Chapter 12: Spark Standalone Cluster

Ex1: NLP - Tags

Requirement: Build a tags filter. Use the various NLP tools and a classifier, to predict tag for one question. In future questions could be auto-tagged by such a classifier or tags could be recommended to users prior to posting.

- Dataset: stack-overflow-data.csv. It contains Stack Overflow questions and associated tags.
- Link tham khảo: http://benalexkeen.com/multiclass-text-classification-with-pyspark/ (http://benalexkeen.com/multiclass-text-classification-with-pyspark/)

In [1]:

```
# Link HDFS: http://172.24.40.251:50070/
```

In [2]:

```
import findspark
findspark.init()
```

In [3]:

```
import pyspark
```

In [4]:

```
from pyspark.sql import SparkSession
from pyspark import SparkContext
from pyspark import SparkConf
```

In [5]:

```
SparkContext.setSystemProperty('spark.executor.memory', '6g')

sc = SparkContext(master='spark://172.25.53.2:7077', appName='Stack_Overflow')
```

In [7]:

```
sc
```

Out[7]:

SparkContext

Spark UI (http://PM503-GV:4041)

Version

```
v2.4.4
```

Master

```
spark://172.25.53.2:7077
```

AppName

```
Stack_Overflow
```

In [8]:

```
spark = SparkSession(sc)
```

In [10]:

```
file_name = "hdfs://172.24.40.251:19000/stack_overflow_data.csv"
```

In [11]:
```
data = spark.read.csv(file_name, inferSchema=True,header=True)
```

In [12]:
```
data.show(5)
```

```
+--------------------+-----------+
|                post|       tags|
+--------------------+-----------+
|what is causing t...|         c#|
|have dynamic html...|    asp.net|
|how to convert a ...|objective-c|
|.net framework 4 ...|       .net|
|trying to calcula...|     python|
+--------------------+-----------+
only showing top 5 rows
```

In [13]:
```
data.groupby('tags').count().show(30)
```

```
+-------------+-----+
|         tags|count|
+-------------+-----+
|       iphone| 2000|
|      android| 2000|
|           c#| 2000|
|         null|20798|
|      asp.net| 2000|
|         html| 2000|
|        mysql| 2000|
|       jquery| 2000|
|   javascript| 2000|
|          css| 2000|
|          sql| 2000|
|          c++| 2000|
|            c| 2000|
|  objective-c| 2000|
|         java| 2000|
|          php| 2000|
|         .net| 2000|
|          ios| 2000|
|       python| 2000|
|    angularjs| 2000|
|ruby-on-rails| 2000|
+-------------+-----+
```

In [14]:
```
tags_null_data = data.filter(data.tags.isNull())
```

In [15]:
```
tags_null_data.count()
```

Out[15]:
```
20798
```

In [16]:
```
data = data.filter(data.tags.isNotNull())
```

```
In [17]: data.count()

Out[17]: 40000

In [18]: from pyspark.sql.functions import *
```

Clean and Prepare the Data

** Create a new length feature: **

```
In [19]: from pyspark.sql.functions import length

In [20]: data = data.withColumn('length',length(data['post']))

In [21]: data.show()
```

```
+--------------------+------------+------+
|                post|        tags|length|
+--------------------+------------+------+
|what is causing t...|          c#|   833|
|have dynamic html...|     asp.net|   804|
|how to convert a ...| objective-c|   755|
|.net framework 4 ...|        .net|   349|
|trying to calcula...|      python|  1290|
|how to give alias...|     asp.net|   309|
|window.open() ret...|   angularjs|   495|
|identifying serve...|      iphone|   424|
|unknown method ke...|ruby-on-rails|  2022|
|from the include ...|   angularjs|  1279|
|when we need inte...|          c#|   995|
|how to install .i...|         ios|   344|
|dynamic textbox t...|     asp.net|   389|
|rather than bubbl...|           c|  1338|
|site deployed in ...|     asp.net|   349|
|connection in .ne...|        .net|   228|
|how to subtract 1...| objective-c|    62|
|ror console show ...|ruby-on-rails|  2594|
|distance between ...|      iphone|   336|
|sql query - how t...|         sql|  1037|
+--------------------+------------+------+
only showing top 20 rows
```

