《数据科学导论》文本大作业 实验报告

中国人民大学 信息学院 崔冠宇 2018202147

备注:

为了减小压缩包体积, 短信数据以及中间结果不被包含在压缩包内, 运行时请将短信数据文件夹置于 notebook 所在目录内.

一、实验目的

- 进一步熟悉文本表达的有关内容;
- 回顾分类模型, 利用旧有、新学的模型进行短信分类与预测, 并测试分类效果;
- 初步学习 Elasticsearch® 框架的使用, 用其搭建一个简易搜索引擎.

二、解决方案结构

本次大作业是以 jupyter notebook 的形式完成的, 代码共分为三大部分:

1. **预处理:** 主要是观察及清洗数据、分词. 由于分词系统开销大, 运行较慢, 所以将分词后的结果存入文件, 供之后直接使用.
2. **文本表达与分类:** 用 tf-idf 做文本表达, 对比去除停用词与不去除停用词的效果, 对比数个常见分类模型(朴素贝叶斯、KNN以及决策树)的效果, 选出这些模型中效果最理想的一个, 作为后续预测打标签的工具.
3. **文本检索:** 将无标签短信用上一步的模型进行预测, 将预测结果与有标签短信合并, 写入 Elasticsearch 中, 并进行查询.

三、代码说明及各步结果

(具体代码也可见“文本模块大作业.ipynb”)

1. **预处理:**

   (1) 首先使用自己编写的 my_read_txt() 函数加载带标签短信数据. 用 value_counts() 观察数据, 发现共有 800k 条数据, 其中有 720k 条正常信息, 80k 条垃圾短信.

```
# 加载短信内容
def my_read_txt(file, is_labeled = True, names = None):
    with open(file, 'r', encoding = 'utf-8') as f:
        if is_labeled:
            texts = [[int(line.strip()[0]), line.strip()[2:]] for line in f]
        else:
            texts = [line.strip() for line in f]
        return pd.DataFrame(texts, columns = names)
# 带标签
df_wlabel = my_read_txt('短信数据/带标签短信.txt', names = ['标签', '内容'])
df_wlabel
```

	标签	内容
0	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
1	1	南口阿玛施新春第一批限量春装到店啦口口口春暖花开淑女裙、冰蓝色公主衫口...
2	0	带给我们大常州一场壮观的视觉盛宴
3	0	有原因不明的泌尿系统结石等
4	0	23年从盐城拉回来的麻麻的嫁妆
...	...	...
799995	0	助排毒缓解痛经预防子宫肌瘤&amp
799996	0	这是今年首次启动I级防台应急响应
799997	0	丽江下飞机时迎接我们的是凉风
799998	0	费了半天劲各种找关系终于联系上心仪公司的内部人
799999	0	是汉奸还是被强奸自己对号入座吧

800000 rows × 2 columns

```
# 观察数据分布
df_wlabel['标签'].value_counts()
```

```
0    720000
1     80000
Name: 标签, dtype: int64
```

(2) 然后将两类短信各自用 sample() 函数抽样15条, 观察. 首先是垃圾短信:

```
# 垃圾短信抽样
df_wlabel[df_wlabel['标签'] == 1].sample(n = 15)
```

