垃圾短信识别系统

> 组长：盛思远　成员：杨攀、张家华、刘志刚

Part 1 引言

垃圾短信识别的主要任务是根据一条短信的文本内容，判断是不是垃圾短信。这是一个典型的机器学习分类任务。既然是机器学习任务，自然就分为训练以及推理。训练会有一系列训练数据，训练数据是一条条短信文本内容以及对应短信是否为垃圾短信的标注。由于训练数据中给定了标注，所以这是监督学习任务。

完整的垃圾短信识别系统需要提供从模型训练到推理的一条龙服务。具体来说，训练数据需要支持动态添加，模型也需要周期性地重新训练。另外，系统需要支持实时地在线推理。在满足这些基本的功能要求后，系统需要做到可扩展，高可靠，高易用性。

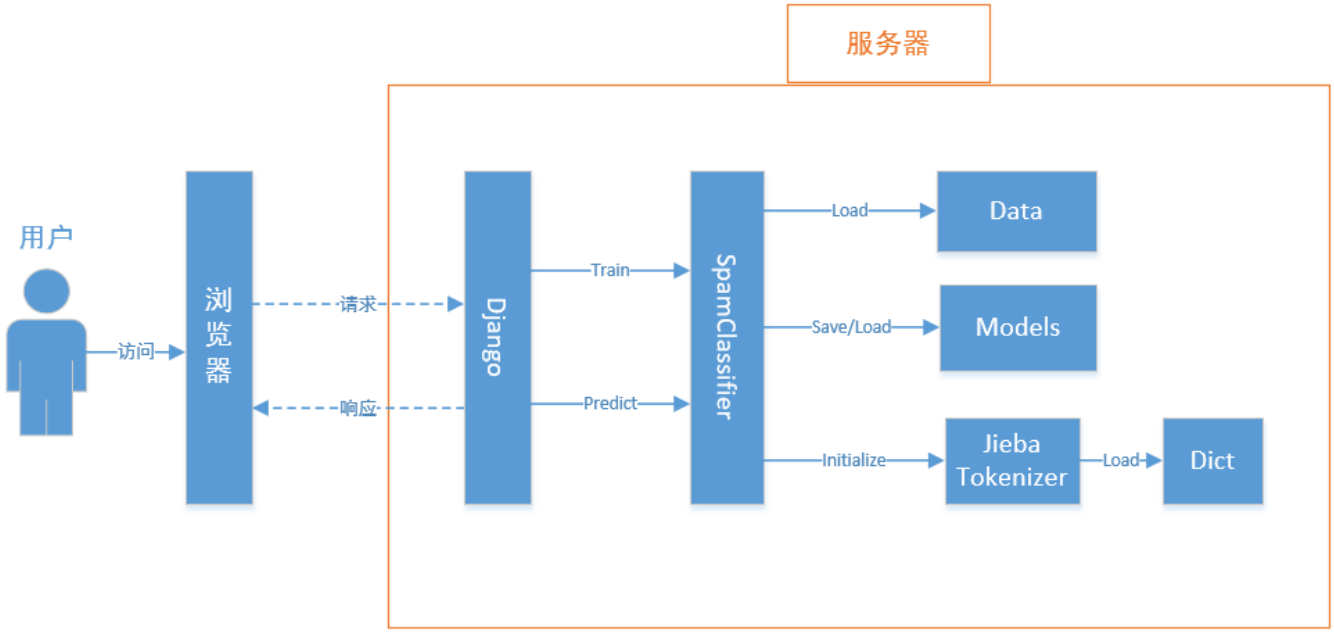
要做出一个垃圾短信识别系统，面临如下挑战：

1. 垃圾短信是文本内容，不能直接使用机器学习算法，需要进行合适的预处理
2. 选取合适的机器学习算法
3. 如何使系统分布式，并且做到可扩展，高可靠，易用

本报告介绍了我们如何应对这些挑战，并且构建出一套完整的垃圾短信分类系统。

Part 2 整体架构介绍

首先是整体架构设计。为了支持从模型训练到推理的一条龙服务，我们设计了如下的系统：



如图所示，垃圾短信识别系统主要分为前端与后端部分。

1、前端

前端Django负责处理用户的请求，对外提供网络服务，用户通过浏览器等终端进行访问。用户可以选择导入、更新训练数据，或者在线对一条短信进行分类。

前端部分的代码对应OfflineDemo文件夹，前端页面等静态资源存在于SpamClassierVisualization下的static文件夹和templates文件夹。而后端的请求处理代码则存在于polls下的urls.py、views.py文件中。

