网络数据挖掘大作业报告——垃圾短信分类

摘要

短信业务的迅猛发展在丰富了人们的沟通方式的同时，同样遭受到垃圾短信的困扰。对于运营商来说，垃圾短信造成基础设施资源的巨大浪费；对于移动用户来说，大量的垃圾短信使用户不能够及时查看正常的短信，干扰了用户的正常生活。垃圾短信的识别已经成为一个亟待解决的问题，而传统的基于黑白名单、关键字进行过滤的效果有限，不能起到很好的识别效果。针对该问题，我们基于垃圾短信的文本内容，将文本分类算法应用到垃圾短信的分类中。我们使用了SVM、LR、GBDT 和决策树算法进行垃圾短信的识别工作，最后我们制作了线上演示系统。结果显示系统在垃圾短信的识别上有着良好的表现。

关键词：垃圾短信、文本分类、GBDT、LR

一、概述

垃圾短信日益成为困扰运营商和手机用户的难题，严重影响人们的日常生活。根据腾讯《2018 年上半年手机安全报告》[1]显示，2018 年上半年，用户通过腾讯手机管家共举报垃圾短信近 8.89 亿条，其中广告类短信占比高达 96.71%。除了广告类短信，根据短信内容不同，骚扰类、欺诈类、非法广告短信、危害国家安全、散布谣言、侮辱或诽谤他人的短信都属于垃圾短信的范畴。

对于运营商来说，大量的垃圾短信会耗费过多的资源，造成资源的浪费，并增加了网络遭到恶意攻击的风险；对于用户来说，大量的垃圾短信使用户无法及时查看到有用的信息，带来不良的用户体验。因而对垃圾短信进行有效识别是十分必要的。传统的基于黑白名单、关键词过滤等方法对于垃圾短信的识别效果有限。从垃圾短信的内容角度进行考虑，垃圾短信的识别问题可以归为文本二分类问题，目前在文本的二分类问题方面已经有诸多成熟的方法，因而可以将这些方法应用到垃圾短信的识别工作中。

我们针对垃圾短信的文本内容，对垃圾短信的文本进行特征提取并利用SVM、LR、GBDT、决策树方法进行分类工作，最后我们制作了线上演示系统来识别垃圾短信。

本文的组织结构如下：第一部分介绍了垃圾短信的概念与现状及本文的主要工作；第二部分介绍了文本分类的相关工作；第三部分为数据分析；第四部分为研究方法，介绍了我们所使用的分类算法；第五部分为实验设计，介绍了实验的过程及结果；第六部分为总结，包括对实验总结与人员分工情况。

二、相关工作

文本分类最早可以追溯到 20 世纪 60 年代，在这之前主要是采用手工分类的方法。进入 60 年代后，Maron 发表了具有里程碑作用的论文《Automatic Indexing: An Experimental Inquiry》[2]，采用贝叶斯公式进行文本分类，大大推进了文本分类工作。在该文中，Maron 还假设特征间是相互独立的，这就是后来被广泛采用的“贝叶斯假设”。

在随后的二十多年，主要是采用知识工程（Knowledge Engineering, KE）的方法进行文本分类，它通过在专家知识基础上手工建立一系列分类规则来构建分类器。知识工程方法需要大量领域的专家和工程师参与，势必耗费很多人力物力，当电子文档急剧增长时将无法满足需求。这种方法最典型的应用实例为由 Carnegie Group 开发的 CONSTRUE 系统[3]，该系统用来对路透社的新闻稿件自动分类。

直到进入 20 世纪 90 年代，随着 Internet 的迅猛发展，为了能够更好地处理大量的电子文档，并且伴随着人工智能、机器学习、模式识别、统计理论等学科的发展，基于知识工程的文本分类方法渐渐退出了历史舞台，文本分类技术进入了更深入的自动分类时代。由于基于机器学习的自动文本分类系统几乎可以达到与人类专家相当的正确度，但是却不需要任何知识工程师或领域专家的干预，节约了大量的人力，并且分类效率远远高于人类专家。