	标签	内容
357947	1	您好，我是十九中数学老师，家住石板坡xxx号大院，与同事英语，物理，化学老师周末可在我家为孩...
135390	1	优惠打折不敢想象：开年抄底最好机会！来吧，首付分期x万起总价xx.xx万起，xx平方x室x厅...
223342	1	【路卡迪龙】回馈老顾客巨额优惠！x.x日—x.x日，全场冬装x—x折，货品、号码不足，非常抱...
141880	1	本人承办信誉小额x到x万贷款，无需抵押，快速下款，联系电话xxxxxxxxxx
146626	1	大桥外语虹光校区在x月x日安排了新班英语公开课，欢迎家长带孩子试听，咨询电话:xxxxxx...
698659	1	亲爱的:我是潇潇，祝您节日快乐哦！x.x节活动来啦！【JNBY】春季新款购满x件x折优惠，更...
80355	1	亲你好，我是鸥美药妆现代广场的，之前打电话预约您这个周末过来参加十年回馈史无前例的三八大型促...
524962	1	xxxx"装修大红包"让您开年就鸿运当头！现场加名仕VIP客户微信群抢xxx元红包等更多惊喜...
220007	1	仅需xx元就可享受价值xxxx元的光量子嫩肤/局部脱毛一次，您的闺蜜也可享受柏雅美业价值xx...
102917	1	尊敬的业主，新年好，我公司专业现浇钢结构，封阳台，已在中铁西城二期(西苑)做了许多家样板，品...
178872	1	迎新年贺新春，康怡xxxx年开门红开始抢购啦！[闪电][闪电]做美容闺密同行，xxx可享受x...
53011	1	温州华腾店感谢您的来电，店面全面升级，年中冲量!全系车型特惠酬宾，最高优惠达x万，更有全新马...
176456	1	您好，恒大山水城：石露平xxxxxxxxxx，新年将至，又一度楼市调控，新房开盘，优先楼层...
242995	1	尊敬的会员朋友您好:本月x日至xx日本店推出x.x元的特价商品，事量有限，欢迎回店购买。...
414910	1	英超精选第二场，xx:xx伯恩利VS斯旺西，由平手分析看好主队不败，推荐伯恩利让平手。今日共...

其次是正常短信:

```
# 普通短信抽样
df_wlabel[df_wlabel['标签'] == 0].sample(n = 15)
```

	标签	内容
305119	0	哦凑看了这么久的花千骨也是才意识到
68954	0	怎么到了南京连名字都给我改了
164985	0	百度欲动用120亿美元的现金和等价资产投资用户服务
552959	0	在PIAUI州沿海的PARNAIBA市
260154	0	美国智能手机用户每天花费在移动应用上的时间为127分钟
598192	0	采用独有的个性化设计和制作
324821	0	南京哪10个路口最容易闯红灯吗
427519	0	民生控股合肥百货小天鹅A通程控股吉林化纤南京中北
645284	0	官员说了算是一切腐败的根源
308248	0	中国好声音怎么几乎全是华裔选手
282397	0	有個黑粉常在指原莉乃的Twitter上罵她
674694	0	由设计师YooriKoo带来的Elasty爱疯4手机外壳
606378	0	中国电信与医疗器械行业云端峰会在贵阳召开
64316	0	今天下午真相大白的感觉就是心从天堂到地狱
324991	0	李中堂认为商业保险可以解决一切问题

(3) 之后再用 drop_duplicates() 进行去重工作, 发现去掉了 10k+ 条数据:

```
# 去除重复短信
df_wlabel.drop_duplicates(subset = None, inplace = True)
df_wlabel = df_wlabel.reset_index(drop = True)
df_wlabel
```

	标签	内容
0	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
1	1	南口阿玛施新春第一批限量春装到店啦口口口春暖花开淑女裙、冰蓝色公主衫口...
2	0	带给我们大常州一场壮观的视觉盛宴
3	0	有原因不明的泌尿系统结石等
4	0	23年从盐城拉回来的麻麻的嫁妆
...	...	...
786595	0	助排毒缓解痛经预防子宫肌瘤&amp
786596	0	这是今年首次启动I级防台应急响应
786597	0	丽江下飞机时迎接我们的是凉风
786598	0	费了半天劲各种找关系终于联系上心仪公司的内部人
786599	0	是汉奸还是被强奸自己对号入座吧

