databricks ML Example 09-1-23 Notebook 2023-10-09 11:54:18

(https://databricks.com)

```
# Extracting features form the text
some_text = spark.createDataFrame([
    ['''
Apache Spark achieves high performance for both batch
and streaming data, using a state-of-art DAG scheduler,
a query optimizer, and a physical execution engine.
''']
,['''
Apache Spark is a fast and general-purpose cluseter computing
system. It provides high-level APIs in Java, Scala, Python
and R, and an optimized engine that supports general execution
graphs. It also supports a rich set of higher-level tools including
Spark SQL for SQL and structured data processing, MLlib for machine
learning, GraphX for graph processing, and Spark Streaming.
''']
,['''
Machine learning is a filed of computer science that often uses
statistical techniques to give computers the ability to "learn"
(i.e., progessively improve performance on a specific task)
with data, without being explicitly programmed.
''']
], ['text'])


import pyspark.ml.feature as feat
import pyspark.sql.functions as f
```

```
splitter = feat.RegexTokenizer(
    inputCol = 'text',
    outputCol = 'text_split',
    pattern = '\s+|[,.\"]'
)
```

```
splitter.transform(some_text).select('text_split').take(1)
```

```
Out[7]: [Row(text_split=['apache', 'spark', 'achieves', 'high', 'performance', 'for', 'both', 'batch', 'and', 'streaming', 'data',
'using', 'a', 'state-of-art', 'dag', 'scheduler', 'a', 'query', 'optimizer', 'and', 'a', 'physical', 'execution', 'engine'])]
```

```
sw_remover = feat.StopWordsRemover(
    inputCol = splitter.getOutputCol()
    ,outputCol = 'no_stopWords'
)
```

```
sw_remover.transform(splitter.transform(some_text)).select('no_stopWords').take(1)
```

```
Out[9]: [Row(no_stopWords=['apache', 'spark', 'achieves', 'high', 'performance', 'batch', 'streaming', 'data', 'using', 'state-of-ar
t', 'dag', 'scheduler', 'query', 'optimizer', 'physical', 'execution', 'engine'])]
```

```
hasher = feat.HashingTF(
    inputCol = sw_remover.getOutputCol()
    ,outputCol = 'hashed'
    ,numFeatures = 20
)
```

```
hasher.transform(sw_remover.transform(splitter.transform(some_text))).select('hashed').take(1)
```

```
Out[11]: [Row(hashed=SparseVector(20, {0: 1.0, 3: 1.0, 6: 1.0, 8: 2.0, 9: 1.0, 11: 1.0, 12: 1.0, 13: 1.0, 15: 2.0, 16: 2.0, 17: 2.0,
18: 1.0, 19: 1.0}))]
```

```
idf = feat.IDF(
    inputCol= hasher.getOutputCol()
    ,outputCol='features'
)
```

```
idfModel = idf.fit(hasher.transform(sw_remover.transform(splitter.transform(some_text))))
```

```
idfModel.transform(hasher.transform(sw_remover.transform(splitter.transform(some_text)))).select('features').take(1)
```

```
Out[14]: [Row(features=SparseVector(20, {0: 0.0, 3: 0.0, 6: 0.2877, 8: 0.0, 9: 0.2877, 11: 0.2877, 12: 0.0, 13: 0.0, 15: 0.0, 16: 0.
0, 17: 1.3863, 18: 0.6931, 19: 0.0}))]
```

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

```
pipeline = Pipeline(stages=[splitter, sw_remover, hasher, idf])
pipelineModel = pipeline.fit(some_text)
```

```
pipelineModel.transform(some_text).select('text', 'features').take(1)
```

```
Out[17]: [Row(text='\nApache Spark achieves high performance for both batch\nand streaming data, using a state-of-art DAG schedule
r,\na query optimizer, and a physical execution engine.\n', features=SparseVector(20, {0: 0.0, 3: 0.0, 6: 0.2877, 8: 0.0, 9: 0.2877,
11: 0.2877, 12: 0.0, 13: 0.0, 15: 0.0, 16: 0.0, 17: 1.3863, 18: 0.6931, 19: 0.0}))]
```

```
w2v = feat.Word2Vec(
    vectorSize=5
    ,minCount=2
    ,inputCol= sw_remover.getOutputCol()
    ,outputCol='vector'
)
```

```
model = w2v.fit(sw_remover.transform(splitter.transform(some_text)))
```

```
Out[21]: [Row(vector=DenseVector([0.0076, 0.0077, -0.0017, -0.015, -0.004]))]
```