常用的文本分类算法主要包括三大类。一类是基于概率和信息理论的分类算法,如朴素贝叶斯算法（Naive Bayes）,最大熵算法（Maximum Entropy）等；另一类是基于 TFIDF 权值计算方法的分类算法，这类算法包括 Rocchio 算法，TFIDF 算法，k 近邻算法（k Nearest Neighbors）等；第三类是基于知识学习的分类算法，如决策树（Decision Tree），人工神经网络（Artificial Neural Networks），支持向量机（Support Vector Machine），逻辑回归模型（Logistic Regression）等算法。

三、数据分析

在进行数据分类之前，首先我们对数据进行了分析。此次垃圾短信分类共有 80 万条标注数据，其中垃圾数据 80000 条，其余为非垃圾数据。由此可见数据中正负样本不均衡的问题非常严重。

考虑到上采样方法会导致过拟合，下采样会浪费过多的实验数据面对这样的情况，我们的思路是改变错分数据的代价。对不同的数据赋予不同的权重，使得不同类别的错分代价不同。我们对垃圾短信赋予更高的权重，使其在分类过程中被错分的代价更大，根据经验，我们将权重比例设置为 9:1。

四、研究方法

4.1 逻辑回归（Logistic Regression）

逻辑回归模型一般都是用来二分类情况的，只有 0，1 两类。定义一个样本 x_i 是类别 1 的概率为：

$$p(y_i=1|x_i)=\frac{\exp(w^T x_i)}{1+\exp(^T x_i)}=\frac{1}{1+\exp(-w^T x_i)}$$

该函数的输出值的取值范围为[0,1]。

所以记

$$h(x_i)=p(y_i=1|x_i)=\frac{\exp(w^T x_i)}{1+\exp(^T x_i)}=\frac{1}{1+\exp(-w^T x_i)}$$

逻辑回归模型的代价函数为：

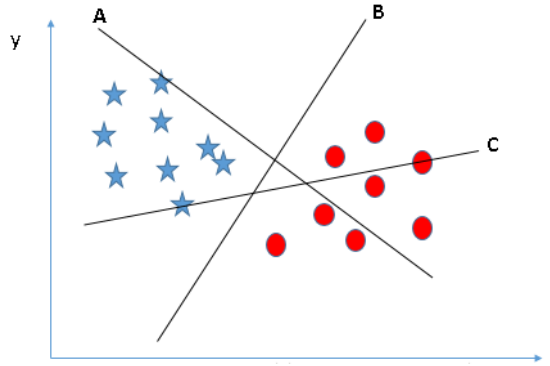
$$\begin{aligned}L(w)&=-\frac{1}{N}\log\prod_{i=1}^{N}h(x_i)^{y_i}(1-h(x_i))^{1-y_i}\\&=-\frac{1}{N}\sum_{i=1}^{N}[y_i\log h(x_i)+(1-y_i)\log(1-h(x_i))]\\&=-\frac{1}{N}\sum_{i=1}^{N}[y_i(w^T x_i)-\log(1+\exp(w^T x_i))]\end{aligned}$$

然后用梯度下降就可以求得 w，带入最开始的概率公式，只要概率大于 0.5，就可以划分到类别 1。

4.2 支撑向量机（SVM）

SVM 的核心思想是求出一组权重系数，在线性表示之后可以进行分类。我们先使用一组训练集来训练 SVM 中的权重系数，然后可以对测试集进行分类。

说的更加高大上一些：SVM 就是先训练出一个分割超平面（separation hyperplane），该平面就是分类的决策边界，分在平面两边的是两个类别的数据。显然，经典的 SVM 算法只适用于两类分类问题，当然，经过改进之后，SVM 也可以适用于多类分类问题。