786600 rows × 2 columns

(4) 接下来用 jieba 包分词, 并将分词后的 DataFrame 保存, 便于后续使用:

```python
import jieba

# 分词函数
# 传入列，停用词文件
def word_cut(column, file_stopwords = 'hit_stopwords.txt'):
    result = []
    if file_stopwords is not None:
        with open(file_stopwords) as f:
            stoplist = f.read();
    for s in column:
        words_list = []
        bagofwords = jieba.cut(s, cut_all = False)
        if file_stopwords is not None:
            for word in bagofwords:
                if word not in stoplist:
                    words_list.append(word)
        else:
            for word in bagofwords:
                words_list.append(word)
        result.append((' '.join(words_list)))
    return pd.Series(result)
```

```python
# 由于查找停用词加分词很慢，先做初步分词，保存csv，后续直接加载节省时间
# 处理 stopwords
#parsed_df_wlabel = pd.DataFrame(df_wlabel['标签'], columns = ['标签'])
#parsed_df_wlabel['内容'] = word_cut(df_wlabel['内容'])
#parsed_df_wlabel
```

```python
#parsed_df_wlabel.to_csv('短信数据(分词后)/带标签短信(分词后).csv', index = False)
```

```python
# 不处理 stopwords
#parsed_df_wlabel_wstop = pd.DataFrame(df_wlabel['标签'], columns = ['标签'])
#parsed_df_wlabel_wstop['内容'] = word_cut(df_wlabel['内容'], file_stopwords = None)
#parsed_df_wlabel_wstop.to_csv('短信数据(分词后)/带标签短信(分词后，含停用词).csv', index = False)
```

2. 文本表达与分类:

(1) 首先加载分词后文件, 进行训练集、测试集数据分割, 分割比例 0.15:

```python
# 注意：因为分词后会有空内容，需要处理一下
train_parsed_df = pd.read_csv('短信数据(分词后)/带标签短信(分词后).csv')
train_parsed_df['内容'].fillna('', inplace = True)
train_parsed_df
```

	标签	内容
0	0	商业秘密 秘密性 维系 商业价值 垄断 地位 前提条件
1	1	南口 阿玛施 新春 第一批 限量 春装 店□□□春暖花开 淑女 裙 冰 蓝色 公主 ...
2	0	带给 大 常州 一场 壮观 视觉 盛宴
3	0	原因 不明 泌尿系统 结石
4	0	23 盐城 拉回来 麻麻 嫁妆
...	...	...
786595	0	助 排毒 缓解 痛经 预防 子宫 肌瘤 amp
786596	0	这是 今年 首次 启动 I 级 防台 应急 响应
786597	0	丽江 飞机 迎接 凉风
786598	0	费 半天 劲 找 关系 终于 联系 心仪 公司 内部

```python
# 分测试集
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    train_parsed_df['内容'], train_parsed_df['标签'],
    test_size = 0.15, random_state = 0)
```

(2) 然后利用 sklearn 的 CountVectorizer 和 TfidfTransformer 进行文本特征提取, 用于后续分类处理:

```python
# 特征提取
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer

# 训练集文本特征提取
vectorizer = CountVectorizer()
X_train_termcounts = vectorizer.fit_transform(X_train)
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_termcounts)

# 测试集进行文本特征提取
X_test_termcounts  = vectorizer.transform(X_test)
X_test_tfidf = tfidf_transformer.transform(X_test_termcounts)
```

(3) 首先展示在朴素贝叶斯 (MultinomialNB) 下的训练、预测结果:

```python
# 建立朴素贝叶斯分类器并进行训练
from sklearn.naive_bayes import MultinomialNB, GaussianNB
MNBclf = MultinomialNB().fit(X_train_tfidf, y_train)
```

我认为，在垃圾短信分类的情景中，对于正常信息要同时关注准确度(precision),

和召回率(recall)，因为我们不能"错杀"用户的正常短信;