2、后端

后端部分负责处理模型的训练以及推理。其中SpamClassier是核心，它对后端事先准备的训练数据进行预处理，将结果保存到磁盘上，安排离线训练；同时，接受前端的分类请求，从磁盘中加载预先训练好的模型，利用模型，在线推理。

后端部分的代码对应SpamClassifier文件夹，原始数据及预处理数据保存在Data文件夹下，导出模型保存在Models文件夹下。机器学习相关的核心代码都在SpamClassifier.py文件中。

Part 3 具体实现

下面我们介绍系统的每部分是如何实现的，具体用到了什么框架，什么技术。

1、前端展示

前端部分，我们使用了python的django web框架，来为用户提供轻松易用的网页界面，用户通过浏览器，完成一键短信分类、算法性能比较等功能。

2、预处理及数据切分

数据预处理以及分词部分，使用了python的相关库进行预处理，使用jieba库进行中文文本分词。

首先是去除无关字符等基本的预处理。

再接下来对文本的处理，我们基于词袋模型，也就是我们只关心一个文本中包含多少词以及各个词的频率，而对词之间的语法结构与语义不关心。

但是要知道文本中包含什么词，我们首先需要进行分词。对于英文而言，利用其单词与单词间的空格可以很容易地分割单词，但是中文就得用一套算法进行分词。这里我们使用的是较为成熟的第三方分词工具jieba。

jieba会加载预先训练好的词典，基于词典对中文的句子进行分词。将每条短信看作一个text，则分词的结果就是list of words or terms。然后我们可以统计term-frequency，得到tf向量。再计算出tf-idf向量，这样子，短信文本就变成了向量，就可以用机器学习处理了。

因为我们使用的是分布式的机器学习框架，每次做训练或者预测时，都需要启用新的jvm进程，则需要重新加载词典用于分词，这个开销是比较大的。为了提高效率，我们在预处理的最后阶段，会将分词后的结果直接切分成五份保存到磁盘中，避免每次训练或预测都要做重复性的工作带来额外的开销。

切分成五份的目的是为了使用交叉验证法。这里将切分后的数据保存下来，可以保证不同算法间使用同样的训练集和测试集，使得模型测试结果具有可比性。

3、分类算法

机器学习算法的实现部分，我们使用了Spark ML库实现了如下算法，下面简单介绍一下每种算法。

（1）Multi-layer Perceptron

多层感知器（Multilayer Perceptron,缩写MLP）是一种前向结构的人工神经网络，映射一组输入向量到一组输出向量。MLP可以被看作是一个有向图，由多个的节点层所组成，每一层都全连接到下一层。除了输入节点，每个节点都是一个带有非线性激活函数的神经元（或称处理单元）。一种被称为反向传播算法的监督学习方法（BP算法）常被用来训练MLP。MLP是感知器的推广，克服了感知器不能对线性不可分数据进行识别的弱点。

（2）Linear SVM

支持向量机（SVM）是一种监督学习模型，可以分析数据，识别模式，用于分类和回归分析。给定一组训练样本，每个标记为属于两类，一个SVM训练算法建立了一个模型，分配新的实例为一类或其他类，使其成为非概率二元线性分类。SVM使得不同类别的训练样例再超平面中的距离尽可能大，继而提供了较好的泛化能力。

