Machine Learning Library (MLlib)

Big Data & Predictive Analysis Lanjut

Arif Laksito, M. Kom

Machine Learning Library (MLlib)

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:

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

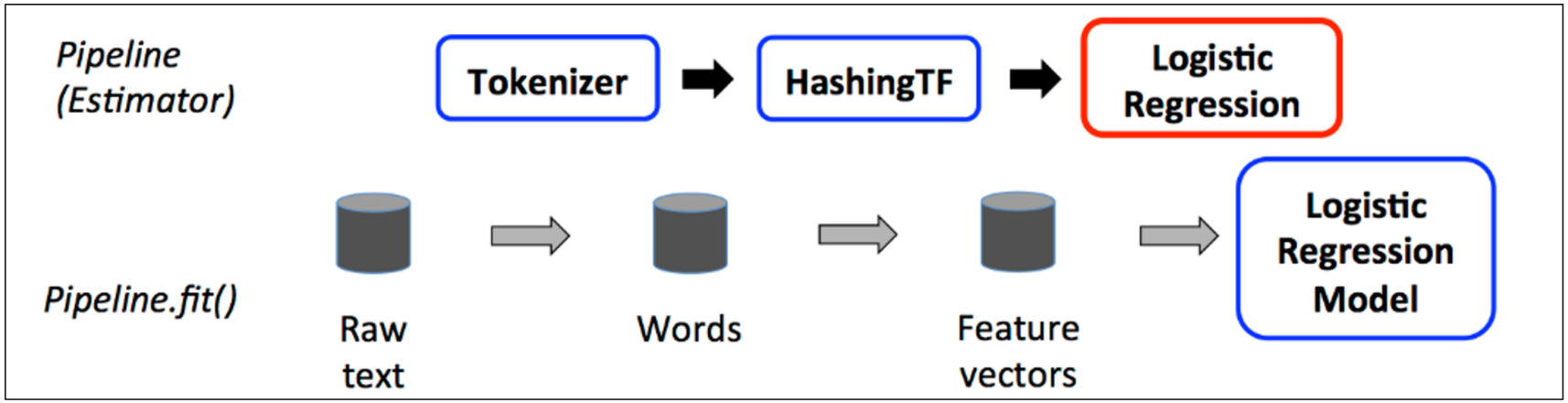
.