对于垃圾短信，准确度尽量要高，同样是不能"错杀"正常短信; ¶

但我们可以允许漏掉一些，那么对于召回率要求相对较弱

```python
from sklearn.metrics import classification_report, confusion_matrix

# 训练集上的评估
y_train_predicted = MNBclf.predict(X_train_tfidf)
print(classification_report(y_train, y_train_predicted))

# 测试集上的评估
y_test_predicted = MNBclf.predict(X_test_tfidf)
print(classification_report(y_test, y_test_predicted))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99    601412
           1       0.98      0.91      0.94     67198

    accuracy                           0.99    668610
   macro avg       0.99      0.95      0.97    668610
weighted avg       0.99      0.99      0.99    668610

              precision    recall  f1-score   support

           0       0.99      1.00      0.99    106048
           1       0.98      0.88      0.93     11942

    accuracy                           0.99    117990
   macro avg       0.98      0.94      0.96    117990
weighted avg       0.99      0.99      0.99    117990
```

可见模型对于正常短信的精度和召回率都很高, 但对于垃圾短信的召回率则有明显的降低. 但这也符合实际情景——我们对于垃圾短信的选择要谨慎, 尽量不要将正常短信错误标记为垃圾短信, 这样会影响用户的正常通信.

(4) 接下来, 我对比了上述模型不去除停用词的效果, 如下图所示, 发现差距不大:

```
# 看看不去掉停用词的训练效果
train_parsed_df = pd.read_csv('短信数据(分词后)/带标签短信(分词后, 含停用词).csv')
train_parsed_df['内容'].fillna('', inplace = True)

X_train, X_test, y_train, y_test = train_test_split(
    train_parsed_df['内容'], train_parsed_df['标签'],
    test_size = 0.15, random_state = 0)

# 训练集文本特征提取
vectorizer = CountVectorizer()
X_train_termcounts = vectorizer.fit_transform(X_train)
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_termcounts)

# 测试集进行文本特征提取
X_test_termcounts  = vectorizer.transform(X_test)
X_test_tfidf = tfidf_transformer.transform(X_test_termcounts)

classifier = MultinomialNB().fit(X_train_tfidf, y_train)

# 训练集上的评估
y_train_predicted = classifier.predict(X_train_tfidf)
print(classification_report(y_train, y_train_predicted))

# 测试集上的评估
y_test_predicted = classifier.predict(X_test_tfidf)
print(classification_report(y_test, y_test_predicted))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99    601412
           1       0.98      0.91      0.94     67198

    accuracy                           0.99    668610
   macro avg       0.99      0.95      0.97    668610
weighted avg       0.99      0.99      0.99    668610

              precision    recall  f1-score   support

           0       0.99      1.00      0.99    106048
           1       0.98      0.88      0.93     11942

    accuracy                           0.99    117990
   macro avg       0.98      0.94      0.96    117990
weighted avg       0.99      0.99      0.99    117990
```

(5) 同时我还比较了 GaussianNB、KNN以及决策树的分类效果, 如下所示:

```
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import Normalizer
from sklearn.pipeline import make_pipeline

# GaussianNB 不接受稀疏矩阵, 尝试SVD降维
svd = TruncatedSVD(5)
normalizer = Normalizer(copy = False)
lsa = make_pipeline(svd, normalizer)
X_train_new = lsa.fit_transform(X_train_tfidf)
X_test_new = lsa.transform(X_test_tfidf)
```

```
GNBclf = GaussianNB().fit(X_train_new, y_train)
# 训练集上的评估
y_train_predicted = GNBclf.predict(X_train_new)
print(classification_report(y_train, y_train_predicted))

# 测试集上的评估
y_test_predicted = GNBclf.predict(X_test_new)
print(classification_report(y_test, y_test_predicted))
```

```
              precision    recall  f1-score   support

           0       0.99      0.92      0.95    567776
           1       0.56      0.90      0.69     63656

    accuracy                           0.92    631432
   macro avg       0.77      0.91      0.82    631432
weighted avg       0.95      0.92      0.93    631432

              precision    recall  f1-score   support

           0       0.99      0.92      0.95    100355
           1       0.56      0.91      0.69     11075

    accuracy                           0.92    111430
   macro avg       0.78      0.91      0.82    111430
weighted avg       0.95      0.92      0.93    111430
```