我们希望找到离分隔超平面最近的点，并确保它们离分隔面的距离尽可能远。这里点到分隔面的距离被称为间隔（margin）。我们希望这个间隔尽可能的大。支持向量（support vector）就是离分隔超平面最近的那些点，我们要最大化支持向量到分隔面的距离。那么为了达到上面的目的，我们就要解决这样的一个问题：如何计算一个点到分隔面的距离？这里我们可以借鉴几何学中点到直线的距离，在 SVM 中，需要计算的是点到超平面的距离：

$$\frac{|ax_0+by_0+C|}{\sqrt{A^2+B^2}} \Rightarrow \frac{|w^T x+b|}{\|w\|}$$

除了定义距离之外，需要设定两类数据的类别标记分别为 1 和-1，这样做是因为我们将标记 y 和距离相乘，不管属于哪一类，乘积都是一个正数，这样有利于我们设计目标函数。这样我们便可以定义我们 SVM 的目标函数：

$$\arg\max_{w,b}\left\{\min_{n}(label\cdot(w^T x+b))\cdot\frac{1}{\|w\|}\right\}$$

然后我们可以转化为凸二次规划：

$$\min\frac{1}{2}\|w\|^2 \quad s.t.y_i(w^T x_i+b)\geq 1, i=1,...,n$$

构造拉格朗日函数：

$$\Phi(w,b,a)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{n}\alpha_i(y_i(w^T x_i+b)-1)$$

通过 KKT 条件和对偶问题等方法就可以求解了。

4.3 决策树（Decision Tree）

决策树（Decision Tree）是是分类问题中最常用的模型之一，它的优势在于：①能够接受类别型的特征，②分类效果与其他分类算法相当，③训练和测试的效率高。决策树模型呈树形结构，在分类问题中，表示基于特征对实例进行分类的过程。决策树学习通常包括三个步骤：特征选择、决策树的生成和决策树的修剪。

决策树学习常用的算法有 ID3、C4.5 和 CART。

决策树在构建子节点时需要选择合适的特征进行决策，划分数据，生成子节点。“合适”指的是尽量大的减少划分后子数据集的混杂度（尽可能使同一个类别的数据分到同一个子节点中）。ID3 算法使用信息增益来进行特征选择，C4.5 使用信息增益率进行特征选择，C&RT 作为分类树时使用基尼系数进行特征选择，作为回归树时使用最小平方误差进行特征选择。

决策树的生成采用贪心策略，从根节点开始扩张（自顶向下），已建立的树不再改变。每次选择局部信息增益（率）最大的特征划分数据，只考虑当前节点之后选择哪个特征来建立子节点。对子节点递归调用以上方法，直到达到停止条件（所有特征的信息增益都很小或者没有特征可以选择，early-stopping）停止建树。

在垃圾文本识别的问题中，将文本类的数据转化为词向量进行计算，因而需要对数值型的数据进行处理。决策树对于数值型的特征的处理办法通常是将特征离散化，寻找阈值，将数值划分为一个或多个区间。

4.4 梯度提升决策树（GBDT）

GBDT（Gradient Boosting Decision Tree）又叫 MART（Multiple Additive Regression Tree），属于集成学习中的 boosting 分支，是一种迭代的决策树算法，该算法由多棵决策树组成，所有树的预测加权累加起来做最终预测。

决策树（DT），分为分类树和回归树两种。梯度提升决策树的决策树主要是针对回归树的，回归树是可以用于回归的决策树模型，一个回归树对应着输入空间（即特征空间）的一个划分以及在划分单元上的输出值。与分类树不同的是，回归树对输入空间的划分采用一种启发式的方法，会遍历所有输入变量，找到最优的切分变量和最优的切分点，即选择特征和它的取值将输入空间划分为两部分，然后重复这个操作。

提升（B），即在原来模型的基础之上做进一步提升，提升决策树 BDT 的基本思想是采用多棵决策树串行建模。具体过程为，对于第一棵树之后的每一棵决策树，都基于前一棵决策树的输出进行二次建模，整个串行建模过程相当于对预测结果朝目标值进行修正。

