

携程Spark算法平台及其应用

2016-12-10

携程旅游网成立于1999年，总部设在上海，
目前有员工**30000**余人

2003年12月9日在美国纳斯达克成功上市

携程拥有超过**2.5亿**的注册会员

酒店预订：在全球**200多个国家**和地区拥有
超过**120万家酒店**的会员酒店

机票预订：产品覆盖全球六大洲**5000多大**
中城市

旅游度假：线路产品覆盖超过100多个目的
地国家和地区；2015年大陆地区度假产品的
服务人次超过2000万



浙江大学本科， 硕士毕业

近10年工作经验， 5年大数据架构的经验

之前在eBay中国研发中心和大众点评工作过， 从0开始组建团队， 搭建起大众点评数据平台的基础架构

目前是携程的大数据平台总监

关注大数据系统领域的发展， 对Hadoop, HIVE, HBASE, Spark, Storm等有所研究， 致力于大数据系统和业务场景的结合和落地， 使数据和系统都能够对业务（Business）产生价值



背景介绍

算法平台的功能

算法平台的架构和实现

算法平台的应用

算法平台未来的方向

大背景：随着携程的业务发展，各个BU的机器学习的应用越来越多，训练的数据集也越来越大，单机的模型训练方式很难满足实际的需要，所以越来越多的同学会考虑使用Spark进行训练

直接使用Spark进行模型训练的难点（特别是对于非工程背景的数据科学家）：

- 入门时间较长
- 环境设置麻烦
- 程序调试困难
- 关注太多细节
- 独立开发，经验，代码无法分享

服务工程经验较少的数据科学家

- 无需编程，模块拖拽完成模型训练处理流程的构建
- 执行过程可视化
- 方便地数据探索功能

覆盖机器学习应用的整个生命周期

- 覆盖模型训练，导出，线上服务整个机器学习应用的生命周期

系统开放，高度可定制

- 简便地模块定制，分享（适合工程能力强的资深用户）

背景介绍

算法平台的功能

算法平台的架构和实现

算法平台的应用

算法平台未来的方向

整个平台由以下4个功能模块构成：

- 模型训练
 - 创建模型训练流程（Pipeline）
 - 运行和调试模型训练流程
 - 数据探索
- 模块定制（高级）
- 训练结果导出
- 线上服务的开发支持

使用协同过滤算法进行电影推荐 (Data from Netflix)

1. 训练数据 (用户对于电影的评分数据)

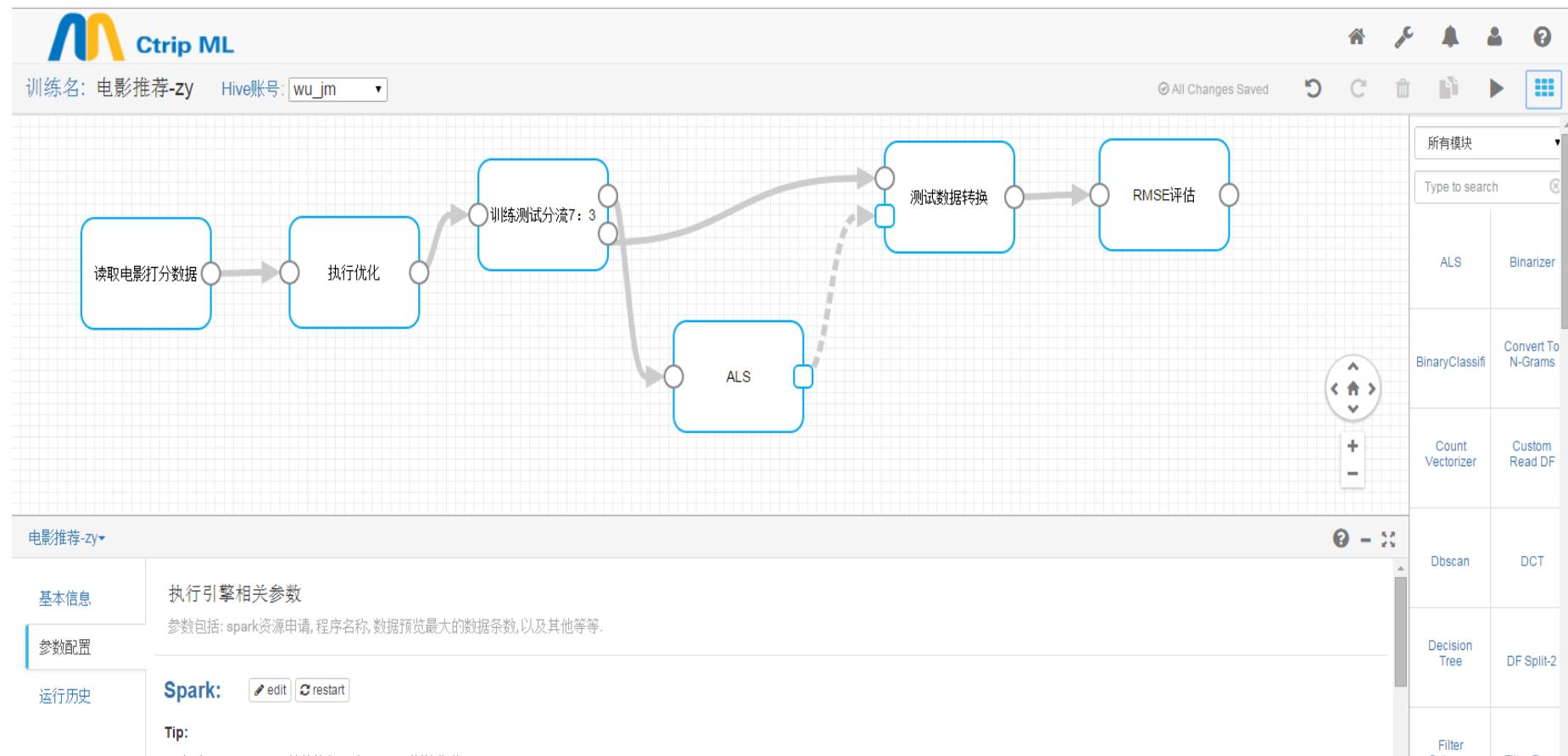
User Id	Movie Id	Rating	Timestamp
214549	68157	5.0	1293189830
214551	223	4.0	1281269370
214551	590	3.0	1281269251
214551	1198	5.0	1281268787