GaussianNB 的表现可谓“一塌糊涂”, 对于垃圾短信的精度都没有及格, 但是召回率稍高于 MultinomialNB.

```python
# KNN模型
from sklearn.neighbors import KNeighborsClassifier

KNNclf = KNeighborsClassifier()
KNNclf.fit(X_train_new, y_train)

# 训练集上的评估
y_train_predicted = KNNclf.predict(X_train_new)
print(classification_report(y_train, y_train_predicted))

# 测试集上的评估
y_test_predicted = KNNclf.predict(X_test_new)
print(classification_report(y_test, y_test_predicted))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99    601412
           1       0.89      0.85      0.87     67198

    accuracy                           0.97    668610
   macro avg       0.94      0.92      0.93    668610
weighted avg       0.97      0.97      0.97    668610

              precision    recall  f1-score   support

           0       0.98      0.98      0.98    106048
           1       0.81      0.79      0.80     11942

    accuracy                           0.96    117990
   macro avg       0.89      0.88      0.89    117990
weighted avg       0.96      0.96      0.96    117990
```

使用 KNN 进行分类, 结果对于垃圾短信的标记精度要好于上一个模型, 但召回率又低了.

```python
from sklearn.tree import DecisionTreeClassifier

DTclf = DecisionTreeClassifier(max_depth = 10, min_samples_split = 2)
DTclf.fit(X_train_new, y_train)

# 训练集上的评估
y_train_predicted = DTclf.predict(X_train_new)
print(classification_report(y_train, y_train_predicted))

# 测试集上的评估
y_test_predicted = DTclf.predict(X_test_new)
print(classification_report(y_test, y_test_predicted))
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98    601412
           1       0.84      0.81      0.83     67198

    accuracy                           0.97    668610
   macro avg       0.91      0.90      0.90    668610
weighted avg       0.97      0.97      0.97    668610

              precision    recall  f1-score   support

           0       0.98      0.98      0.98    106048
           1       0.80      0.78      0.79     11942

    accuracy                           0.96    117990
   macro avg       0.89      0.88      0.88    117990
weighted avg       0.96      0.96      0.96    117990
```

最后是决策树, 可见分类效果和 KNN 相比略差, 但好于 GaussianNB, 而且训练较快.

通过多模型之间的比较, 我们就选出了一个综合水平较好的模型——MultinomialNB, 作为后续的预测所用的模型.

(6) 接下来加载分词后的带标签数据, 用朴素贝叶斯模型进行训练; 同时加载无标签数据, 进行去重:

```
train_parsed_df = pd.read_csv('短信数据(分词后)/带标签短信(分词后).csv')
train_parsed_df['内容'].fillna('', inplace = True)

# 训练集文本特征提取
vectorizer = CountVectorizer()
X_train_termcounts = vectorizer.fit_transform(X_train)
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_termcounts)

# 测试集进行文本特征提取
X_test_termcounts  = vectorizer.transform(X_test)
X_test_tfidf = tfidf_transformer.transform(X_test_termcounts)
# 建立朴素贝叶斯分类器并进行训练
from sklearn.naive_bayes import MultinomialNB, GaussianNB
MNBclf = MultinomialNB().fit(X_train_tfidf, y_train)
```

```
df_wolabel = my_read_txt('短信数据/不带标签短信.txt', is_labeled = False, names = ['内容'])
# 去除重复短信
df_wolabel.drop_duplicates(subset = None, inplace = True)
df_wolabel = df_wolabel.reset_index(drop = True)
df_wolabel
```