In [22]:

```
# Pretty Clear Difference
data.groupby('tags').mean().show()
```

```
+-------------+-----------+
|         tags|avg(length)|
+-------------+-----------+
|       iphone|    709.621|
|      android|  1713.4345|
|           c#|  1145.3065|
|      asp.net|     999.95|
|         html|   891.3105|
|        mysql|   1038.561|
|       jquery|   1081.507|
|   javascript|    964.396|
|          css|    954.809|
|          sql|    870.912|
|          c++|   1295.955|
|            c|  1121.1115|
|  objective-c|   972.8925|
|         java|   1357.308|
|          php|  1123.4205|
|         .net|   731.0075|
|          ios|   970.7565|
|       python|  1018.6695|
|    angularjs|  1294.7545|
|ruby-on-rails|  1244.2055|
+-------------+-----------+
```

Feature Transformations

In [23]:

```
from bs4 import BeautifulSoup

from pyspark import keyword_only
from pyspark.ml import Transformer
from pyspark.ml.param.shared import HasInputCol, HasOutputCol
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType
```

In [24]:

```
class BsTextExtractor(Transformer, HasInputCol, HasOutputCol):

    @keyword_only
    def __init__(self, inputCol=None, outputCol=None):
        super(BsTextExtractor, self).__init__()
        kwargs = self._input_kwargs
        self.setParams(**kwargs)

    @keyword_only
    def setParams(self, inputCol=None, outputCol=None):
        kwargs = self._input_kwargs
        return self._set(**kwargs)

    def _transform(self, dataset):

        def f(s):
            cleaned_post = BeautifulSoup(s).text
            return cleaned_post

        t = StringType()
        out_col = self.getOutputCol()
        in_col = dataset[self.getInputCol()]
        return dataset.withColumn(out_col, udf(f, t)(in_col))
```

In [25]:

```
from pyspark.ml.feature import Tokenizer,StopWordsRemover, CountVectorizer,IDF,St
text_extractor = BsTextExtractor(inputCol="post", outputCol="cleaned_post")
tokenizer = Tokenizer(inputCol="cleaned_post", outputCol="token_text")
stopremove = StopWordsRemover(inputCol='token_text',outputCol='stop_tokens')
count_vec = CountVectorizer(inputCol='stop_tokens',outputCol='c_vec')
idf = IDF(inputCol="c_vec", outputCol="tf_idf")
class_to_num = StringIndexer(inputCol='tags',outputCol='label')
```

In [26]:

```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.linalg import Vector
```

In [27]:

```
clean_up = VectorAssembler(inputCols=['tf_idf','length'],outputCol='features')
```

The Model

We'll use Naive Bayes, but feel free to play around with this choice!

In [28]:

```
from pyspark.ml.classification import NaiveBayes
```

In [29]:

```
# Use defaults
nb = NaiveBayes()
```

Pipeline

In [30]:

```
from pyspark.ml import Pipeline
```

In [31]:
```
data_prep_pipe = Pipeline(stages=[class_to_num,text_extractor,tokenizer,stopremov
```

In [32]:
```
cleaner = data_prep_pipe.fit(data)
```

In [33]:
```
clean_data = cleaner.transform(data)
```

Training and Evaluation!

In [34]:
```
clean_data = clean_data.select(['label','features'])
```

In [35]:
```
clean_data.show()
```

```
+-----+--------------------+
|label|            features|
+-----+--------------------+
| 14.0|(262145,[0,1,2,3,...|
|  3.0|(262145,[0,12,31,...|
| 16.0|(262145,[0,1,2,3,...|
|  8.0|(262145,[0,18,21,...|
|  9.0|(262145,[0,1,4,8,...|
|  3.0|(262145,[0,12,21,...|
| 11.0|(262145,[0,1,3,6,...|
| 18.0|(262145,[0,44,61,...|
|  6.0|(262145,[0,1,14,2...|
| 11.0|(262145,[0,1,3,4,...|
| 14.0|(262145,[0,2,3,6,...|
|  1.0|(262145,[0,18,27,...|
|  3.0|(262145,[0,7,12,1...|
| 13.0|(262145,[0,1,2,3,...|
|  3.0|(262145,[0,11,27,...|
|  8.0|(262145,[0,187,23...|
| 16.0|(262145,[0,10,15,...|
|  6.0|(262145,[0,1,3,12...|
| 18.0|(262145,[0,30,39,...|
|  0.0|(262145,[0,12,15,...|
+-----+--------------------+
only showing top 20 rows
```

In [36]:
```
(training,testing) = clean_data.randomSplit([0.7,0.3], seed=142)
```

