垃圾短信识别实验

# 技术路线图



图 1 垃圾短信文本识别过程

# 具体操作

## 样本情况介绍

样本包含带有0-1标记的文本信息80万条，以CSV格式存储，如下表。标签为1表示为垃圾短信，反之，标签0代表非垃圾短信。经过统计，样本当中垃圾短信和非垃圾短信的数量分别为80000和720000条，样本缺失数量为0，样本重复数量为0，样本总字符数20513652。

表 1 样本内容

|  |  |  |
| --- | --- | --- |
| 序号 | 标签 | 内容 |
| 1 | 0 | 商业秘密的秘密性那是维系其商业价值和垄断地位的前提条件之一 |
| 2 | 1 | 南口阿玛施新春第一批限量春装到店啦   春暖花开淑女裙、冰蓝色公主衫 气质粉小西装、冰丝女王长半裙、 皇 |
| 3 | 0 | 带给我们大常州一场壮观的视觉盛宴 |
| 4 | 0 | 有原因不明的泌尿系统结石等 |
| 5 | 0 | 23年从盐城拉回来的麻麻的嫁妆 |
| … | … | … |
| 799999 | 0 | 费了半天劲各种找关系终于联系上心仪公司的内部人 |
| 800000 | 0 | 是汉奸还是被强奸自己对号入座吧 |



图 2 垃圾短信的词云图



图 3 正常短信的词云图

## 样本预处理

1. 对缺失值样本进行删除处理；
2. 将重复值删除；
3. 将噪声字符（#，￥，&等）删除。

## 使用jieba分词库对短信进行分词

本模型以词向量作为组成文本特征矩阵的基本单位，因此应该以分开的具有独立语言意义的词组作为文本的最小组成要素。有研究表明，分词质量高低与最终的文本分类效果息息相关，因此，快速准确的分词算法是非常重要的。，研究人员不断开发出一些中文分词器，中文分词技术不断地趋向成熟。比较流行的有NLPIＲ(Natural Language Processing ＆ Information Ｒetrieval )、THULAC ( THU Lexical Analyzer for Chinese)、jieba 分词和Snow NLP 等分词器，使用者可以根据应用场景选用适合的中文分词器。由于jieba 分词器是一个用python 语言开发的免费开源的分词工具，并且用户可以根据自己的任务环境自定义词典和词库，所以本文采用jieba 分词器中比较适合于文本分析的精确分词模式对短信文本进行分词处理。

1. 加载用户词库。

jieba.load\_userdict('userdict.txt')

1. 进行分词。

contents\_cut = contents\_clear.astype('str').apply(lambda x: list(jieba.cut(x)))

1. 去除停用词。

stopword\_file = 'stopword.txt'

stopword = pd.read\_csv(stopword\_file, sep = 'bingrong', header = None)

stopword = [' ',',', '会', '的', '】', '【', '月', '日'] + list(stopword[0]) #添加停用词

contents\_stop = contents\_cut.apply(lambda x:[i for i in x if i not in stopword])

## 特征选取

目前主流的特征选取方式有TF-IDF、Word2Vec、CountVectorizer这 3种方式，本文中选取的是CountVectorizer方式。假定有3组文本内容为texts，如下。

texts=["dog cat fish","dog cat cat","fish bird", 'bird']

cv = CountVectorizer()

cv\_fit=cv.fit\_transform(texts)

print(cv.get\_feature\_names())

print(cv\_fit.toarray())

得到的结果是：

#['bird', 'cat', 'dog', 'fish']

#[[0 1 1 1]

# [0 2 1 0]

# [1 0 0 1]

# [1 0 0 0]]

## 模型选择

本文所有的代码、结果已上传至https://github.com/CSQ223/Complex-Network，本文使用朴素贝叶斯模型、支持向量机对数据进行建模、预测。

## 模型评估

实验结果如下表所示。

表 2 实验结果

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **模型 特征向量** | | **朴素贝叶斯** | | | **支持向量机** | | |
| **查准率** | **查全率** | **调和平均率** | **查准率** | **查全率** | **调和平均率** |
| **precision** | **recall** | **f1-score** | **precision** | **recall** | **f1-score** |
| **Count**  **Vectorizer** | **0** | 0.92 | 0.98 | 0.95 | **0.99** | **0.95** | **0.97** |
| **1** | 0.98 | 0.91 | 0.94 | **0.95** | **0.99** | **0.97** |
| **avg** | 0.95 | 0.94 | 0.94 | **0.97** | **0.97** | **0.97** |
| **TF-IDF** | **0** | 0.91 | 0.98 | 0.94 | 0.97 | 0.96 | 0.96 |
| **1** | 0.98 | 0.9 | 0.94 | 0.96 | 0.97 | 0.97 |
| **avg** | 0.94 | 0.94 | 0.94 | 0.97 | 0.97 | 0.97 |

## 结论

从表2中可以看出，支持向量机与CountVectorizer的模型平均查准率为0.97，平均查全率为0.97，调和平均率为0.97，相对其他的结果而言，预测结果比较好。