	内容
0	.x月xx日推出凭证式国债x年期x.xx.xx%，x年期x.xx%到期一次还本付息。真情邮政...
1	x强度等级水泥的必要性和可行性进行深入研究
2	Don'tSellaProduct
3	以上比赛规则由江苏科技大学教职工摄影协会负责解释
4	坐12个小时飞机身体已经疲惫不堪
...	...
198936	之前国内amazon看到有说买到假货
198937	无锡市金羿杭金融服务有限公司产品大纲1
198938	融券改为T+1限制日内回转交易
198939	康大预诊只要你用心做只要下个软件问医生三个问题一个个问还可以发展下家就能赚钱很简单不要交什么...
198940	电话：028—86301255

198941 rows × 1 columns

(7) 将无标签短信数据分词, 用上面训练的模型给无标签数据预测打标签:

```
# 分词, 仍是保存
#parsed_df_wolabel = pd.DataFrame(word_cut(df_wolabel['内容']), columns = ['内容'])
#parsed_df_wolabel
```

```
#parsed_df_wolabel.to_csv('短信数据(分词后)/不带标签短信(分词后).csv', index = False)
```

```
parsed_df_wolabel = pd.read_csv('短信数据(分词后)/不带标签短信(分词后).csv')
# 填充空数据
parsed_df_wolabel['内容'].fillna('', inplace = True)
# 文本特征提取
X_pred_termcounts = vectorizer.transform(df_wolabel['内容'])
X_pred_tfidf = tfidf_transformer.transform(X_pred_termcounts)
y_pred = MNBclf.predict(X_pred_tfidf)
```

```
# 预测
parsed_df_wolabel['标签'] = y_pred
parsed_df_wolabel = parsed_df_wolabel[['标签', '内容']]
parsed_df_wolabel
```

	标签	内容
0	0	月 xx 推出 凭证式 国债 年期 xx xx% 年期 xx% 到期 一次 还本付息 真情 ...
1	0	强度 等级 水泥 必要性 可行性 进行 深入研究
2	0	Don tSellaProduct
3	0	以上 比赛规则 江苏 科技 大学 教职工 摄影 协会 负责 解释
4	0	坐 12 小时 飞机 身体 已经 疲惫不堪
...	...	...
198936	0	之前 国内 amazon 看到 买 假货
198937	0	无锡市 金羿 杭 金融服务 有限公司 产品 大纲 1
198938	0	融券 改为 T 1 限制 日内 回转 交易
198939	0	康大 预诊 用心 做 下个 软件 医生 三个 问题 一个个 发展 下家 赚钱 很 简单 不要...
198940	0	电话 028 86301255

198941 rows × 2 columns

(8) 最后合并两组数据并保存, 为下面搜索引擎的建立做准备:

```
# 整合到原 DF 上
df_wolabel['标签'] = y_pred
df_wolabel = df_wolabel[['标签', '内容']]
```

```
# 拼接去重并保存
messages = pd.concat([df_wlabel, df_wolabel], ignore_index = True)
messages.drop_duplicates(subset = None, inplace = True)
messages = messages.reset_index(drop = True)
messages.to_csv('所有短信.csv', index = False)
messages
```

	标签	内容
0	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
1	1	南口阿玛施新春第一批限量春装到店啦口口口春暖花开淑女裙、冰蓝色公主衫口...
2	0	带给我们大常州一场壮观的视觉盛宴
3	0	有原因不明的泌尿系统结石等
4	0	23年从盐城拉回来的麻麻的嫁妆
...	...	...
979868	0	所拍数目悉数捐于苏州大观音禅寺
979869	0	之前国内amazon看到有说买到假货
979870	0	无锡市金羿杭金融服务有限公司产品大纲1
979871	0	康大预诊只要你用心做只要下个软件问医生三个问题一个个问还可以发展下家就能赚钱很简单不要交什么...
979872	0	电话：028—86301255