In [41]:
```
predictor = nb.fit(training)
```

In [42]:
```
test_results = predictor.transform(testing)
```

In [43]:
```
test_results.show()
```

```
+-----+--------------------+--------------------+--------------------+---------
-+
|label|            features|       rawPrediction|         probability|predictio
n|
+-----+--------------------+--------------------+--------------------+---------
-+
|  0.0|(262145,[0,1,4,11...|[-21169.668465935...|[1.21437232270286...|      12.
0|
|  0.0|(262145,[0,1,4,14...|[-1167.4513441585...|[1.0,7.6094842671...|       0.
0|
|  0.0|(262145,[0,1,5,14...|[-11564.690744964...|[1.0,0.0,0.0,0.0,...|       0.
0|
|  0.0|(262145,[0,1,5,14...|[-10439.407817798...|[1.93390066856389...|      12.
0|
|  0.0|(262145,[0,1,7,9,...|[-7336.5344958838...|[1.0,1.5922509763...|       0.
0|
|  0.0|(262145,[0,1,7,11...|[-8486.4019708681...|[1.0,2.0825090588...|       0.
0|
|  0.0|(262145,[0,1,9,10...|[-3738.5573599120...|[7.30286711565549...|      12.
0|
|  0.0|(262145,[0,1,9,12...|[-6617.1798543878...|[0.99995740820888...|       0.
0|
|  0.0|(262145,[0,1,9,12...|[-1693.4933675940...|[2.32251400846856...|      12.
0|
|  0.0|(262145,[0,1,10,1...|[-3989.2579384985...|[1.0,3.3928388326...|       0.
0|
|  0.0|(262145,[0,1,10,1...|[-2373.9652840133...|[1.0,2.7652251546...|       0.
0|
|  0.0|(262145,[0,1,11,1...|[-8818.6290325138...|[1.0,1.4580653626...|       0.
0|
|  0.0|(262145,[0,1,11,1...|[-4285.8899650084...|[1.0,9.2886878940...|       0.
0|
|  0.0|(262145,[0,1,11,1...|[-2804.2071232226...|[1.0,3.5487447155...|       0.
0|
|  0.0|(262145,[0,1,11,1...|[-2115.4985343601...|[6.48717059350829...|       4.
0|
|  0.0|(262145,[0,1,12,1...|[-2828.9699626115...|[0.13219295903883...|      12.
0|
|  0.0|(262145,[0,1,12,1...|[-5997.8626004770...|[1.0,1.4086114099...|       0.
0|
|  0.0|(262145,[0,1,12,1...|[-3875.2807055205...|[1.0,2.0818433425...|       0.
0|
|  0.0|(262145,[0,1,12,1...|[-5370.3720000707...|[1.0,2.0080477642...|       0.
0|
|  0.0|(262145,[0,1,12,1...|[-7260.3672598605...|[1.0,1.6809702518...|       0.
0|
+-----+--------------------+--------------------+--------------------+---------
-+
only showing top 20 rows
```

In [44]:

```
# Create a confusion matrix
test_results.groupBy('label', 'prediction').count().show()
```

```
+-----+----------+-----+
|label|prediction|count|
+-----+----------+-----+
|  8.0|       3.0|   53|
| 16.0|       8.0|   20|
| 19.0|       5.0|    1|
| 10.0|       1.0|    3|
| 12.0|       5.0|    2|
|  0.0|      12.0|  169|
|  0.0|       8.0|   17|
|  1.0|      19.0|    9|
|  1.0|      12.0|    1|
|  7.0|       3.0|   11|
| 15.0|      11.0|   12|
| 19.0|      12.0|    1|
| 17.0|       7.0|   12|
| 17.0|      19.0|   15|
| 11.0|      17.0|    7|
|  8.0|       6.0|    1|
| 17.0|       9.0|    6|
|  4.0|       6.0|    1|
|  3.0|       9.0|    3|
|  3.0|       5.0|    1|
+-----+----------+-----+
only showing top 20 rows
```

In [45]:

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

In [46]:

```
acc_eval = MulticlassClassificationEvaluator()
acc = acc_eval.evaluate(test_results)
print("Accuracy of model at predicting: {}".format(acc))
```

```
Accuracy of model at predicting: 0.7148108175153408
```

In [48]:

```
# save hdfs
nb.save("hdfs://172.24.40.251:19000/NB_TagFilters_model")
```

- Not very good result! (~72%)
- Solution: Try switching out the classification models! Or even try to come up with other engineered features!...

