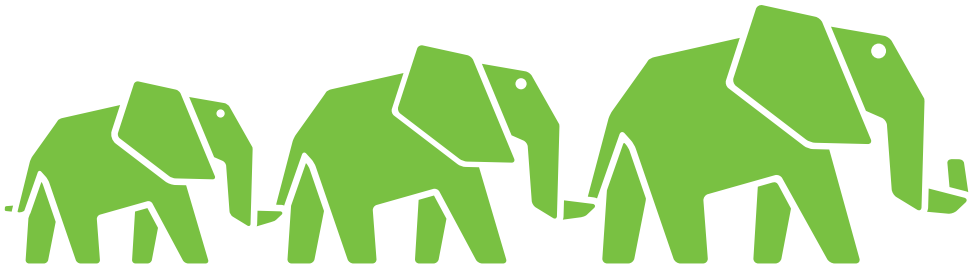


Hortonworks

Spam Classification with Mllib LAB

A Hortonworks University
Hadoop Training Course

Title: LAB GUIDE: Data Science for the Hortonworks Data Platform
Revision 2

Lab: Spam Classifier using Spark MLlib

Objective:	Become familiar with using Spark MLlib to run data science algorithms on a Hadoop cluster.
Successful Outcome:	You will have created a spam classifier with MLlib.
Before You Begin:	Your HDP cluster should be up and running in the classroom VM.

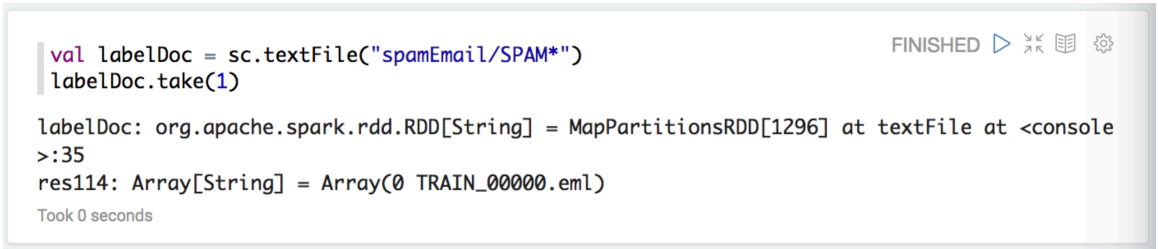
1.1. Before we can begin this lab, we need to retrieve the data and put it in the HDFS. Run the following commands below the screenshot to retrieve the data, unzip the file, and put the data into the hdfs


```
%sh
wget https://www.dropbox.com/s/5lzhi4ubwann65z/spamEmail.zip?dl=1
mv spamEmail.zip?dl=1 spamEmail.zip
unzip spamEmail.zip > out.log
ls
hdfs dfs -put spamEmail
```

Step 2: There are 2 types of files in the spamEmail folder. One is called SPAMTrain.txt, which contains labels for the emails, and the other are training emails (names are like "TRAIN_0####.eml). We need to start with this data in its raw format.

2.1. Create an RDD for the datafile that contains the email name and the labels. Do it take on the rdd to verify it was created correctly. Notice the format of the record. There are two values, the first is a number (0, or 1) which correspond to wether the email with that title is spam or not spam. If it's a 0, it is not spam, if it is 1, it is spam.

```
val labelDoc = sc.textFile("spamEmail/SPAM*")
```



2.2. Next we need to create an RDD for the emails. Each email is contained within its own file, so TRAIN_00000.eml is one email. Because of the nature of the data, we will use a new method for reading the data. The Spark api "wholeTextFiles" will read in all the emails into a PairRdd, where the key is the name of the file and the value is the content of the file. Do a take on the RDD to get an idea of the structure. Notice how the key is the entire file path, and in our labelsRdd, we only have the file name.

```
val spamEmail = sc.wholeTextFiles("spamEmail/T*")
```

2.3. Because the names in our LabelRdd and spamEmailRdd aren't the same, we need to massage that data. In order to get only the file name, we need to strip off everything before the last "/" in the key field. To do so, we'll use a substring and index function in scala.

```
val cleanEmailName = spamEmail.map{case (a,b) =>
    ((a.substring(a.lastIndexOf("/")+1)),b)}
```

```
val cleanEmailName = spamEmail.map{case (a,b) => ((a.substring(a.lastIndexOf("/")+1)),b)}
cleanEmailName.take(1)
```

```
cleanEmailName: org.apache.spark.rdd.RDD[(String, String)] = MapPartitionsRDD[1299] at map at
res118: Array[(String, String)] =
Array((TRAIN_00000.eml,"One of a kind Money maker! Try it for free!From nobody Wed Jul 15 14:
Content-Type: text/html;
        charset="iso-8859-1"
Content-Transfer-Encoding: 7bit
<body lang=EN-US>
<div class=Section1>
<p class=MsoBodyText style='text-align:justify'><b>CONSANTLY</b> being
bombarded by so-called FREE money-making systems that teases you with limited
information, and when it's all said and done, blind-sides you by demanding your
money/credit card information upfront in some slick way,<b> after-the-fact</b>!
Yes, I too was as skeptical about such offers and the Internet in general with
all its hype, as you probably are. Fortunate for me, my main business
slowed-down (<i>I have been self-employed all my life</i>), so I looked for
...
Took 0 seconds
```