2. 分流 (训练集 / 测试集 7: 3)

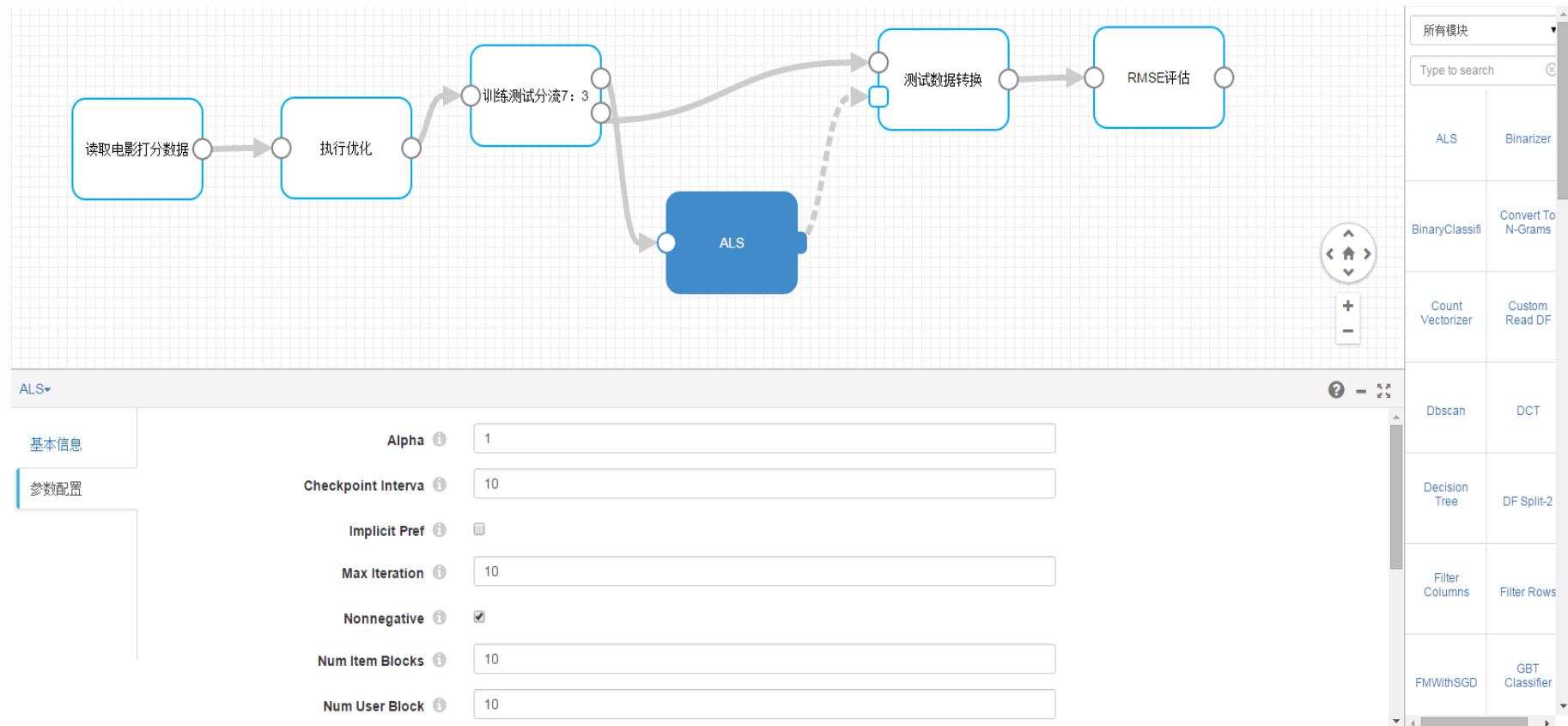
3. 模型训练

4. 模型检验

1. 点击相应模块，拖拽构成整个训练的流程

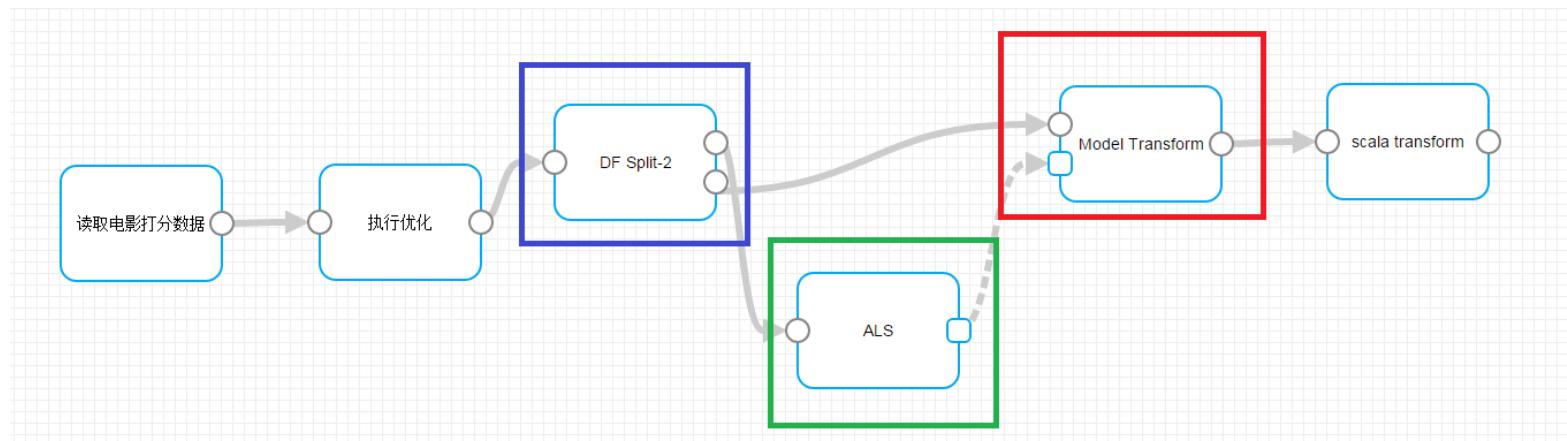


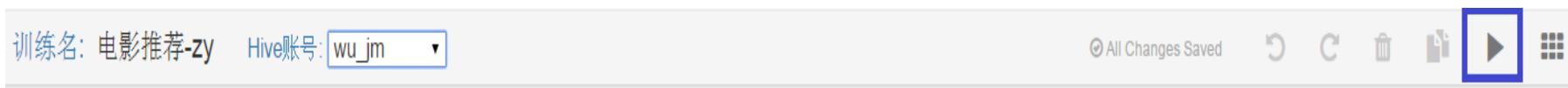
2. 填写相关模块的参数



模块分为两种类型：

- Transformer – 输入是1或多个DataFrame， 输出是1或多个DataFrame
- Model Trainer - 输入是1或多个DataFrame， 输出是一个Model
- 特殊模块 Model Transformer – 输入1个模型， 1或多个DataFrame， 输出1或多个DataFrame





Ctrip ML

训练名: 电影推荐-zy

训练测试分流 7: 3

基本信息

参数信息

参数配置

单点执行 (selected) | 级联执行 | 取消执行 | 数据探索

训练数据比例: 0.7

测试数据比例: 0.3

随机种子: 1

执行信息

开始时间: 2016-11-21 15:35:23
结束时间: 2016-11-21 15:35:24
执行用时: 1 seconds

执行输出:

```
customTransformedDF_01: org.apache.spark.sql.DataFrame = [userId: int, movieId: int, rating: double, timestamp: int]
splits: Array[org.apache.spark.sql.DataFrame] = Array([userId: int, movieId: int, rating: double, timestamp: int])
trainData_01: org.apache.spark.sql.DataFrame = [userId: int, movieId: int, rating: double, timestamp: int]
testData_01: org.apache.spark.sql.DataFrame = [userId: int, movieId: int, rating: double, timestamp: int]
```

Ctrip ML

训练名: 电影推荐-zy

```

graph LR
    A[读取电影打分数据] --> B[执行优化]
    B --> C{训练测试分流 7: 3}
    C --> D[ALS]
    C --> E[测试数据转换]
    D --> F[RMSE评估]
    E --> F
  
```

执行优化 -

基本信息	参数信息	执行信息
参数配置	<input type="checkbox"/> 单点执行 <input type="checkbox"/> 级联执行 <input type="checkbox"/> 取消执行 <input checked="" type="radio"/> 数据探索	开始时间: 2016-11-21 15:34:06 结束时间: 2016-11-21 15:35:23 执行用时: 77 seconds