梯度（G）。梯度的大小反映了当前预测值与目标值之间的距离。因此，上面 B 所述的串行决策树模型，除开第一棵决策树使用原始预测指标建树，之后的每一棵决策树都用前一棵决策树的预测值与目标值计算出来的负梯度（可以理解为残差或者增量）来建树。这相当于给分错的样本加权多次分类，使样本最终的残差趋近于 0。除开第一棵树的其他树，由于都是对目标的残差或增量进行建模预测，因此 GBDT 模型只需把过程中每一棵决策树的输出结果累加，便可得

到最终的预测输出。

GBDT 分类算法在思想上和回归算法没有区别，但是由于样本输出不是连续的值，而是离散的类别，导致无法直接从输出类别去拟合类别输出的误差。为解决此问题，使用类似于逻辑回归的对数似然损失函数的方法,也就是说用的是类别的预测概率值和真实概率值来拟合损失函数。

对于二元 GBDT，如果用类似于逻辑回归的对数似然损失函数，则损失函数表示为

$$L(y, f(x)) = \log(1+\exp(-yf(x)))$$

则此时的负梯度误差为

$$r_{ti} = -[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}]_{f(x)=f_{t-1}(x)} = \frac{y_i}{1+\exp(y_i f(x_i))}$$

对于生成的决策树，各个叶子结点的最佳残差拟合值为

$$c_{tj} = \arg\min \sum_{x_i \in R_{tj}} \log(1+\exp(-y_i(f_{t-1}(x_i)+c)))$$

由于上式比较难优化，一般使用近似值代替，近似值为

$$c_{tj} = \frac{\sum_{x_i \in R_{tj}} r_{ti}}{\sum_{x_i \in R_{tj}} |r_{ti}|(1-|r_{ti}|)}$$

除了负梯度误差计算和各个叶子结点的最佳残差拟合外，二元 GBDT 分类和 GDBT 回归算法过程相同。

五、实验设计

5.1 逻辑回归（Logistic Regression）模型

5.1.1 逻辑回归分类器训练

要做特征提取，首先要进行分词，这里使用了 jieba 精准分词模式。接下来使用 TFIDF 方式来构建词向量。

```
vectorizer = TfidfVectorizer()#实例化
classifier=LogisticRegression()
x_train = vectorizer.fit_transform(Y1)#这里计算的tfidf权重
```

在 sklearn 中，与逻辑回归有关的有三个类。LogisticRegression，LogisticRegressionCV，Logistic_regression_path。LogisticRegression 需要每次自己指定正则化系数，而 LogisticRegressionCV 使用了交叉验证来选择正则化系数。Logistic_regression_path 它拟合数据后不能直接用来做预测，而是用来选择合适的回归系数和正则化系数，主要用在模型选择的时候。

本实验使用的是 LogisticRegression。相应的参数调优为：

1、正则化系数选择 L2。

2、Solver 优化参数选择开源的 liblinear 库实现，内部使用了坐标轴下降法来迭代优化损失函数。

3、分类方法选择参数 ovr，即无论多少元逻辑回归，我们都可以看做二元逻辑回归。

4、类型权重参数采用默认值。

5.1.2 实验结果及分析

本实验通过分类的准确度，召回率，f1 值对模型进行评估。

在没有考虑交叉验证的情况下，结果如表 1 所示：

表 1 LR 模型分类结果（无交叉验证）

	Precision	Recall	F1-score
0	0.99	1.00	0.99
1	0.97	0.93	0.95

可以看到分类效果比较理想，但是因为垃圾短信类型分布不平衡，即使将短信全部分类为正常短信也能获得 0.9 以上的正确率。所以后期可以修改垃圾短信和正常短信的权重，来达到更好的分类效果。

