接口并不能很好地检测出来，所以这一步的意义不是特别大。

1.3 分词处理

在分词的处理上，我们采用了python 中jieba这个第三方库。

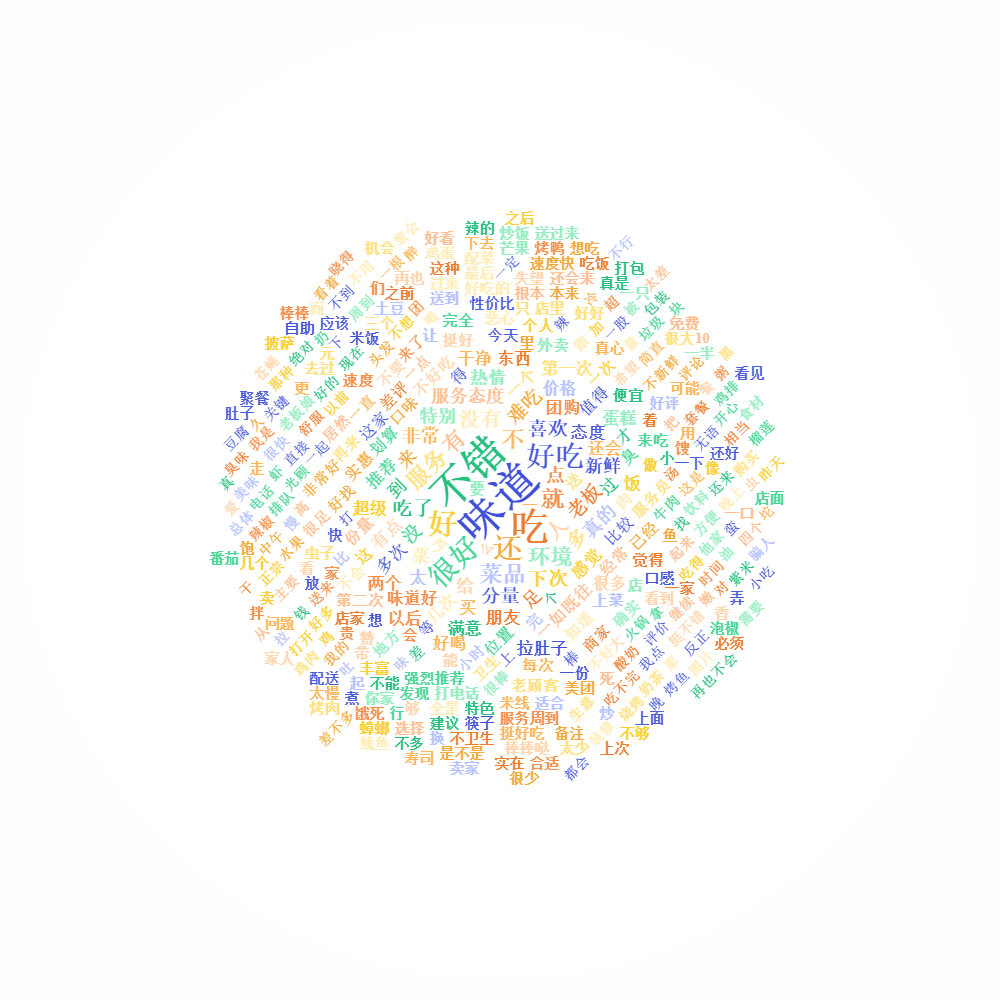
为了让分词更加精确，我们专门去搜狗输入法的词库下载有关餐饮的scel词库，并将它们转化为txt文件，作为jieba中的用户词库。

在停词表上，我们在github上找了`中文停用词表.txt`、`哈工大停用词表.txt`、`百度停用词表.txt`和`四川大学机器智能实验室停用词库.txt`，并将它们合并为我们的停词表。