979873 rows × 2 columns

(9) 除此之外, 我还看了一下两类短信的词云:

```
# 看看两者词云吧
import matplotlib
import matplotlib.pyplot as plt
# 词云
from wordcloud import WordCloud,STOPWORDS
# 标题使用中文
myfont = matplotlib.font_manager.FontProperties(fname="HYQiHei-105.ttf", size=30)
matplotlib.rcParams['axes.unicode_minus'] = False

# 绘制词云
# words: 词列表; title: 词云标题; color: 背景色; mask: 轮廓图
def draw_wordcloud (words, title = None, color = 'white', mask = None):
    wordcloud = WordCloud(stopwords = STOPWORDS,
                          font_path="HYQiHei-105.ttf",
                          background_color=color,
                          mask=mask,
                          width=1000,
                          height=1500
                         ).generate(words)
    plt.figure(1, figsize=(10, 5))
    # 标题中文
    plt.title(title, fontproperties = myfont)
    plt.imshow(wordcloud)
    plt.axis('off')
    plt.show()
```

我们可以发现, 正常短信是具有较强的生活气息的.

```
# 导入背景轮廓图用
import PIL .Image as image

messages_parsed = pd.concat([train_parsed_df, parsed_df_wolabel], ignore_index = True)
messages_parsed.drop_duplicates(subset = None, inplace = True)
messages_parsed = messages_parsed.reset_index(drop = True)

# 收集词汇
words_normal = ' '.join(messages_parsed[messages_parsed['标签'] == 0]['内容'])
print('正常短信词云:')
draw_wordcloud(words_normal, title ='正常短信词云', color='white')
```

正常短信词云:

而垃圾短信则多是推销信息.

```
words_spam = ' '.join(messages_parsed[messages_parsed['标签'] == 1]['内容'])
print('垃圾短信词云:')
draw_wordcloud(words_spam, title ='垃圾短信词云', color='white')
```

垃圾短信词云:

垃圾短信主要是推销信息

3. **文本检索:**

需要指出, 查询时的分词需要我们安装一个 Elasticsearch 的插件: 在 elasticsearch 目录下执行: ./bin/elasticsearch-plugin install https://github.com/medcl/elasticsearch-analysis-ik/releases/download/v7.7.0/elasticsearch-analysis-ik-7.7.0.zip

安装完插件后, 建立搜索引擎以及查询时需要保证 Elasticsearch 正常运行.

(1) 加载所有短信文件, 在 Elasticsearch 中创建“message”索引, 设置好 mapping 后, 使用helpers.bulk() 将短信批量加入搜索引擎中.

```
# 导入包
from elasticsearch import Elasticsearch
es = Elasticsearch()
# 创建 message 索引
es.indices.delete(index = 'messages', ignore = [400, 404])
es.indices.create(index = 'messages', ignore = 400)
# mapping 分词
mapping = {
    'properties':{
        'content': {
            'type': 'text',
            'analyzer': 'ik_max_word',
            'search_analyzer': 'ik_max_word'
        }
    }
}
result = es.indices.put_mapping(index = 'messages', body = mapping)
result
```

```
{'acknowledged': True}
```

```
from elasticsearch import helpers
# 按格式写到列表中, 批量添加到 elasticsearch 中
actions = [
    {
        '_index':'messages',
        '_source':
        {
            'label': row['标签'],
            'content': row['内容']
        }
    } for index, row in messages.iterrows()
]
res = helpers.bulk(es, actions)
res
```

```
(985541, [])
```

(2) 然后我编写了一个查询函数, 参数是查询内容和显示条数, 以 DataFrame 的形式返回查询结果:

```
import json
# 查询函数，querystr 表示用户输入查询关键词，size 表示需要的条数
def es_query(querystr, size = 10):
    dsl = {
        'query':{
            'match':{
                'content': querystr
            }
        }
    }
    # 查询结果
    result = es.search(index = 'messages', body = dsl, size = size)
    contents = []
    # 处理成 tuple，方便显示结果给用户
    for element in result['hits']['hits']:
        contents.append(
            tuple(
                (element['_source']['content'],
                 '是' if element['_source']['label'] == 1 else '否',
                 element['_score'])
            )
        )
    queryresult = pd.DataFrame(contents, columns = ['内容', '是否垃圾短信', '相关性分值'])
    return queryresult
```