（3）Binomial Logistic Regression

逻辑斯蒂回归是在线性回归的基础上套用了一个逻辑斯蒂函数。它很好地解决了线性回归鲁棒性差的问题。

（4）Decision Tree

决策树是一个预测模型；它代表的是对象属性与对象值之间的一种映射关系。树中每个叶节点表示某个或某类对象，而每个分叉路径则代表某个可能的属性值。决策树具有训练速度快，可解释性好的特点。

（5）GBDT

GBDT是一种Boosting算法，但是与传统的Adaboost不同。Adaboost，利用前一轮迭代弱学习器的误差率来更新训练集的权重，并且一轮轮的迭代下去。GBDT使用了前向分布算法，以及CART回归树模型，同时迭代思路和Adaboost也有所不同。GBDT具有预测精度高，适合低维数据的优点。

4、分布式

我们使用Spark框架来解决分布式的相关问题。

Spark 是专为大规模数据处理而设计的快速通用的计算引擎。它是开源的类Hadoop MapReduce的通用并行框架，拥有Hadoop所具有的优点；但不同于Hadoop的是，Spark任务中间输出结果可以保存在内存中，不再需要读写HDFS，从而减少了中间过程磁盘IO的开销，提高了性能。同时，相比于传统的map-reduce编程范式，Spark支持更多操作，因而能更好地适用于数据挖掘与机器学习等需要迭代的算法。Spark解决了与分布式相关的性能、可靠、易用性等问题。

Spark ML通过数据并行，进一步提升了模型训练的效率。同时最新版本的Spark ML通过Dataframe取代了最初RDD的实现，分布式计算、序列化反序列化等方面的性能得到了优化。由于我们的数据量偏大，而模型属于中小型，因此非常适合Spark ML的分布式学习的方式。

Part 4 模型训练

1、训练集划分

每个算法在训练过程中使用了5折交叉验证。每次使用其中一份作为测试集，剩余四份作为训练集。这样每种算法可以得到五个模型。

2、处理过大的词典

在分词后，每条短信（document）已经被分成了一串词（term）。而要统计term-frequency，我们必须要使用一个全局的词典，记录每个term总共出现了多少次。

由于term很多，词典也可能很大（假设其大小为N）。为了解决这个问题，我们使用hash以及counting bloom filter进行统计。具体方法是用户指定总共有多少个hash bucket（假设为M个），然后对于每一个term，我们将它hash到某个bucket，并且将这个bucket计数值加1。这样，我们将N个term，映射为了M个“term”，最后输出的是这M个“term”的“term”-frequency。将词典大小从N缩小到M，并进行统计。

这种映射也可以看做是一种降维。例如document用原有的大小为N的词典来进行tf-idf向量化表示的话，则有N维，而进行降维后，将只有M维，这极大地降低了训练的复杂度。

虽然我们通过近似计算的方式损失了一定程度上的精度，但获得了时间和空间效率的极大提升。实际上，如果选用合适的哈希函数，比如LSH，我们可以认为哈希到同一个桶里的单词具有某种程度的相似性，并不会对最终模型的性能和效果产生太大的影响。

3、向量化

对于每个短信分词后生成的word list，通过遍历可以得到每个词在该短信中的tf值。

再遍历整个训练集，我们可以得到每个词一共在多少个短信中出现过，即df值。

通过对df做discount，我们得到了最终的tf-idf向量。这样就转化成了最常见的有监督学习的分类任务。通过采用前面提到的不同的分类算法，完成整个模型的训练过程。