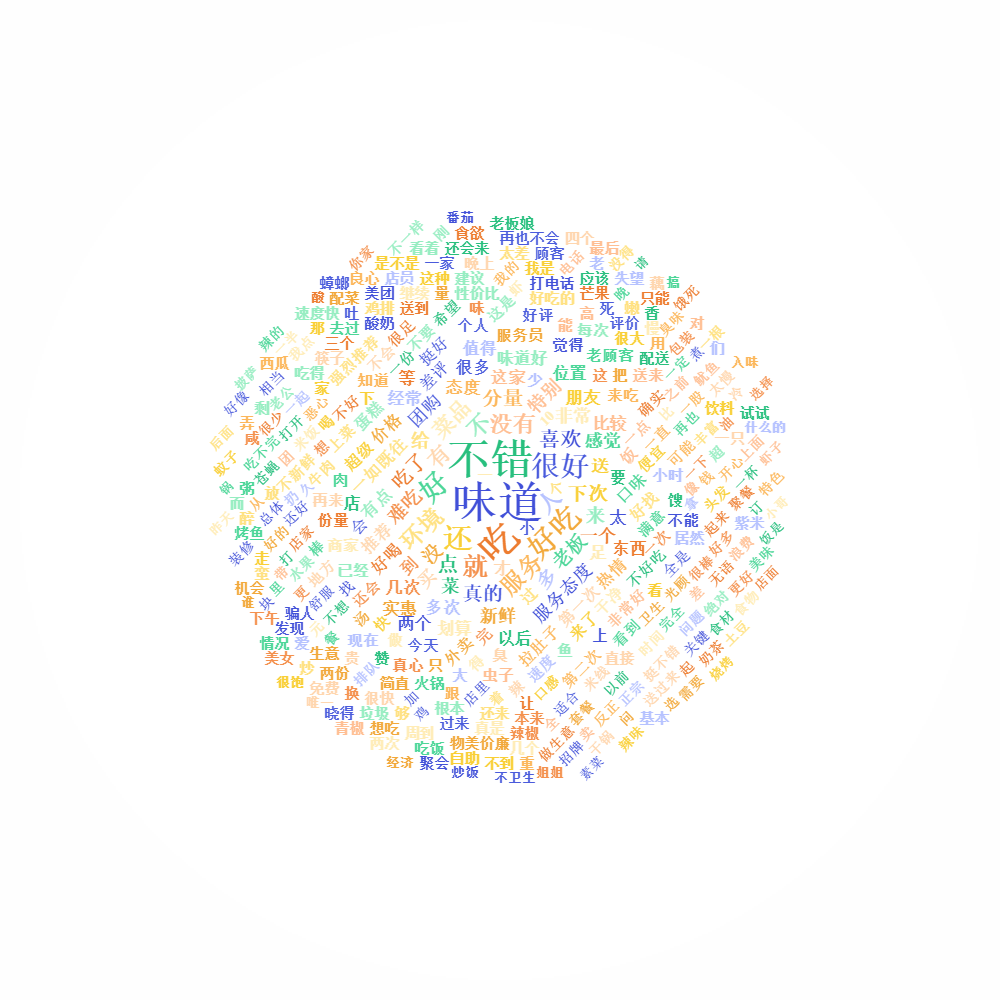
此后，我们通过人工观察停词后的结果，对我们的用户词典和停词表进行改进，再进行分词，以此循环改进分词效果。

下面是我们部分重点词语的频率统计情况。

train 前400个词



test 前400个词



1.3 目前数据处理存在的弊端

虽然添加了很多词库和人工制作词库，分词的结果还是存在一些分错的现象，评论中存在个别错别字无法纠正。

评论中有一些是凑字数的评论，在这一点上还没有想到怎么解决，与此相似的有，有一些评论完全与餐饮无关，起初是想利用分词后，再利用停词表将其删除，但是发现它们出现的词很多，人工添加停词不太实现。

**三、模型介绍**

3.1 word2vec

在Word2vec，为一群用来产生词向量的相关模型。这些模型为浅层双层的神经网络，用来训练以重新建构语言学之词文本。网络以词表现，并且需猜测相邻位置的输入词，在word2vec中词袋模型假设下，词的顺序是不重要的。