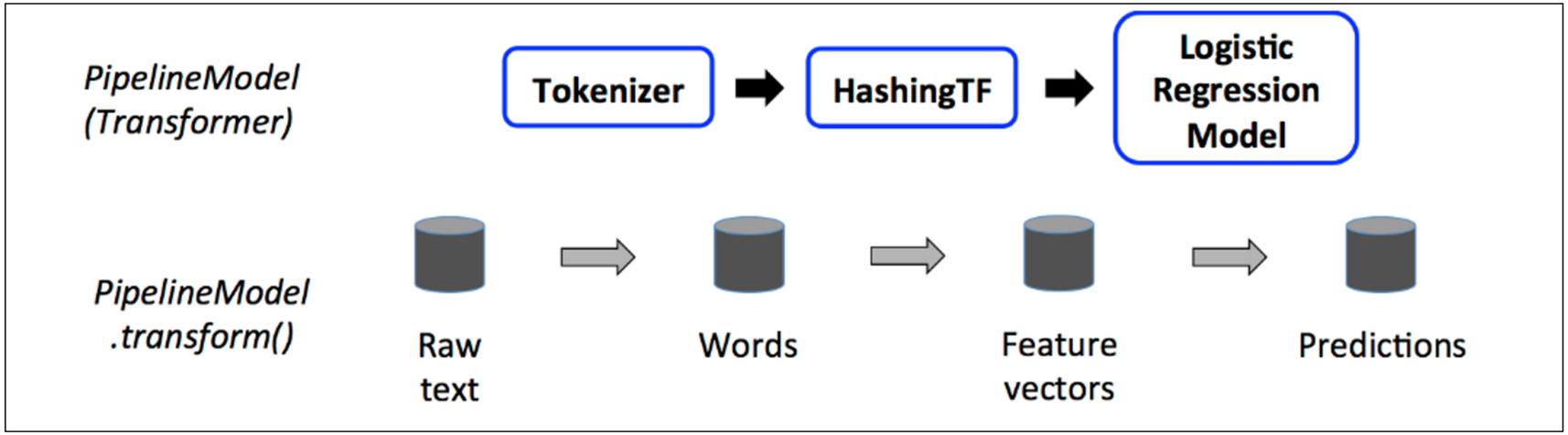
Pipeline

- A Pipeline is specified as a sequence of stages, and each stage is either a Transformer or an Estimator.
- A Pipeline is an Estimator. Thus, after a Pipeline's fit() method runs, it produces a PipelineModel, which is a Transformer. This PipelineModel is used at test time; the figure below illustrates this usage.

.

Pipeline


Pipeline
(Estimator)
Tokenizer
HashingTF
Logistic
Regression
Pipeline.fit()
Raw
text
Words
Feature
vectors
Logistic
Regression
Model


PipelineModel
(Transformer)
Tokenizer
HashingTF
Logistic
Regression
Model
PipelineModel
.transform()
Raw
text
Words
Feature
vectors
Predictions

Let's code

PySpark Data Manipulation

Featurization

TF-IDF

In [18]:

```
from pyspark.ml.feature import HashingTF, IDF, Tokenizer

sentenceData = spark.createDataFrame([
    (0.0, "Hi I heard about Spark"),
    (0.0, "I wish Java could use case classes"),
    (1.0, "Logistic regression models are neat")
], ["label", "sentence"])

tokenizer = Tokenizer(inputCol="sentence", outputCol="words")
wordsData = tokenizer.transform(sentenceData)

hashingTF = HashingTF(inputCol="words", outputCol="rawFeatures", numFeatures=20)
featurizedData = hashingTF.transform(wordsData)
# alternatively, CountVectorizer can also be used to get term frequency vectors

idf = IDF(inputCol="rawFeatures", outputCol="features")
idfModel = idf.fit(featurizedData)
rescaledData = idfModel.transform(featurizedData)

rescaledData.select("label", "features").show()
```

Featurization

Word2Vec

```
from pyspark.ml.feature import Word2Vec

# Input data: Each row is a bag of words from a sentence or document.
documentDF = spark.createDataFrame([
    ("Hi I heard about Spark".split(" "), ),
    ("I wish Java could use case classes".split(" "), ),
    ("Logistic regression models are neat".split(" "), )
], ["text"])

# Learn a mapping from words to Vectors.
word2Vec = Word2Vec(vectorSize=3, minCount=0, inputCol="text", outputCol="result")
model = word2Vec.fit(documentDF)

result = model.transform(documentDF)
for row in result.collect():
    text, vector = row
    print("Text: [%s] => \nVector: %s\n" % (", ".join(text), str(vector)))
```

Featurization

Countvectorizer

```
from pyspark.ml.feature import CountVectorizer

# Input data: Each row is a bag of words with a ID.
df = spark.createDataFrame([
    (0, "a b c".split(" ")),
    (1, "a b b c a".split(" "))
], ["id", "words"])

# fit a CountVectorizerModel from the corpus.
cv = CountVectorizer(inputCol="words", outputCol="features", vocabSize=3, minDF=2.0)

model = cv.fit(df)

result = model.transform(df)
result.show(truncate=False)
```

References

- https://campus.datacamp.com/courses/introduction-to-pyspark
- https://www.tutorialspoint.com/pyspark/pyspark_sparkcontext.htm
- https://sparkbyexamples.com/pyspark/pyspark-column-functions/
- https://sparkbyexamples.com/pyspark/pyspark-what-is-sparksession