第二课 Spark ML Pipelines（ML管道）

目录

目录

ML Pipelines

ML Pipelines（ML管道）

Spark 2.2.0 Overview Programming Guides API Docs Deploying More

MLlib: Main Guide

- Basic statistics
- Pipelines
- Extracting, transforming and selecting features
- Classification and Regression
- Clustering
- Collaborative filtering
- Frequent Pattern Mining
- Model selection and tuning
- Advanced topics

MLlib: RDD-based API Guide

- Data types
- Basic statistics
- Classification and regression
- Collaborative filtering
- Clustering
- Dimensionality reduction
- Feature extraction and transformation
- Frequent pattern mining
- Evaluation metrics
- PMML model export
- Optimization (developer)

Machine Learning Library (MLlib) Guide

MLlib is Spark's machine learning (ML) library. Its goal is to make practical machine learning scalable and easy. At a high level, it provides tools such as:

- ML Algorithms: common learning algorithms such as classification, regression, clustering, and collaborative filtering
- Featurization: feature extraction, transformation, dimensionality reduction, and selection
- Pipelines: tools for constructing, evaluating, and tuning ML Pipelines
- Persistence: saving and load algorithms, models, and Pipelines
- Utilities: linear algebra, statistics, data handling, etc.

Announcement: DataFrame-based API is primary API

The MLlib RDD-based API is now in maintenance mode.

As of Spark 2.0, the RDD-based APIs in the `spark.mllib` package have entered maintenance mode. The primary Machine Learning API for Spark is now the DataFrame-based API in the `spark.ml` package.

What are the implications?

- MLlib will still support the RDD-based API in `spark.mllib` with bug fixes.
- MLlib will not add new features to the RDD-based API.
- In the Spark 2.x releases, MLlib will add features to the DataFrames-based API to reach feature parity with the RDD-based API.
- After reaching feature parity (roughly estimated for Spark 2.3), the RDD-based API will be deprecated.
- The RDD-based API is expected to be removed in Spark 3.0.

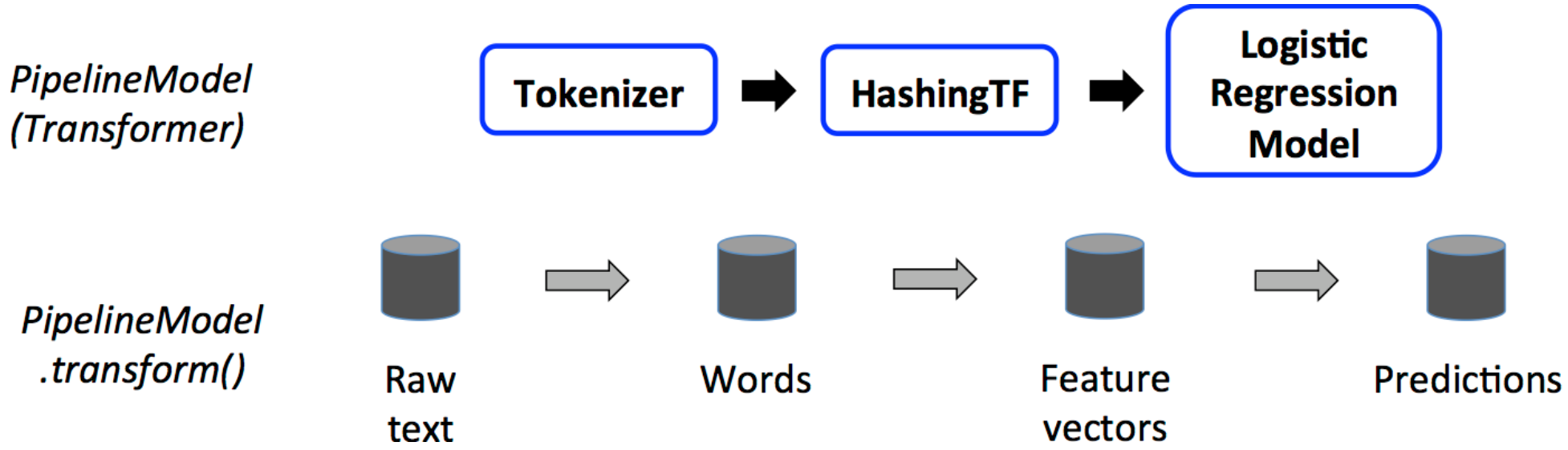
Why is MLlib switching to the DataFrame-based API?

- DataFrames provide a more user-friendly API than RDDs. The many benefits of DataFrames include Spark Datasources, SQL/DataFrame queries, Tungsten and Catalyst optimizations, and uniform APIs across languages.
- The DataFrame-based API for MLlib provides a uniform API across ML algorithms and across multiple languages.
- DataFrames facilitate practical ML Pipelines, particularly feature transformations. See the Pipelines guide for details.