训练完成之后，word2vec模型可用来映射每个词到一个向量，可用来表示词对词之间的关系。

得到词向量后，我们针对人工选取的关键字对词向量进行选择，得到一个由多个词向量经过特定运算得出的一个词向量，然后将这些向量放入Logical Regression 和SVM中进行分类，但是得到的结果并不是很好，在测试集上的表现只有0.80左右。

3.2 TextCNN

该模型没有进行详细的研究，只是跑了别人的代码，效果也不是很好。

3.3 Fasttext

3.3.1应用场景

fastText是一种Facebook AI Research在16年开源的一个文本分类器。 其特点就是fast。相对于其它文本分类模型，如SVM，Logistic Regression和neural network等模型，fastText在保持分类效果的同时，大大缩短了训练时间。

3.3.2优缺点

适合大型数据+高效的训练速度：能够训练模型“在使用标准多核CPU的情况下10分钟内处理超过10亿个词汇”

支持多语言表达：利用其语言形态结构，fastText能够被设计用来支持包括英语、德语、西班牙语、法语以及捷克语等多种语言。FastText的性能要比时下流行的word2vec工具明显好上不少，也比其他目前最先进的词态词汇表征要好。

fastText专注于文本分类，在许多标准问题上实现当下最好的表现（例如文本倾向性分析或标签预测）。

3.3.3 应用于比赛数据集的成果

model = train\_supervised(*input*=train\_data, *epoch*=50, *lr*=1.0, *wordNgrams*=5, *verbose*=2, *minCount*=1)

使用fasttext经过调参后提交成绩为**0.88077855000**

3.2 Bert

3.2.1 中文Bert模型对比

ChnSentiCorp

在情感分析任务中，二分类的情感分类数据集ChnSentiCorp。 评测指标为：Accuracy

| **模型** | **开发集** | **测试集** |
| --- | --- | --- |
| BERT | 94.7 (94.3) | 95.0 (94.7) |
| ERNIE | 95.4 (94.8) | 95.4 **(95.3)** |
| **BERT-wwm** | 95.1 (94.5) | 95.4 (95.0) |
| **BERT-wwm-ext** | 95.4 （94.6) | 95.3 (94.7) |
| **RoBERTa-wwm-ext** | 95.0 (94.6) | 95.6 (94.8) |
| **RoBERTa-wwm-ext-large** | **95.8 (94.9)** | **95.8** (94.9) |

根据几种不同模型在情感二分类任务中的表现，我们选择了百度的ERNIE与哈工大讯飞实验室的RoBERTa-wwm-ext-large两个预训练模型进行训练。ERNIE 使用了百度贴吧、百度知道等网络数据，对非正式文本建模有优势，所以重点使用了ERNIE进行训练。

| **模型** | **BERT** | **ERNIE** | **BERT-wwm\*** |
| --- | --- | --- | --- |
| ChnSentiCorp | 2e-5 | 5e-5 | 2e-5 |

根据哈工大讯飞实验室发布的各模型在情感二分类上的最佳学习率，ERNIE与BERT-wwm的最佳学习率为5e-5与2e-5，我们依据这个数据进行调参。

3.2.2 利用各模型得到的成绩

| **模型** | **ERNIE** | **BERT-wwm** |
| --- | --- | --- |
| 在训练集验证数据上的F1-SCORE | **0.9920** | **0.9891** |
| 在测试集上的F1-SCORE（提交） | **0.9425** | **0.9307** |

**四、实验总结**

在这次实验，我们在数据处理上花费了不少数据，更进一步了解数据的处理的难度，因为平时实验中接触到的数据都是别人已经处理好的数据，而且大多数是已经经过了数据化这一步骤。这次也是我们第一次用文本数据进行机器学习，可能对数据进行处理所采用的方法是站在人的角度出发进行思考，可能并不适合机器的实际处理，对此，需要进一步了解文本的处理。