2.4. Now that the spamEmailRDD names line up with the labelDocRdd names, we can do a join. But before we join, remember the data in the labelDoc is not in the correct format. So we need fix that.

```
Array[String] = Array(0 TRAIN_00000.eml)
```

```
val cleanLabelDoc = labelDoc.map(line =>
    line.split(" ")).map(line => (line(1),line(0)))
```

```
val cleanLabelDoc = labelDoc.map(line => line.split(" ")).map(line => (line(1),line(0)))
cleanLabelDoc.take(1)
```

```
cleanLabelDoc: org.apache.spark.rdd.RDD[(String, String)] = MapPartitionsRDD[1301] at map
res120: Array[(String, String)] = Array((TRAIN_00000.eml,0))
Took 0 seconds
```

2.5. Join the two RDD's together and only keep the values, since the names of the emails aren't important to us. Also, this part takes a while as some data is moving around the network. We want to store a copy of the output data in memory so the rest of this is faster.

```
val labelsAndEmails =
cleanLabelDoc.join(cleanEmailName).values
labelsAndEmails.take(1)
labelsAndEmails.cache()
labelsAndEmails.count()
```

```
val labelsAndEmails = cleanLabelDoc.join(cleanEmailName).values
labelsAndEmails.take(1)
labelsAndEmails.cache()
labelsAndEmails.count()

labelsAndEmails: org.apache.spark.rdd.RDD[(String, String)] = MapPartitionsRDD[1309] a
res124: Array[(String, String)] =
Array((1,"Re: Disable server so it does not start on reboot (even after
        upgrade)?From nobody Wed Jul 15 14:06:45 2015
Content-Type: text/plain; charset=ISO-8859-1
On Fri, May 14, 2010 at 04:29, Stan Hoeppner <stan@hardwarefreak.com> wrote:
```

2.6. Now that we have the data in the correct format, and a copy of the intermediate stored in memory, we can start working with the ML-Lib aspect of application.

Step 3: Lets do some Machine Learning

3.1. First we need to import the relevant packages

```
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.feature.HashingTF
import
org.apache.spark.mllib.classification.LogisticRegressionWithSGD
```

```
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.feature.HashingTF
import org.apache.spark.mllib.classification.LogisticRegressionWithSGD

import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.feature.HashingTF
import org.apache.spark.mllib.classification.LogisticRegressionWithSGD
Took 0 seconds
```

3.2. We now need to convert our words into numbers, using a HashingTF. This will convert the emails of words into a feature vector of numbers.

```
val HTF = new HashingTF(1000000)
val emailFeatures = labelsAndEmails.mapValues(feature =>
HTF.transform(feature.split(" ")))
```

```
val HTF = new HashingTF(1000000)
val emailFeatures = labelsAndEmails.mapValues(feature => HTF.transform(feature.split(" ")))

HTF: org.apache.spark.mllib.feature.HashingTF = org.apache.spark.mllib.feature.HashingTF@58fcl
emailFeatures: org.apache.spark.rdd.RDD[(String, org.apache.spark.mllib.linalg.Vector)] = Mapl
 mapValues at <console>:47
Took 1 seconds
```

3.3. Now that we have our features built, we need to label the features. In addition, this is the last step before we run our machine algorithm, so lets cache the dataset in memory, and take a look at the data.

```
val labelsAndFeatures = emailFeatures.map{case (a,b) =>
LabeledPoint(a.toInt,b)}
labelsAndFeatures.take(1)
labelsAndFeatures.cache()
labelsAndFeatures.count()
```

```
val labelsAndFeatures = emailFeatures.map{case (a,b) => LabeledPoint(a.toInt,b)}
labelsAndFeatures.take(1)
labelsAndFeatures.cache()
labelsAndFeatures.count()

labelsAndFeatures: org.apache.spark.rdd.RDD[org.apache.spark.mllib.regression.LabeledPoint] = Ma
2] at map at <console>:49
res132: Array[org.apache.spark.mllib.regression.LabeledPoint] = Array((1.0,(1000000,[0,62,73,79,
117,3123,3211,3365,3370,3371,3500,3543,3551,3555,3676,3707,14170,29746,44742,48745,61285,66370,6
991,84889,86838,89017,96727,101577,109267,111375,113291,113360,114732,114801,116103,117487,14309
786,153737,167067,173507,182877,187047,198223,209629,259841,261149,273964,298824,299032,308008,3
0,327396,336088,349771,353198,386457,389188,397534,402031,410173,415681,482894,512927,535600,536
49179,558823,558941,559070,579577,587246,599293,603460,609555,616397,619197,619420,627429,641872
34,652676,670953,696232,697954,700133,734940,739805,746592,746675,751800,757538,775713,779555,80
33: labelsAndFeatures.type = MapPartitionsRDD[1312] at map at <console>:49
res134: Long = 4323
Took 1 seconds
```

3.4. Now we have all of our data in the right format, lets build the model. It may take a couple minutes to build the model

```
val classified =
LogisticRegressionWithSGD.train(labelsAndFeatures,100)
```

```
val classified = LogisticRegressionWithSGD.train(labelsAndFeatures,100)

classified: org.apache.spark.mllib.classification.LogisticRegressionModel = org.apache.spark.
ogisticRegressionModel: intercept = 0.0, numFeatures = 1000000, numClasses = 2, threshold = 0
Took 263 seconds (outdated)
```

Step 4: The model is build, we can now start playing with the model and see how it predicts some tests. If the model predicts 0, it is spam, if the model predicts 1, it is not spam.

4.1. Lets test the model with a couple of made up sentences. Note, this is not the normal way to test models, but a fun and easy way. We will talk about evaluation in a bit.

```
val positiveResult = HTF.transform("Poker for money against
real playersGet your favorite Poker action!".split(" "))
val negativeResult = HTF.transform("Please report to
principal's office...".split(" "))

classified.predict(positiveResult)
classified.predict(negativeResult)
```