What is "Spark ML"?

ML Pipelines（ML管道）

- spark.ml 则提供了基于DataFrames 高层次的API，可以用来构建机器学习工作流（PipeLine）。



Pipelines的主要概念

- **DataFrame**（数据模型）：

1）DataFrame 作为 ML 的数据集。

2） ML可以应用于各种各样的数据类型，比如向量，文本，图形和结构化数据。**API** 采用 **Spark Sql** 的 **DataFrame** 就是为了支持各种各样的数据类型。

Pipelines的主要概念

- **Transformer**（转换器）：

1） 转换器是特征变换和机器学习模型的抽象。转换器必须实现transform方法，这个方法将一个 DataFrame 转换成另一个 DataFrame，通常是附加一个或者多个列。

例1 ：一个特征变换器是输入一个 DataFrame，读取一个列（比如：text），将其映射成一个新列（比如，特征向量），然后输出一个新的 DataFrame 并包含这个映射的列。

例2 ：一个机器学习模型是输入一个 DataFrame，读取包含特征向量的列,预测每个特征向量的标签,并输出一个新的 DataFrame ，并附加预测标签作为一列。

Pipelines的主要概念

- **Estimators（模型学习器）**

1）Estimators 模型学习器是拟合和训练数据的机器学习算法或者其他算法的抽象。

2）Estimator（模型学习器）实现 fit() 方法，这个方法输入一个 DataFrame 并产生一个 Model 即一个 Transformer（转换器）。

3）例如：一个机器学习算法是一个 Estimator 模型学习器,比如这个算法是 LogisticRegression（逻辑回归），调用 fit() 方法训练出一个 LogisticRegressionModel,这是一个 Model,因此也是一个Transformer（转换器）.

Pipelines的主要概念

- **Pipeline**（管道）

1）Pipeline 将多个 Transformers 和 Estimators 绑在一起形成一个工作流.

在机器学习中,通常会执行一系列算法来处理和学习模型，比如，一个简单的文本文档处理流程可能包括这几个步骤：

A.把每个文档的文本分割成单词.

B.将这些单词转换成一个数值型特征向量.

C.使用特征向量和标签学习一个预测模型.

MLlib 代表一个流水线， 就是一个 Pipeline（管道），Pipeline（管道） 包含了一系列有特定顺序的管道步骤（Transformers（转换器） 和 Estimators（模型学习器））

Pipelines的主要概念

- **Parameter**（参数）

1）MLib 的 Estimators（模型学习器）和 Transformers（转换器）使用统一的 API 来指定参数。Param 是具有自包含定义的参数，ParamMap 是一组（参数,值）对.

2）将参数传递给算法主要有两种方式

例1： **lr是LogisticRegression**实例，调用**lr.setMaxIter(10)** 使**lr.fit()** 最多10次迭代。

例2：传递一个 **ParamMap** 给 **fit()** 函数和 **transform()** 函数。**ParamMap** 的任意参数将会覆盖前面调用实例通过 **setter** 方法指定的参数。

参数属于特定的 **Estimators**（模型学习器）和 **Transformers**（转换器）的实例。

例如，如果我们有两个 **LogisticRegression** 实例 **lr1** 和 **lr2**，然后我们可以使用指定的 **maxIter** 参数来构建**ParamMap**：**ParamMap(lr1.maxIter -> 10, lr2.maxIter -> 20)**。