考虑了交叉验证（5 折）的分类结果如表 2 所示：

表 2 LR 模型分类结果（五折交叉验证）

	Precision	Recall	F1-score
1	0.98396488	0.95705903	0.97002055
2	0.98321227	0.95694792	0.96961127
3	0.98328924	0.95963542	0.97108295
4	0.98345064	0.95965278	0.97116779
5	0.98493729	0.95808681	0.9710238
Average	0.9837	0.9583	0.9706

交叉集的验证结果一定程度反映了分类器真实的预测能力。从表 2 中可以看出模型达到了初步的分类效果。

在测试集上我使用了五万条数据用来测试（之前没有被用于训练），最终的精确率为 0.98，召回率为 0.95，f1 值为 0.97，该结果与训练集比较接近。

5.2 支持向量机（Support Vector Machine）模型

5.2.1 数据预处理

首先将短信和标签分别提取出来并存储，然后利用 jieba 对短信内容进分词，提取文本特征，也就是将文本数据转化成特征向量的过程。比较常用的文本特征表示法为词袋法：不考虑词语出现的顺序，每个出现过的词汇单独作为一列特征。这些不重复的特征词汇集合为词表，每一个文本都可以在很长的词表上统计出一个很多列的特征向量。如果每个文本都出现的词汇，一般被标记为停用词，不计入特征向量。本模型并没有去除停用词。

5.2.2 训练模型

通过上面的过程，我们准备好了待训练的数据和训练需要的参数，其实可以理解为这个准备工作就是在为 svm.train()函数准备实参的过程。来用 svm.train()函数进行训练，并使用 joblib 保存模型：

```
self.clf.fit(self.training_data, self.training_target)
joblib.dump(self.clf, 'model/SVM_sklearn.pkl')
```

5.2.3 模型评估

这里用的混淆矩阵来评估模型，表 3 为 SVM 模型分类结果：

Confusion matrix	
8950	24
5	968

表 3 SVM 模型分类结果

	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	80901
1	0.99	0.98	0.99	9099
Micro avg	1.00	1.00	1.00	90000
Macro avg	1.00	0.99	0.99	90000
Wighted avg	1.00	1.00	1.00	90000

可以看到，即使训练部分数据时，效果也很出色，符合预期值。SVM 是一种有坚实理论基础的新颖的小样本学习方法。它基本上不涉及概率测度及大数定律等，因此不同于现有的统计方法。从本质上看,它避开了从归纳到演绎的传统过程，实现了高效的从训练样本到预报样本的“转导推理”，大大简化了通常的分类和回归等问题。

5.3 决策树（Decision Tree）模型

5.3.1 决策树分类器训练

1、数据划分

为了对训练的模型的效果进行有效测试，将数据集划分为训练集和验证集两个部分，使用 sklearn 的 StratifiedKFold()函数将数据分成五个部分，每次随机选取四个部分作为训练集，另一部分作为验证集。

```
if offline:

    xx_score = []

    skf = StratifiedKFold(n_splits=5, random_state=1126, shuffle=True)
    X = np.array(X)
    Y = np.array(Y)

    for index, (train_index, test_index) in enumerate(skf.split(X, Y)):
        X_train, X_valid, y_train, y_valid = X[train_index], X[test_index], Y[train_index], Y[test_index]
```

2、特征提取

使用 python 的 jieba 工具包来进行分词的操作，对全部垃圾短信数据进行精准模式分词，分词后使用 TF-IDF 算法来构建短信内容的特征向量。

在 sklearn 工具包中，TfidfVectorizer 类可以用来计算词语的 TF-IDF 值。对训练集和验证集分别做处理，得到每一条短信的 TF-IDF 特征。

```
vectorizer = TfidfVectorizer()
tfidf = vectorizer.fit(X)
X_train = vectorizer.transform(X_train)
X_valid = vectorizer.transform(X_valid)
print('---tfidf finish---\n')
```