	<pre> 1 def custom_transform(inputDF: DataFrame): DataFrame = { 2 // custom code ... 3 4 5 val pDF = inputDF.repartition(40) 6 pDF.cache() 7 pDF 8 } </pre>	执行输出: <pre> import org.apache.spark.sql.DataFrame custom_transform: (inputDF: org.apache.spark.sql.DataFrame)org.apache.spark.sql.DataFrame input_csv_data_01: org.apache.spark.sql.DataFrame = [userId: int, movieId: int, rating: double, timestamp: int] customTransformedDF_01: org.apache.spark.sql.DataFrame = [userId: int, movieId: int, rating: double, timestamp: int] </pre>

对于每个模块输出的DataFrame，我们都提供了数据预览和用户自定义SQL（SparkSQL）查询的功能

节点 Model Transform 的数据预览

基本预览 高级查询

User Id	Movie Id	Rating	Timestamp	Prediction
28032	31	2.0	845576381	3.1496027
81032	31	4.0	1139795208	3.5287213
82232	31	2.0	1112328270	2.7571738
89432	31	4.0	1421522468	3.3133657
89632	31	2.5	1428876096	2.3066936
93632	31	3.5	1144035625	2.936814
109632	31	4.0	1122139720	3.1650481
112232	31	5.0	848317639	3.7361348
164632	31	3.0	840788576	3.7190595
201632	31	3.0	1012703881	3.3234942
218632	31	4.0	1394658601	3.1955771
220032	31	3.0	1191291629	2.8859036
232432	31	0.5	1268919187	1.9686025
247232	31	4.0	875465387	3.2277513
18637	31	3.0	834461297	2.9800732
49437	31	3.0	1378230648	3.2970371
59237	31	2.0	844641625	2.9724126
64237	31	4.0	831055670	3.429466

Total Items: 200 (Showing Items: 18)