以下是几个查询示例:

```
es_query('春装到店啦', 5)
```

	内容	是否垃圾短信	相关性分值
0	原梧高《名店服饰》 新货到！春装到！ 名店新款春衣裳， 俏俏丽人换新装。 全天营业，恭候光临！	是	17.094020
1	南口阿玛施新春第一批限量春装到店啦口口口 春暖花开淑女裙、冰蓝色公主衫口 ...	是	17.077433
2	您好：振华商厦五楼THENxRTHFACE（北面）现秋冬装x折优惠，还有新品春装到店，欢迎来选购！	是	16.967121
3	你好姐姐，我是银泰哥弟店长的晓玲，店内亮色新款春装到店，糖果色系春装让您靓丽迷人，多款多色风...	否	15.231396
4	三八女人节唯有宠爱自己，东方广场置地名店旗下倪儿麦新款春装到货啦！x号至八号特推出新款x.x...	是	15.061478

```
es_query('优惠活动', 5)
```

	内容	是否垃圾短信	相关性分值
0	您好现在草原铁骨丹有大型的优惠活动	是	15.609194
1	你好 新年好 现飞鹤奶粉有优惠活动	是	15.175663
2	【翼支付】翼支付优惠再次来袭，绿姿蛋糕消费xxx返xx，超市消费每周三xx折。翼支付消费立返...	否	15.118975
3	奶粉x月x日当天x.x折的优惠活动 活动啦！！！	否	14.974382
4	新春优惠-回馈活动；一亿红包-任性抢； 详情-上-金-沙-娱-乐【xxxxxx、C xM}；...	是	14.771811

```
es_query('烧烤 奶茶', 5)
```

	内容	是否垃圾短信	相关性分值
0	奶茶MM会来南京执掌京东JD+奶茶馆	否	13.459444
1	妈的回徐州老子要看电影看电影看电影看电影看电影看电影吃烧烤吃烧烤吃烧烤吃烧烤吃烧烤吃烧烤	否	12.856813
2	横河觅城鲜饮奶茶店招聘奶茶操作员两名	否	12.078279
3	3、将烧烤酱与酱油、蜂蜜调成烧烤汁	否	11.951901
4	几天前跑去downtown喝奶茶	否	11.768408

```
es_query('人民大学', 5)
```

	内容	是否垃圾短信	相关性分值
0	简介为人民大学08金融博士	否	24.427206
1	该男生简介为人民大学xx金融博士	否	22.990513
2	中国人民大学MBA学历&amp	否	22.333730
3	称所有人民大学xxxx届学生中	否	22.333730
4	有质疑称武汉大学的录取通知书“抄袭”了人民大学	否	21.887304

从上面的几个搜索示例来看, 我们的分类模型确实是有些保守了, 比如我们漏掉了几个不是很明显的垃圾短信. 但是我们将正常短信几乎都标记为否, 不会耽误用户的正常使用.

四、其他想法

1. 我认为, 我们的项目还可扩展. 比如可以做一个不依赖 jupyter notebook 的搜索引擎, 写一个网页前端, 后端调用 Elasticsearch 作为搜索引擎. 这个网页前端有输入框, 用户可以输入, 并交给后端查询, 结果会如我们常见的搜索引擎那样呈现给用户. 由于我没有 java、HTML 的编程经验, 我只能用之前的代码, 使用 python 模拟实现了一个交互式查询功能软件.

   首先需要运行 Elasticsearch, 然后运行 SearchEngine 目录下的 backend.py, 利用之前的数据导入代码建立查询的数据库. 以上是“服务商”的工作, 然后用户通过运行 frontend.py 和我们建立的查询数据库进行交互. 如下图所示:

2. 似乎我们的模型准确度还不够高, 垃圾短信的召回率没有达到我的预期(95%+), 所以后续还可以继续考虑如何提高垃圾短信的召回率.