3、模型训练

在 sklearn 工具包中，使用 DecisionTreeClassifier()来实现决策树模型，并设定特征选择方法为基尼系数，并将分割策略设置为 best，为了防止过拟合，在一个节点中含有少于 2 个样本的情况下停止分割。

4、应用到系统中的模型

使用全量数据来训练线上模型，并使用 joblib 保存生成的 TF-IDF 特征数据及决策树模型数据。

```
if online:
    vectorizer = TfidfVectorizer()
    tfidf = vectorizer.fit_transform(X)
    joblib.dump(vectorizer, 'tfidf.m', protocol=2)
    print('--tfidf finish--')

    cls.fit(tfidf, Y)
    cls_rf.fit(tfidf, Y)
    joblib.dump(cls, 'dtree.m')
    joblib.dump(cls_rf, 'rf.m')
```

5.3.2 实验结果及分析

使用 5 折交叉验证来进行评估模型，使用准确率、召回率和 F1-score 作为评价指标，测试的各项指标的结果如表 4 所示。

表 4 决策树模型分类结果

	Precision	Recall	F1-Score
1	0.975948473	0.954100694	0.964900927
2	0.974029417	0.953736111	0.963775952
3	0.976212743	0.953361111	0.964651613
4	0.974507729	0.953420139	0.963848606
5	0.974512832	0.951822917	0.963034244
Average	0.975042239	0.953288194	0.96404251

从实验结果可以看出，决策树模型对于垃圾短信的分类有良好的效果，但仍有很大的提升空间，在大规模数据的情况下，仍会有大量的垃圾短信不能被正确识别。后期考虑对于垃圾文本的内容特征进行进一步提取，如考虑文本中的情感特征，特殊字符数量等，进行进一步分析。

5.4 梯度提升决策树（GBDT）模型

5.4.1 GBDT 分类器训练

1、特征生成

首先使用 jieba 分词库对所有短信数据进行精准模式分词，得到分词后的短信集合。由于单纯统计词频会导致一些几乎在所有短信中都出现过的词的重要性非常高，而这些词在对短信是否是垃圾短信的分类上是完全不具有可判别性的特征，相比之下，那些在一条短信中词频高而在其他短信中出现频率很少的词汇对判别这条短信是否是垃圾短信更具有重要的判别性。所以，使用 TF-IDF 公式生成的词语的特征更适合用来作为垃圾短信分类的特征。在 scikit-learn 库中存在 TfidfTransformer 类用来计算该特征，通过调用 fit_transform 函数将所有分词后的短信数据转换为 TF-IDF 特征，同时保存下来特征文件。

2、模型训练

在 scikit-learn 库中，针对梯度提升决策树算法的建模，在 sklearn.ensemble 中存在 GradientBoostingClassifier 这个类，使用该类创建梯度提升决策树模型，直接使用默认配置参数对生成的 TF-IDF 短信数据特征进行拟合，即可得到最终的 GBDT 分类器模型，将生成的模型保存供垃圾短信分类测试使用。

5.4.2 实验结果及分析

对训练的分类器采用 5 折交叉验证去评估模型的实际效果，评价指标选择的

是准确率（Precision_macro）、召回率（Recall_macro）和 F 度量（F1-score_macro），测试的各项指标的结果如表 5 所示。

表 5 GBDT 分类结果

Metric	Precision	Recall	F1-Score
1	0.97670188	0.88220139	0.92291139
2	0.97676571	0.88470833	0.92454531
3	0.9716426	0.88649653	0.92373754
4	0.978278	0.88268	0.923806
5	0.9794757	0.8881007	0.9277328
Average	0.976572786	0.88483739	0.924546608