数据绘图：集成Pandas, matplotlib

节点 执行优化的数据预览

查询-0 查询-1 查询-2 **查询-3** +

图形 **数据**

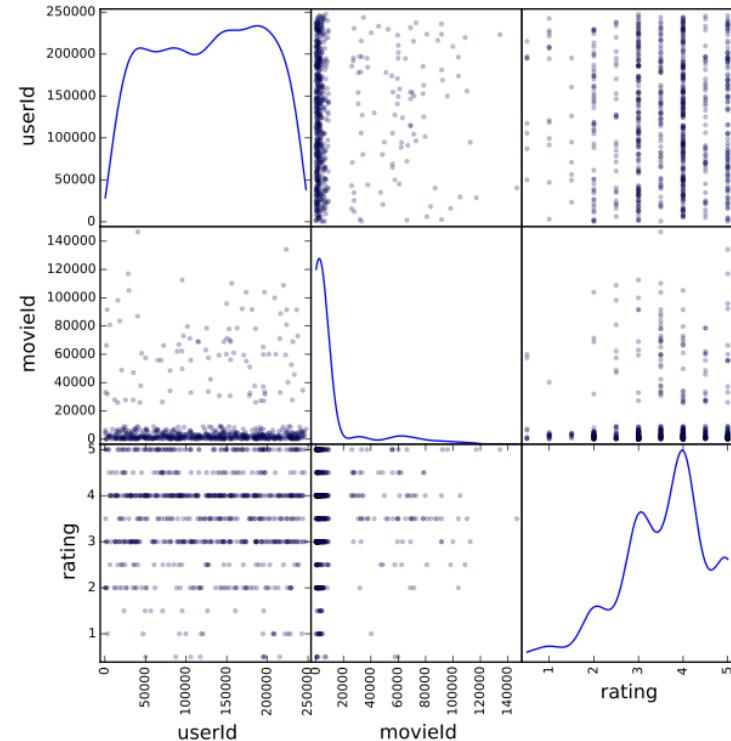
查询注释: | x

执行/取消 D W | x

```

1 %pyspark
2 #python数据探索主要是将spark DataFrame的数据collect到Driver端,
#然后进行计算，工具可以使用python的pandas，API可以点击页头的链接
3
4 import matplotlib
5 import pandas as pd
6 matplotlib.use('Agg')
7
8 import numpy as np
9 import matplotlib.mlab as mlab
10 import matplotlib.pyplot as plt
11 import StringIO
12 from pandas.tools.plotting import scatter_matrix
13
14 #这是必须的，否则图表不显示
15 def show(p):
16     img = StringIO.StringIO()
17     p.savefig(img, format='svg')
18     img.seek(0)
19     print "%html <div>" + img.buf + "</div>"
20     plt.close()
21
22 #从临时表中获取DataFrame
23 #dfName = sqlContext.table("dfName")
24 customTransformedDF_01 = sqlContext.table("customTransformedDF_01").select("userId","movieId","rating")
25
26 #print customTransformedDF_01.count()
27
28 sampleData = customTransformedDF_01.sample(False, 0.00003, 7)
29
30 #spark DataFrame转换成Pandas DataFrame，然后通过pandas api进行科学统计或可视化
31 pdDF=sampleData.toPandas()
32
33 #print pdDF
34 # python 计算与画图code
35 #pdDF["rating"].plot.area();
36 plt.figure()
37 #pdDF.diff().hist(color='k', alpha=0.6, bins=50)
38 scatter_matrix(pdDF, alpha=0.2, figsize=(7, 7), diagonal='kde')
39
40 show(plt);

```



基础信息

模块名称	ALS
模块描述	推荐模型
模块归属	推荐
共享级别	全局共享

代码模板

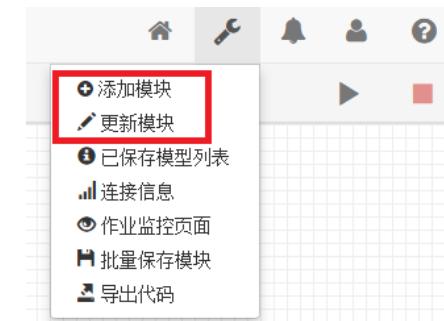
```

1 import org.apache.spark.ml.recommendation.ALS
2 val $inputDF$ = sqlContext.table("$inputDF$")
3 val als = new ALS()
4   .setAlpha($alpha$)
5   .setCheckpointInterval($checkpointInterval$)
6   .setImplicitPrefs($implicitPref$)
7   .setMaxIter($maxIter$)
8   .setNonnegative($nonnegative$)
9   .setNumItemBlocks($numItemBlocks$)
10  .setNumUserBlocks($numUserBlocks$)
11  .setRank($rank$)
12  .setRegParam($regParam$)
13  .setSeed($seed$)
14  .setRatingCol("ratingCol$")
15  .setItemCol("itemCol$")
16  .setUserCol("userCol$")
17  .setPredictionCol("predictionCol$")
18 val $outputMDS$ = als.fit($inputDF$)

```

配置信息

Key	Key显示名	描述信息	参数类型	默认参数值	可选选项	参数归属
inputDF	输入训练数据		STRING		格式: v1,v2...	输入!
alpha	Alpha		NUMBER	1	格式: v1,v2...	模块:
checkpointInterval	Checkpoint Interva		NUMBER	10	格式: v1,v2...	模块:
implicitPref	Implicit Pref		BOOLEAN	false	格式: v1,v2...	模块:



模块代码：

- 一段普通的Spark的代码
- 参数使用特殊的占位符
- 参数信息自动识别为配置项

* 如果模块需要保存，则相关逻辑必须封装为spark.ml包中Transformer和Estimator的形式

导出的训练模型会保存在HDFS相关的路径下

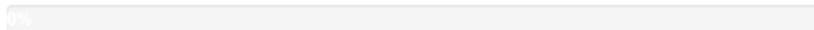
批量保存模块 (不可保存模块已经设置为不可选!)