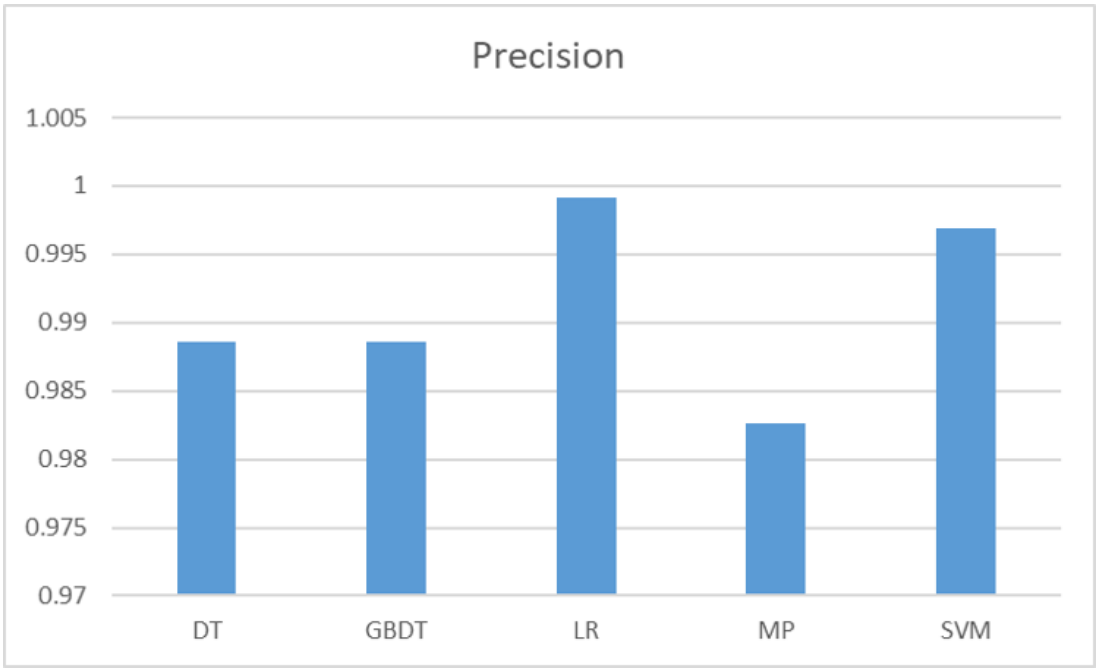
Part 5 模型评估与实验

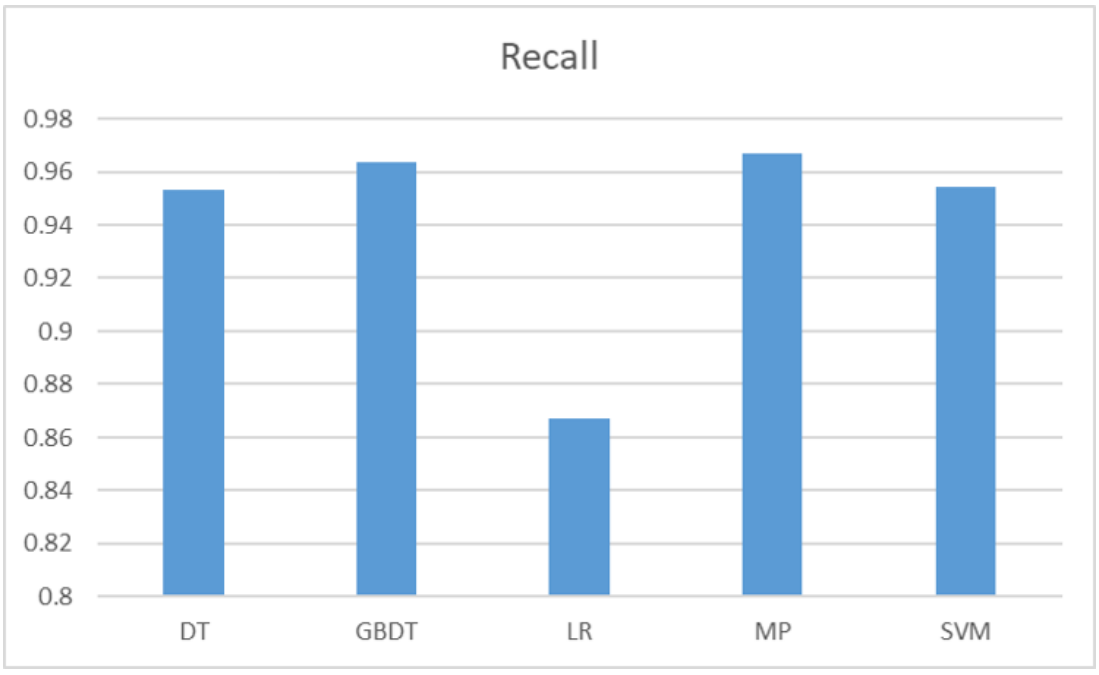
1、实验环境

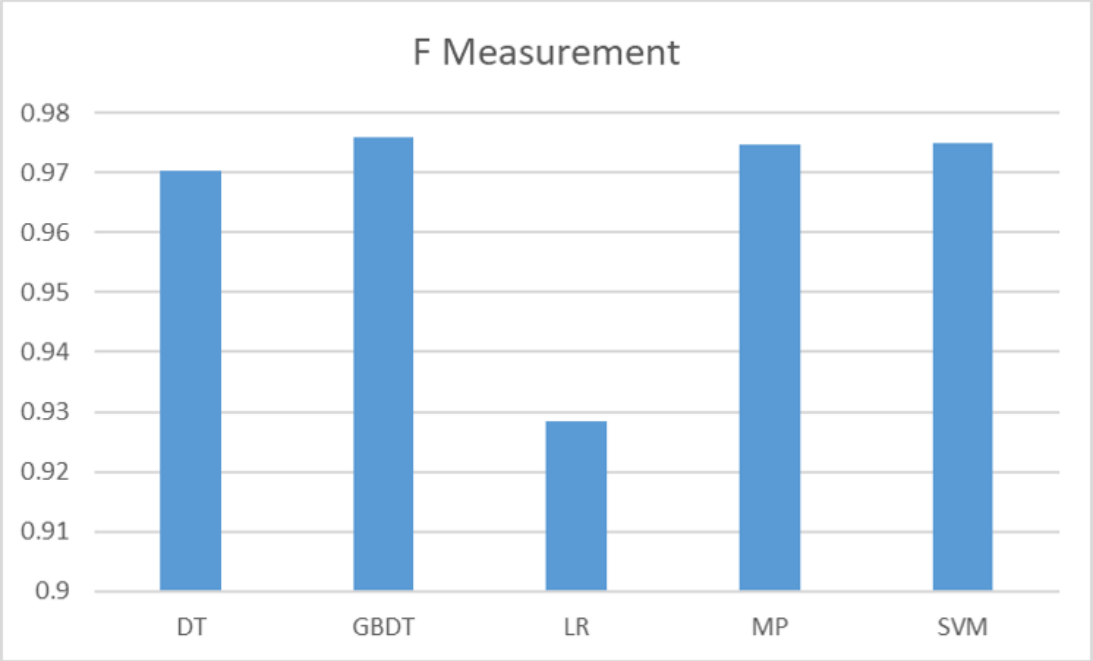
使用型号为RH2288H V3的服务器，有62.5G的内存和900G不到的磁盘空间，10颗型号为Intel(R) Xeon(R) CPU E5-2630 v4的CPU，每颗CPU有四个核，每个核支持两个线程。训练过程中Spark ML的配置为：最大进程数为16，每个jvm的最大运行内存为16G。