从表 5 中可以看出，GBDT 算法的预测准确率是比较高的，但是召回率相比之下就比较低了，这导致了 F1-Score 结果也不是很高。初步分析，使用默认配置的 GBDT 算法进行模型训练的垃圾短信预测效果一般，后期可以考虑对其它的配置进行实验。首先，增加“子模型数”用来降低整体模型的方差，且不会对子模型的偏差和方差有任何影响，模型的准确度一般会随着“子模型数”的增加而提高；其次，尝试调整“最大树深度”、“最大叶结点数”、“分裂所需最小样本数”、“叶结点最小样本数”等超参数细粒度地调整决策树的结构，防止模型过拟合现象的发生，提高模型的泛化能力和垃圾短信分类能力。

5.5 垃圾短信识别系统

我们线上的垃圾短信识别系统地址为 http://spamclassify.ml/ 图 1 为系统界面。

图 1 线上垃圾短信识别系统界面

系统可以对在输入框中输入的垃圾短信进行识别，在下方显示各个模型的识别结果，并统计每个模型识别该条短信所用时间。识别结果如图 2、图 3 所示。

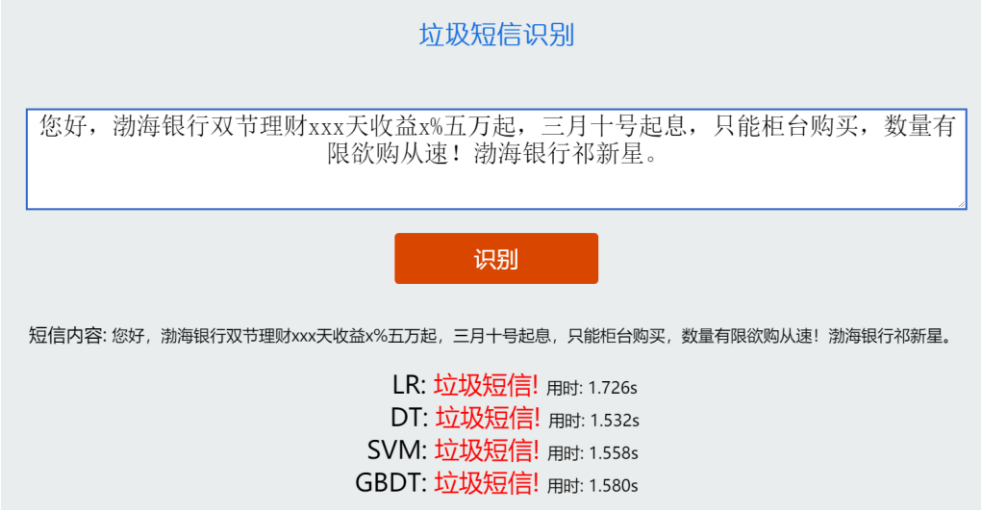


图 2 系统对短信进行识别结果为垃圾短信

图 3 系统对垃圾短信进行识别结果为非垃圾短信

六、总结及分工

在本次实验中，通过垃圾短信进行识别，对数据挖掘的常用方法有了较为全面的认识，能够使用 python 对各类算法进行实现和应用。在本次实验中，SVM 模型对于垃圾短信的分类效果最好。

小组分工如下（按姓名首字母排序）：

- 刘　浩：GBDT 模型的设计与实现及报告撰写，组织小组分工进行工作
- 梁付槐：SVM 模型的设计与实现及报告撰写，垃圾短信识别系统的设计与实现
- 骆　宁：决策树模型的设计与实现及报告撰写、汇总和排版，系统使用说明
- 任鸿飞：LR 模型的设计与实现及报告撰写，报告汇总及相关工作调研、

数据分析

参考文献

[1] 腾讯安全移动实验室 2018 年上半年手机安全报告.腾讯安全移动实验室.[EB/OL]. https://m.qq.com/security_lab/news_detail_471.html.2018-07-23.

[2] Maron M E . Automatic Indexing: An Experimental Inquiry[J]. Journal of the Acm, 1961, 8(3):404-417.

[3] 王爽. 基于知识库的自动分类系统设计与实现[D]. 厦门大学, 2007.