保存描述信息(选填):

Test

保存模块进度:



✓	模块ID	用户ID	保存状态	出错信息	工程名	模型名称	
✓	7714	S52060	Not_Need_Save		电影推荐-zy	执行优化	
✓	7715	S52060	Not_Need_Save		电影推荐-zy	DF Split-2	
✓	7716	S52060	Ready		电影推荐-zy	ALS	
✓	7717	S52060	Not_Need_Save		电影推荐-zy	Model Transform	
✓	7718	S52060	Not_Need_Save		电影推荐-zy	scala transform	

Total Items: 6

Save

Close

查看已经保存的模型

所有已经保存模型列表(根据保存时间排序,最新排序最前)



	保存时间(version)	用户ID	工程名称	模型名称	注释	模型类名	保存路径	操作
<input checked="" type="checkbox"/>	1479369240000 (1)	S52060	电影推荐-zy	ALS		org.apache.spark.ml.r...	/user/atomml/spark_transformer/S52060/...	
<input checked="" type="checkbox"/>	2016-11-17 15:54:00							

Total Items: 1 (Showing Items: 2)

1 / 1 items per page

1 - 2 of 2 items

提供Jar包，用户只要在maven中添加依赖，就能够使用相关的API

开发步骤：

1. 初始化
2. 将输入数据转化为DataFrame
3. Load起已经保存的Transformer和Model
4. 按照顺序调用Transform和predict方法

```
public static void main( String[] args )
{
    String prefixPath = "/spark/data/transformer/";

    // 准备工作, 两个步骤:
    // step 1: 初始化predict engine
    PredictEngine.getInstance().initPredictEngine();

    // step 2: 给predict engine 加载各种transformer/model
    Loader modelLoader = new ModelLoader();
    Loader transformerLoader = new TransformerLoader();

    modelLoader.load("org.apache.spark.ml.classification.LogisticRegressionModel", prefixPath+"di_tao_LogisticRegression");

    transformerLoader.load("org.apache.spark.ml.feature.StringIndexerModel", prefixPath+"di_tao_StringIndexer");
    transformerLoader.load("org.apache.spark.ml.feature.StopWordsRemover", prefixPath+"di_tao_RemoveStopwords");
    transformerLoader.load("org.apache.spark.ml.feature.RegexTokenizer", prefixPath+"di_tao_RegexTokenizer");
    transformerLoader.load("org.apache.spark.ml.feature.CountVectorizerModel", prefixPath+"di_tao_CountVectorizer");

    List<String> testData = new ArrayList<>();
    String email = "Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...";

    // 单条数据 预测过程
    for (int i=0; i<500; i++) {
        testData.clear();
        testData.add(email+ " " + UUID.randomUUID().toString());
        testPredict(testData);
    }
}
```