Pipelines的主要概念

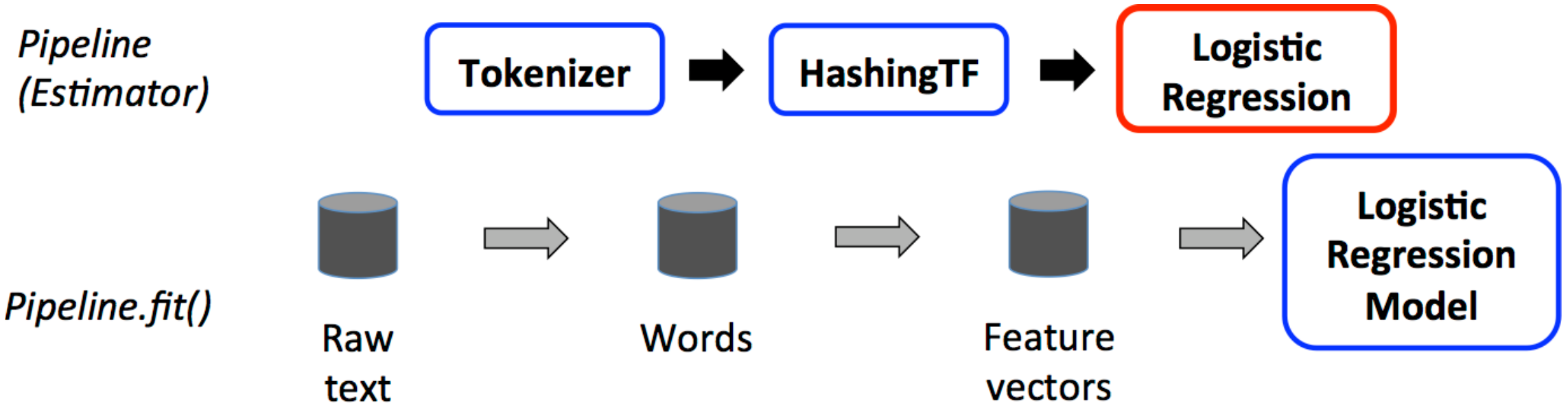
- 保存和加载管道

通常情况下，将模型或管道保存到磁盘上以供将来使用。从 Spark 1.6开始，模型导入导出功能被加入到 Pipeline API。大多数基本的 transformers（转换器）是被支持的,包括一些更基本的 ML Models。请根据算法 API 文档判断是否支持保存和加载

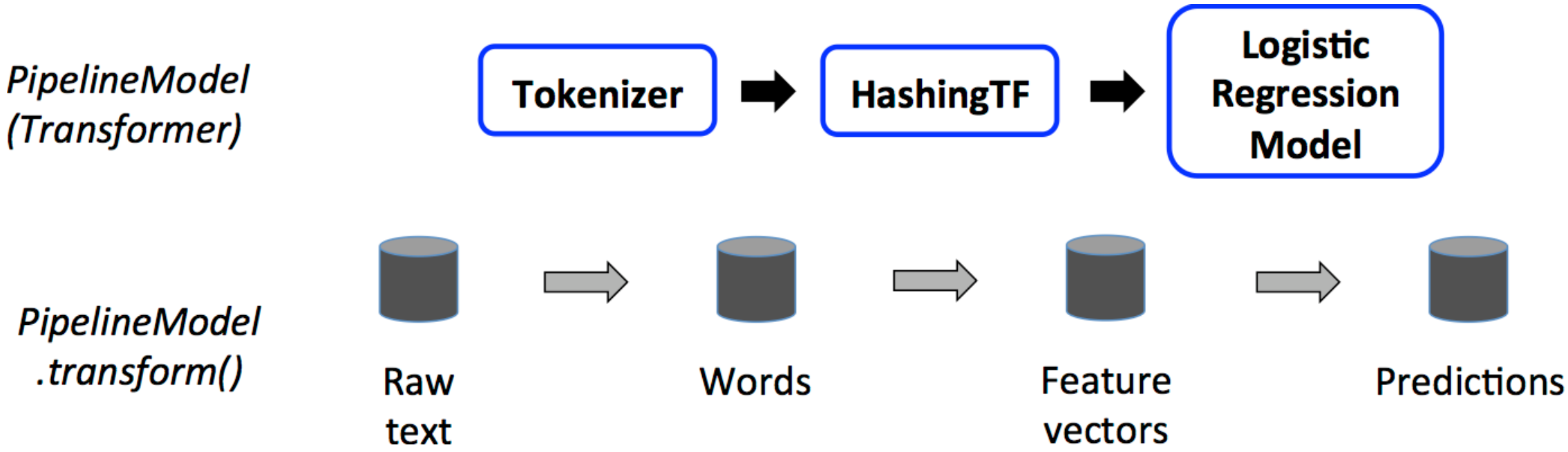
Pipelines的实例

- 1）一个 pipeline 由多个步骤组成，每一个步骤都是一个 Transformer（转换器）或者 Estimator（模型学习器）。
- 2）这些步骤按顺序执行，首先输入的 DataFrame ，然后通过每个阶段进行转换。
- 3）在 Transformer （转换器）步骤中，DataFrame 会调用 transform() 方法。
- 4）在 Estimator（模型学习器）步骤中，fit() 方法被调用并产生一个 Transformer（转换器）(会成为 PipelineModel（管道模型）的一部分,或者适配 Pipeline),并且 DataFrame 会调用这个 Transformer’ s（转换器）的transform()方法。