Use LogisticRegression/Random Forest

Logistic Regression

In [49]:

```
from pyspark.ml.classification import RandomForestClassifier, LogisticRegression
```

In [50]:
```
lg = LogisticRegression(maxIter=20, regParam=0.3, elasticNetParam=0)
```

In [51]:
```
predictor_1 = lg.fit(training)
```

In [52]:
```
test_results_1 = predictor_1.transform(testing)
```

In [53]:
```
# Create a confusion matrix
test_results_1.groupBy('label', 'prediction').count().show()
```

```
+-----+----------+-----+
|label|prediction|count|
+-----+----------+-----+
|  8.0|       3.0|   47|
| 16.0|       8.0|   10|
| 19.0|       5.0|    1|
| 10.0|       1.0|    1|
|  2.0|       0.0|    2|
| 15.0|      16.0|    1|
| 12.0|       5.0|    3|
|  0.0|      12.0|   94|
|  0.0|       8.0|    8|
|  1.0|      19.0|   12|
|  1.0|      12.0|    3|
|  7.0|       3.0|    6|
| 15.0|      11.0|    5|
| 19.0|      12.0|    5|
| 11.0|      17.0|   15|
| 17.0|       7.0|   13|
| 17.0|      19.0|    2|
| 17.0|       9.0|   11|
|  8.0|       6.0|    5|
|  6.0|       1.0|    2|
+-----+----------+-----+
only showing top 20 rows
```

In [54]:
```
acc_eval = MulticlassClassificationEvaluator()
acc_1 = acc_eval.evaluate(test_results_1)
print("Accuracy of model at predicting: {}".format(acc_1))
```

```
Accuracy of model at predicting: 0.7182544021412621
```

In [55]:
```
## It's not better result!!!
```

In [57]:
```
# save hdfs
lg.save("hdfs://172.24.40.251:19000/LG_TagFilters_model")
```

Random forest

In [58]:

```
rf = RandomForestClassifier(labelCol="label", \
                            featuresCol="features", \
                            numTrees = 500, \
                            maxDepth = 5, \
                            maxBins = 64)
```

In [59]:

```
predictor_2 = rf.fit(training)
```

In [60]:

```
test_results_2 = predictor_2.transform(testing)
```

In [61]:

```
# Create a confusion matrix
test_results_2.groupBy('label', 'prediction').count().show()
```

```
+-----+----------+-----+
|label|prediction|count|
+-----+----------+-----+
|  8.0|       3.0|   33|
| 16.0|       8.0|   31|
| 19.0|       5.0|    1|
|  0.0|      12.0|  268|
|  1.0|      19.0|    5|
|  1.0|      12.0|   19|
|  0.0|       8.0|    3|
| 19.0|      12.0|   28|
|  7.0|       3.0|    1|
| 15.0|      11.0|    1|
| 11.0|      17.0|    4|
| 17.0|       7.0|   16|
|  8.0|       6.0|    4|
| 17.0|       9.0|    5|
|  6.0|       1.0|    1|
|  3.0|       9.0|    1|
|  4.0|       6.0|    1|
|  6.0|       8.0|    9|
| 15.0|      19.0|    1|
| 14.0|       7.0|    3|
+-----+----------+-----+
only showing top 20 rows
```

In [62]:
```
test_results_2.groupBy('prediction').count().show()
```

```
+----------+-----+
|prediction|count|
+----------+-----+
|       8.0| 1046|
|       0.0|  375|
|       7.0|  582|
|      18.0|  426|
|       1.0|  524|
|       4.0|  762|
|      11.0|  580|
|      14.0|  251|
|       3.0|  491|
|      19.0|  396|
|       2.0|  647|
|      17.0|  402|
|      10.0|  624|
|      13.0|  720|
|       6.0|  578|
|      15.0|  605|
|       5.0|  520|
|       9.0|  592|
|      16.0|  656|
|      12.0| 1113|
+----------+-----+
```

In [63]:
```
acc_eval = MulticlassClassificationEvaluator()
acc_2 = acc_eval.evaluate(test_results_2)
print("Accuracy of model at predicting: {}".format(acc_2))
```

```
Accuracy of model at predicting: 0.7231428382621157
```

In [64]:
```
## It has higher accuracy but is not a better result!!!
```

In [66]:
```
# save hdfs
rf.save("hdfs://172.24.40.251:19000/RF_TagFilters_model")
```

In [67]:
```
sc.stop()
```