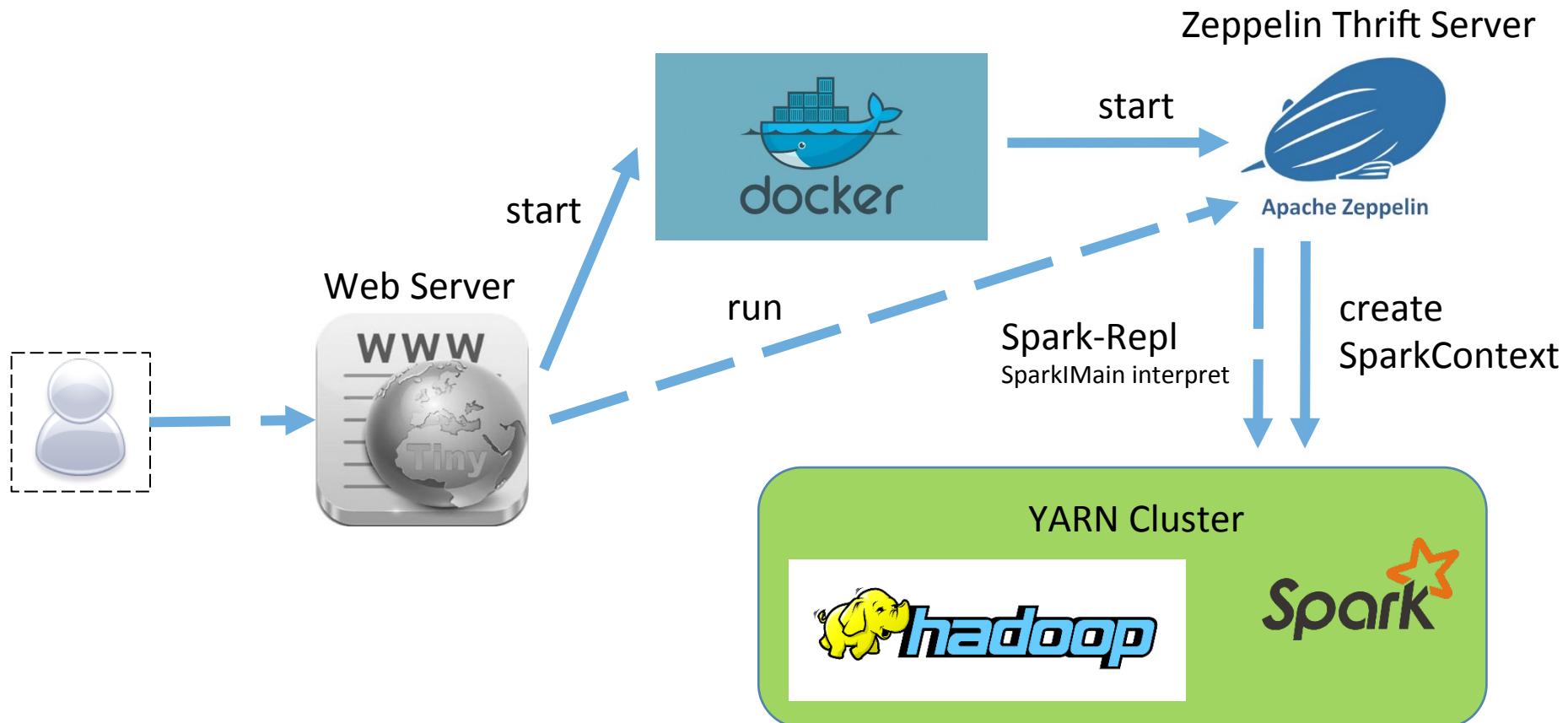
背景介绍

算法平台的功能

算法平台的架构和实现

算法平台的应用

算法平台未来的方向



每个训练任务（称为pipeline）会单独起一个Zeppelin的Thrift Server

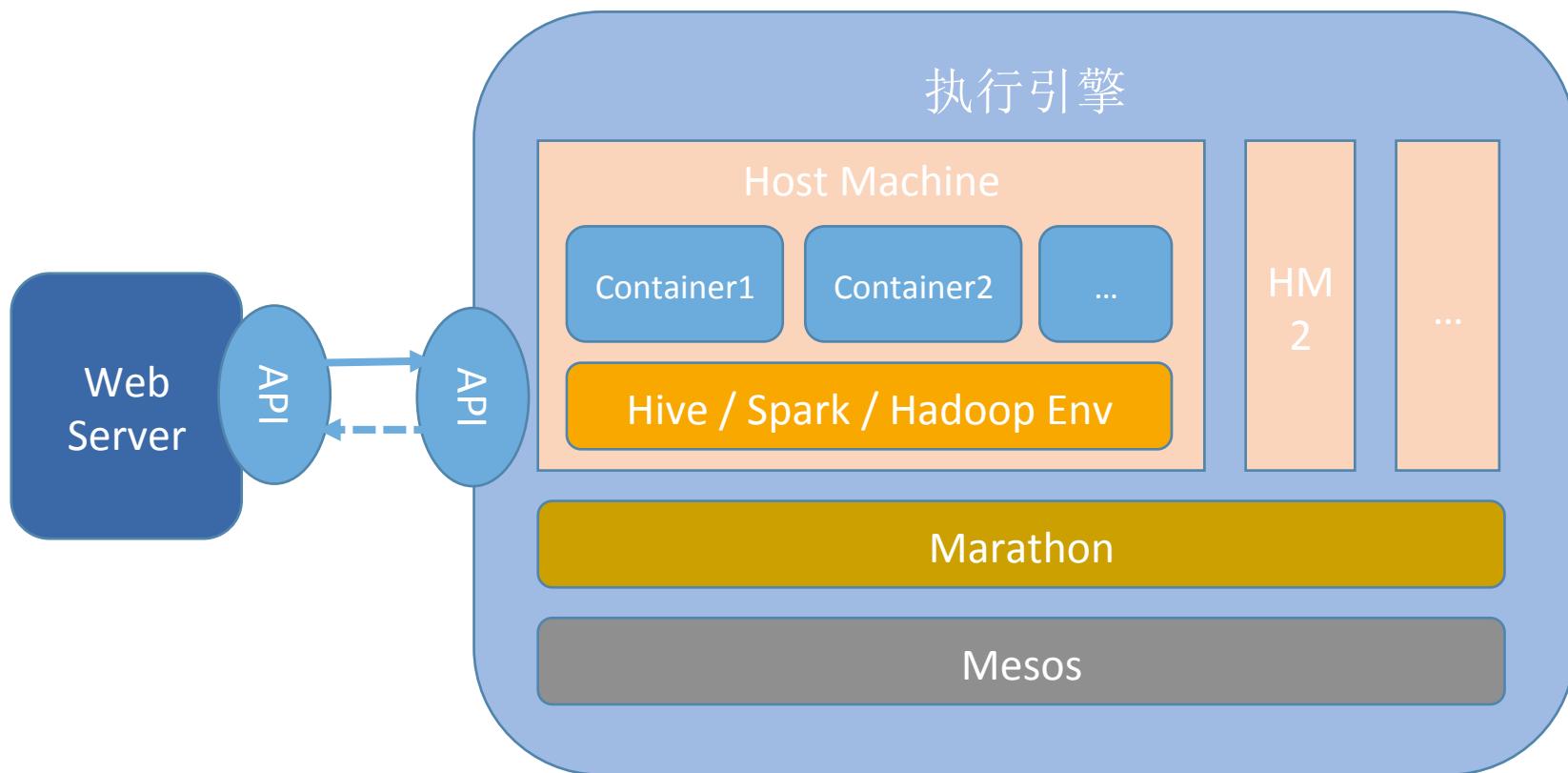
初始化(Lazy):