Pipelines的实例



Pipelines的实例



目录

实例

实例: Estimator, Transformer, Param

```
import org.apache.spark.ml.feature._
import org.apache.spark.ml.classification.LogisticRegression
import org.apache.spark.ml.{ Pipeline, PipelineModel }
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.ml.linalg.{Vector, Vectors}
import org.apache.spark.sql.Row
import org.apache.spark.sql.SparkSession
    //1 训练样本
    val training = spark.createDataFrame(Seq(
      (1.0, Vectors.dense(0.0, 1.1, 0.1)),
      (0.0, Vectors.dense(2.0, 1.0, -1.0)),
      (0.0, Vectors.dense(2.0, 1.3, 1.0)),
      (1.0, Vectors.dense(0.0, 1.2, -0.5)))).toDF("label", "features")
```

实例: Estimator, Transformer, Param

```
    //2 创建逻辑回归Estimator
    val lr = new LogisticRegression()
    println("LogisticRegression parameters:\n" + lr.explainParams() + "\n")

    //3 通过setter方法设置模型参数
    lr.setMaxIter(10)
      .setRegParam(0.01)

    //4 训练模型
    val model1 = lr.fit(training)
    println("Model 1 was fit using parameters: " +
model1.parent.extractParamMap)
```

实例: Estimator, Transformer, Param

```
//5 通过ParamMap设置参数方法
val paramMap = ParamMap(lr.maxIter -> 20)
  .put(lr.maxIter, 30)
  .put(lr.regParam -> 0.1, lr.threshold -> 0.55)

//5 ParamMap合并.
val paramMap2 = ParamMap(lr.probabilityCol -> "myProbability")
val paramMapCombined = paramMap ++ paramMap2

//6 训练模型，采用paramMap参数
// paramMapCombined会覆盖所有lr.set设置的参数
val model2 = lr.fit(training, paramMapCombined)
println("Model 2 was fit using parameters: " +
model2.parent.extractParamMap)
```

实例: Estimator, Transformer, Param

```
//7 测试样本
val test = spark.createDataFrame(Seq(
  (1.0, Vectors.dense(-1.0, 1.5, 1.3)),
  (0.0, Vectors.dense(3.0, 2.0, -0.1)),
  (1.0, Vectors.dense(0.0, 2.2, -1.5)))).toDF("label", "features")

//8 对模型进行测试
model2.transform(test)
  .select("features", "label", "myProbability", "prediction")
  .collect()
  .foreach {
  case Row(features: Vector, label: Double, prob: Vector, prediction: Double) =>
      println(s"($features, $label) -> prob=$prob, prediction=$prediction")
  }
```

实例: Pipeline

```
import org.apache.spark.ml.feature._
import org.apache.spark.ml.classification.LogisticRegression
import org.apache.spark.ml.{ Pipeline, PipelineModel }
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.ml.linalg.{ Vector, Vectors }
import org.apache.spark.ml.feature.{ HashingTF, Tokenizer }
import org.apache.spark.sql.Row
import org.apache.spark.sql.SparkSession
    //1 训练样本
    val training = spark.createDataFrame(Seq(
      (0L, "a b c d e spark", 1.0),
      (1L, "b d", 0.0),
      (2L, "spark f g h", 1.0),
      (3L, "hadoop mapreduce", 0.0))).toDF("id", "text", "label")
```

实例: Pipeline

```
//2 ML pipeline 参数设置，包括三个过程：首先是tokenizer，然后是hashingTF，最后是lr。
  val tokenizer = new Tokenizer()
    .setInputCol("text")
    .setOutputCol("words")
  val hashingTF = new HashingTF()
    .setNumFeatures(1000)
    .setInputCol(tokenizer.getOutputCol)
    .setOutputCol("features")
  val lr = new LogisticRegression()
    .setMaxIter(10)
    .setRegParam(0.001)
  val pipeline = new Pipeline()
    .setStages(Array(tokenizer, hashingTF, lr))
```

实例: Pipeline

```
//4 保存pipeline模型
model.write.overwrite().save("/tmp/spark-logistic-regression-model")

//5 保存pipeline
pipeline.write.overwrite().save("/tmp/unfit-lr-model")

//6 加载pipeline模型
val sameModel = PipelineModel.load("/tmp/spark-logistic-regression-model")
```

实例: Pipeline

```
//7 测试样本
val test = spark.createDataFrame(Seq(
  (4L, "spark i j k"),
  (5L, "l m n"),
  (6L, "spark hadoop spark"),
  (7L, "apache hadoop"))).toDF("id", "text")

//8 模型测试
model.transform(test)
  .select("id", "text", "probability", "prediction")
  .collect()
  .foreach {
    case Row(id: Long, text: String, prob: Vector, prediction: Double) =>
        println(s"($id, $text) --> prob=$prob, prediction=$prediction")  }
```

目录

构建机器学习工作流
案例展示

Thanks

FAQ时间