2、实验结果及分析

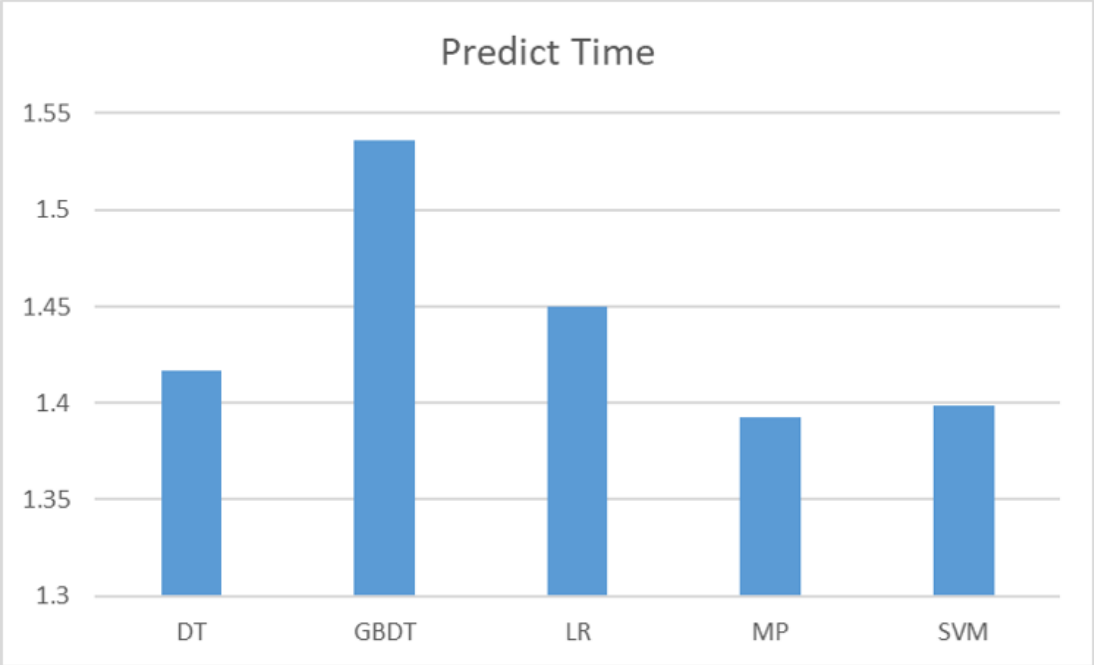
在测试给定的某种算法时，我们采用了Precision、Recall进行了模型评估。对每个模型使用相应的测试集进行测试，通过平均值来衡量算法的分类效果。光使用Precision和Recall的评价都较为片面，因此我们考虑综合的指标F Measurement。从下面的图来看，除了Logistic Regression，其他算法都得到了不错的效果。







在预测时，对于给定的某种算法，使用离线训练好的五个模型vote之后的结果作为最终的预测结果返回给前端。下图是模型预测的时间。



从预测时间上来看，GBDT的时间较长，因此综合时间开销与预测精度来看，DT、MP、SVM都是较好的选择。

实际上GBDT的预测时间过长与其模型复杂度过高有关，它也是这五个算法当中训练时间最长的一个。而关于LR的Recall，可能是因为LR找到的分隔面容错性不足。而其他算法，比如SVM通过最大化边距，得到的模型泛化能力要强于LR，因此也得到了更高的Recall和F Measurement。

3、可能的改进方案

对于训练时间过长的分类算法，经分析可能是由于模型本身的复杂程度导致的。我们可以考虑引入模型并发的方式，通过类似Parameter Server的架构，来提高模型训练的效率。

对于模型精度不高的分类算法，可以考虑使用Adaboost等方式，将弱分类器混合或级联在一起，提高整体的性能与分类效果。

Part 6 相关工作及总结

1、相关工作

垃圾短信分类本质上就是一个二分类问题，与之类似的问题有很多。我们以文本分析和图像分类两个例子，来比较一下它们的异同点。

与文本分析相比，二者都涉及到一些自然语言处理的知识。不同之处在于，短信分类可能只需要用到词袋模型，进行简单的二分类。而文本分析可能需要用到上下文相关的信息，利用词项的分布式表达，进行复杂的分类、话题建模等。

与图像分类相比，二者都是有监督的分类问题。不同之处可能在于，短信分类更多的是通过文本处理的方式进行向量化，而图像分类则需要使用卷积、池化等图像相关的特定操作得到向量化的数据。

2、团队分工

盛思远: 数据处理与模型训练

杨攀: 前端展示与演示demo

刘志刚: 项目报告

张家华: 实验及分析

3、成果

我们提交的内容有：

1. 完整的前后端源代码(OfflineDemo, SpamClassifier)
2. 中间数据与导出模型(Data, Models)
3. 录制的演示视频(display-demo.mp4)
4. 项目使用说明(README.md)

你也可以从这里获得完整的项目：https://github.com/LGN520/SpamClassifier :)