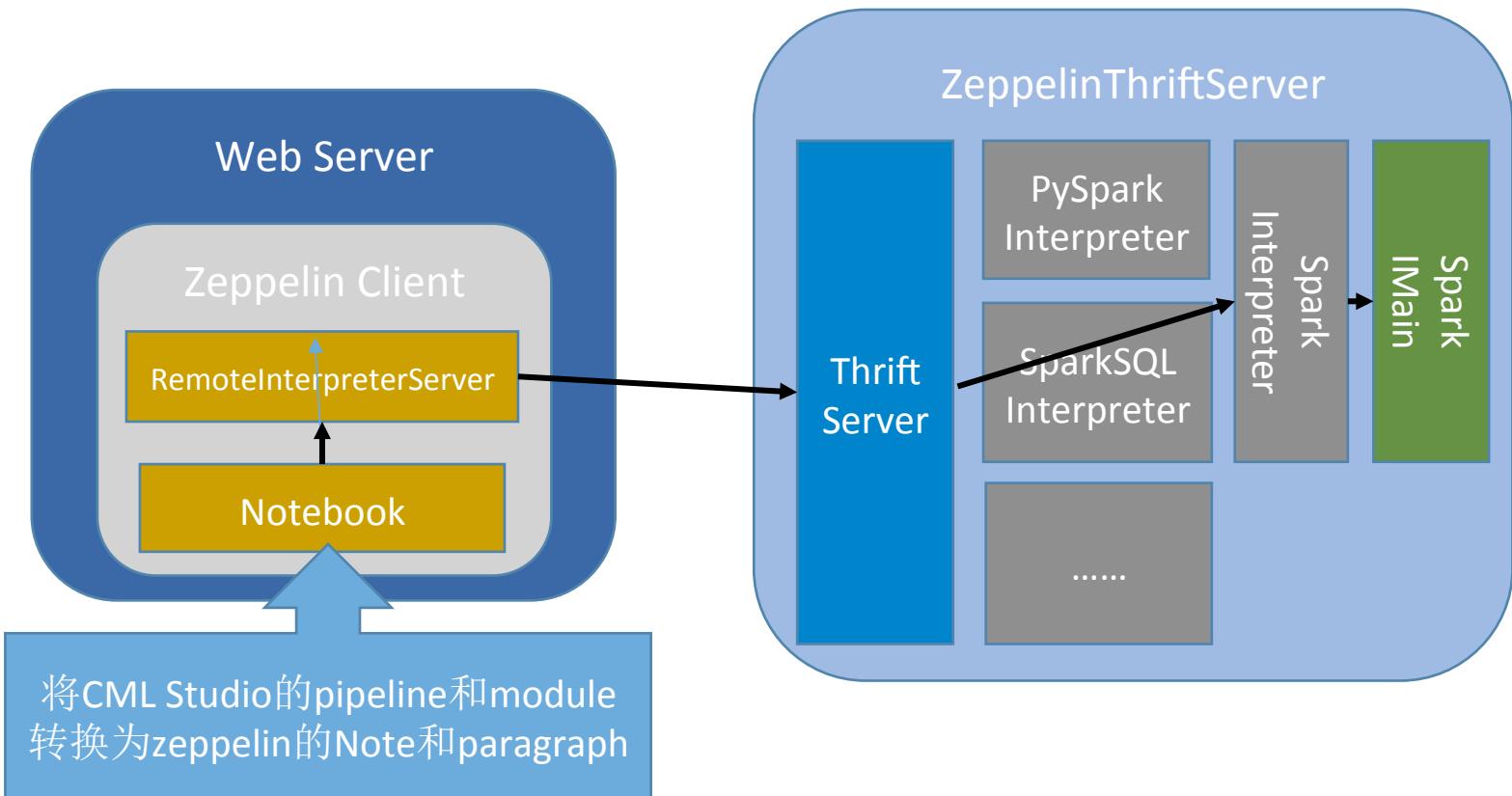
- Web Server调用Marathon(Mesos)的Restful API启动Zeppelin Thrift Server
- Zeppelin Thrift Server会在YARN集群中创建一个常驻的Spark任务

执行:

- Web Server中集成了Zeppelin Client，我们先将Pipeline和Module映射成为Zeppelin的Note和Paragraph，然后就会被提交到Zeppelin的Thrift Server执行
- Zeppelin的Thrift Server使用Spark-Repl(Spark的交互界面就是使用这个实现的)，将scala的代码编译后提交到常驻在YARN集群的Spark任务中执行

负责Zeppelin Thrift Server的生命周期的管理





选择Docker和Zeppelin的原因：

- Mesos提供了方便的资源管理的功能
- Marathon提供了简便的应用管理的功能
- Zeppelin则提供了一个Spark交互执行的引擎

由于人力资源的原因，使我们选择使用开源产品或是在开源项目的基础上进行2次开发

- 整体加速了项目的进展
- 在二次开发的过程中锻炼了团队的技术能力

- 不断的打磨产品
 - 目前的产品形态经过了几个不同版本的演化
 - 产品的易用性和稳定性是它的生命，关注使用细节
 - Example1 最初版本每个模块的结果都会落地，每个模块的输入输出都是需要用户手动填写的
 - Example2 批量结果导出的功能在最初的版本中不存在，只提供用户手动导出单个模块的训练结果的功能
 - Example 3 数据探索的功能是根据用户的需求加入的

技术经验分享（具体技术点）

1. Python代码模块的加入：将Input的Dataframe注册成为临时表
2. Spark资源的释放：Zeppelin的后端获取到当前Zeppelin Thrift Server的状态，如果状态超过一定时间为NO_OP，则关闭它
3. XGBoost模块加入过程中的经验
 - <https://github.com/dmlc/xgboost/issues/1276>
nWorkers的值不能大于可用的核数，否则会出现Hang住不动的情况
 - <https://github.com/dmlc/xgboost/issues/1284>

