Memento

"A retentive memory may be a good thing, but the ability to forget is the true token of greatness." – Elbert Hubbard

Spark: Custom UDF Example

Posted on October 2, 2015

UDF (User defined functions) and UDAF (User defined aggregate functions) are key components of big data languages such as Pig and Hive. They allow to extend the language constructs to do adhoc processing on distributed dataset. Previously I have blogged about how to write custom UDF/UDAF in Pig (here) and Hive(Part I & II) . In this post I will focus on writing custom UDF in spark. UDF and UDAF is fairly new feature in spark and was just released in Spark 1.5.1. So its still in evolution stage and quite limited on things you can do, especially when trying to write generic UDAFs. I will talk about its current limitations later on.

As a motivating example assume we are given some student data containing student's name, subject and score and we want to convert numerical score into ordinal categories based on the following logic:

- A –> if score >= 80
- B –> if score >= 60
- C –> if score >= 35
- D –> otherwise

Below is the relevant python code if you are using pyspark.

```python
# Generate Random Data
import itertools
import random
students = ['John', 'Mike','Matt']
subjects = ['Math', 'Sci', 'Geography', 'History']
random.seed(1)
data = []

for (student, subject) in itertools.product(students, subjects):
    data.append((student, subject, random.randint(0, 100)))

# Create Schema Object
from pyspark.sql.types import StructType, StructField, IntegerType, StringType
schema = StructType([
            StructField("student", StringType(), nullable=False),
            StructField("subject", StringType(), nullable=False),
            StructField("score", IntegerType(), nullable=False)
    ])

# Create DataFrame
from pyspark.sql import HiveContext
sqlContext = HiveContext(sc)
rdd = sc.parallelize(data)
df = sqlContext.createDataFrame(rdd, schema)

# Define udf
from pyspark.sql.functions import udf
def scoreToCategory(score):
    if score >= 80: return 'A'
    elif score >= 60: return 'B'
    elif score >= 35: return 'C'
    else: return 'D'

```

```
34  udfScoreToCategory=udf(scoreToCategory, StringType())
35  df.withColumn("category", udfScoreToCategory("score")).show(10)
```

Line 2-10 is the basic python stuff. We are generating a random dataset that looks something like this:

student	subject	score
John	Math	13
…	…	…
Mike	Sci	45
Mike	Geography	65
…	…	…

Next line 12-24 are dealing with constructing the dataframe. The main part of the code is in line 27-34. We first define our function in a normal python way.

Below is scala example of the same:

```scala
// Construct Dummy Data
import util.Random
import org.apache.spark.sql.Row
implicit class Crossable[X](xs: Traversable[X]) {
  def cross[Y](ys: Traversable[Y]) = for { x <- xs; y <- ys } yield (x, y)
}
val students = Seq("John", "Mike","Matt")
val subjects = Seq("Math", "Sci", "Geography", "History")
val random = new Random(1)
val data =(students cross subjects).map{x  =>  Row(x._1, x._2,random.nextInt(100))}.t

// Create Schema Object
import org.apache.spark.sql.types.{StructType, StructField, IntegerType, StringType}
val schema = StructType(Array(
            StructField("student", StringType, nullable=false),
            StructField("subject", StringType, nullable=false),
            StructField("score", IntegerType, nullable=false)
    ))

// Create DataFrame
import org.apache.spark.sql.hive.HiveContext
val rdd = sc.parallelize(data)
val df = sqlContext.createDataFrame(rdd, schema)

// Define udf
import org.apache.spark.sql.functions.udf
def udfScoreToCategory=udf((score: Int) => {
        score match {
        case t if t >= 80 => "A"
        case t if t >= 60 => "B"
        case t if t >= 35 => "C"
        case _ => "D"
    }})
df.withColumn("category", udfScoreToCategory(df("score"))).show(10)
```

Rate this:

6 Votes

Share:

Email Pinterest Google Tumblr LinkedIn Twitter Facebook

Related

Spark: Custom UDAF Example	Writing Hive Custom Aggregate Functions (UDAF): Part II	On Writing Python UDF for Pig: A perspective
In "General"	In "General"	In "General"

About Ritesh Agrawal

I am a applied researcher who enjoys anything related to statistics, large data analysis, data mining, machine learning and data visualization.

View all posts by Ritesh Agrawal →

This entry was posted in Big Data, Programming and tagged Hadoop, PySpark, scala, Spark. Bookmark the permalink.

Memento

The Twenty Ten Theme. *Create a free website or blog at WordPress.com.*