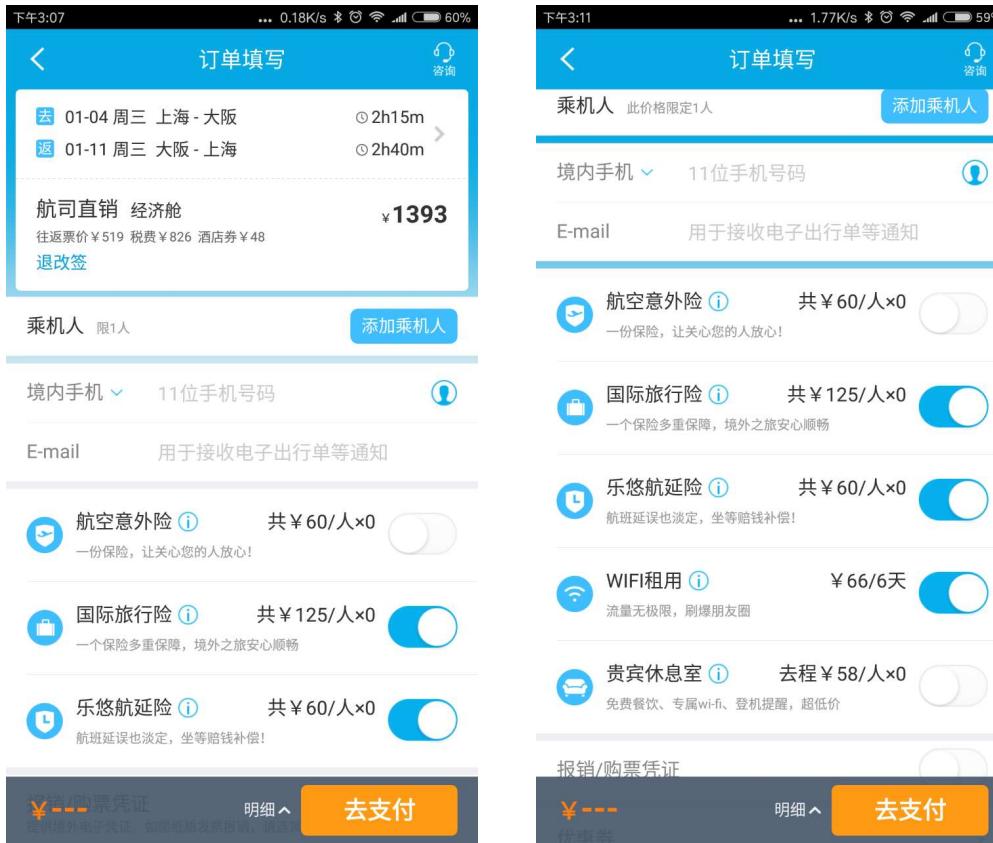
背景介绍

算法平台的功能

算法平台的架构和实现

算法平台的应用

算法平台未来的方向



用户在携程上购买机票后，
携程会给他们推荐一些机票的附加产品

我们根据

- 用户历史的购买行为的信息
- 用户画像的信息
- 航班本身的信息

来预测用户购买这些产品的概率，从而来决定是否默认帮用户勾选上某些产品

根据

- 天气（主要）
- 航班的信息
- 其他信息

来预测需要的客服人员的数量（每周一次）

根据预测的客服的需求量进行客服的自动排班



度假平台上零售、代理产品的刷单比较猖獗，据估计非自营刷单占到订单整体的30%左右。这些刷单产生了严重的负面影响，亟待清理。

行程介绍 | 费用 | 预订须知 | 用户点评(10) [如烟灯](#) | 开始预订

用户点评

5.0 总体评分

5分(10) 4分(0) 3分(0) 2分(0) 1分(0) 图片(0)

全5分好评

评论无图

评论均来自PC端

点评时间均在工作时间

正直第17单品

行程介绍 | 费用 | 预订须知 | 用户点评(199) [如烟灯](#) | 开始预订

用户点评

4.6 总体评分

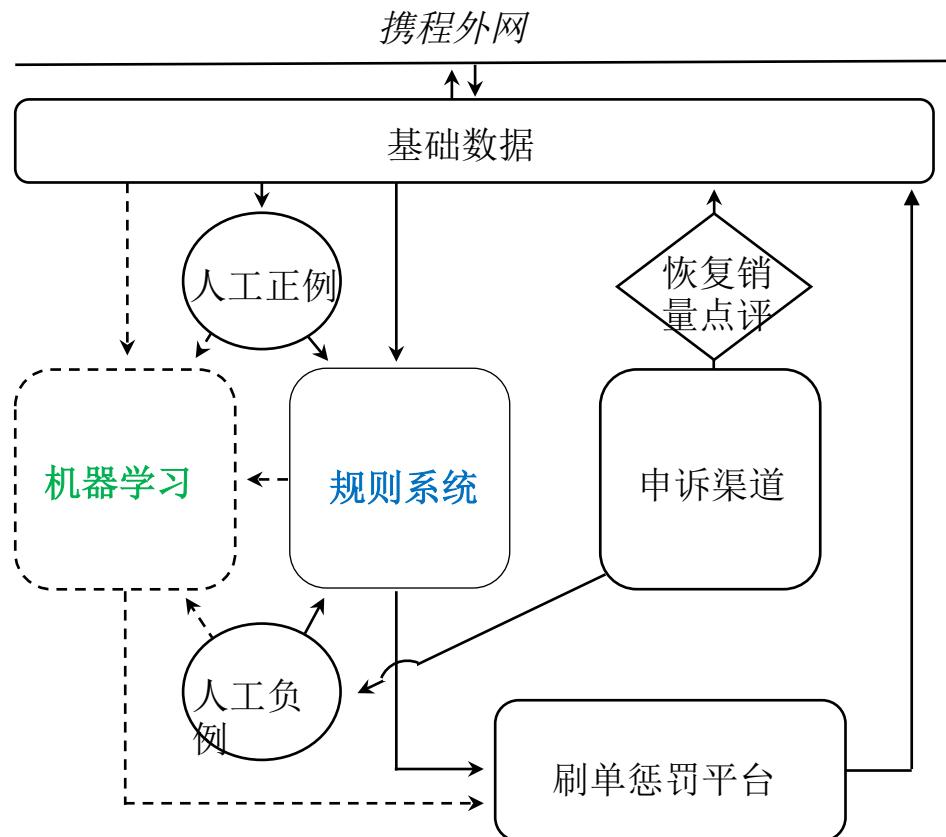
5分(154) 4分(26) 3分(0) 2分(7) 1分(6) 图片(109)

总有差评

评论带图

评论APP、PC都有

点评时间分散



在刷单识别项目开展初期，通过总结业务经验、对各维度数据进行统计分析，拟定了一系列**规则**来识别刷单，取得的一定的进展

将繁复的人工规则设定、阈值调节工作转化为新特征的整理，并通过各个渠道积累正例和反例

通过分类（主要使用XgBoost）的方式训练出自动判别刷单行为的模型

背景介绍

算法平台的功能

算法平台的架构和实现

算法平台的应用

算法平台未来的方向

近期的工作：

- 进一步加强数据探索的功能
- 通用模块易用性的加强
- 集成Cross Validation的功能
- 适配Spark2.0

未来的方向：

- 数据流功能的通用化，支持Spark之外的Module
- 项目开源

yi.zhang@ctrip.com



扫一扫上面的二维码图案，加我微信

对大数据技术感兴趣
“程序员”

有产品思维的“攻城师”

欢迎加入我们！

共同打造业界领先的数据平台和数据应用！

通过数据的技术让更多人的旅行更加精彩和幸福！

THANKS

BDTC 2016中国大数据技术大会
Big Data Technology Conference 2016