databricks NLP_PySpark Notebook 2023-10-16 10:59:49

(https://databricks.com)

```
# create spark session
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('nlp').getOrCreate()
```

```
df = spark.createDataFrame([(1, 'I Really Liked this movie'),
                            (2, 'I Would recommend this movie to my friends'),
                            (3, 'movie was alright but acting was horrible'),
                            (4, 'I am never watching that movie ever again')],
                           ['user_id', 'review'])
```

```
df.show(5,False)
```

```
+-------+------------------------------------------+
|user_id|review                                    |
+-------+------------------------------------------+
|1      |I Really Liked this movie                 |
|2      |I Would recommend this movie to my friends|
|3      |movie was alright but acting was horrible |
|4      |I am never watching that movie ever again |
+-------+------------------------------------------+
```

```
# tokenization
```

```
from pyspark.ml.feature import Tokenizer
```

```
tokenization = Tokenizer(inputCol='review', outputCol='tokens')
```

```
tokenized_df = tokenization.transform(df)
```

```
tokenized_df.show(4,False)
```

```
+-------+------------------------------------------+---------------------------------------------------+
|user_id|review                                    |tokens                                             |
+-------+------------------------------------------+---------------------------------------------------+
|1      |I Really Liked this movie                 |[i, really, liked, this, movie]                    |
|2      |I Would recommend this movie to my friends|[i, would, recommend, this, movie, to, my, friends]|
|3      |movie was alright but acting was horrible |[movie, was, alright, but, acting, was, horrible]  |
|4      |I am never watching that movie ever again |[i, am, never, watching, that, movie, ever, again] |
+-------+------------------------------------------+---------------------------------------------------+
```

```
# Stopwords removal
```

```
from pyspark.ml.feature import StopWordsRemover
```

```
stopword_removal = StopWordsRemover(inputCol='tokens', outputCol='refined_tokens')
```

```
refined_df = stopword_removal.transform(tokenized_df)
```

```
refined_df.select(['user_id', 'tokens', 'refined_tokens']).show(10,False)
```

```
+-------+------------------------------------------------+---------------------------------+
|user_id|tokens                                          |refined_tokens                   |
+-------+------------------------------------------------+---------------------------------+
|1      |[i, really, liked, this, movie]                 |[really, liked, movie]           |
|2      |[i, would, recommend, this, movie, to, my, friends]|[recommend, movie, friends]       |
|3      |[movie, was, alright, but, acting, was, horrible]  |[movie, alright, acting, horrible]|
|4      |[i, am, never, watching, that, movie, ever, again] |[never, watching, movie, ever]    |
+-------+------------------------------------------------+---------------------------------+
```

```
# Count Vectorizer
```

```
from pyspark.ml.feature import CountVectorizer
```

```
count_vec = CountVectorizer(inputCol='refined_tokens', outputCol='features')
```

```
cv_df= count_vec.fit(refined_df).transform(refined_df)
```

```
cv_df.select(['user_id', 'refined_tokens', 'features']).show(4, False)
```

```
+-------+---------------------------------+--------------------------------+
|user_id|refined_tokens                   |features                        |
+-------+---------------------------------+--------------------------------+
|1      |[really, liked, movie]           |(11,[0,3,7],[1.0,1.0,1.0])      |
|2      |[recommend, movie, friends]      |(11,[0,6,9],[1.0,1.0,1.0])      |
|3      |[movie, alright, acting, horrible]|(11,[0,2,4,5],[1.0,1.0,1.0,1.0]) |
|4      |[never, watching, movie, ever]   |(11,[0,1,8,10],[1.0,1.0,1.0,1.0])|
+-------+---------------------------------+--------------------------------+
```

```
count_vec.fit(refined_df).vocabulary
```

```
Out[23]: ['movie',
 'liked',
 'horrible',
 'friends',
 'ever',
 'alright',
 'recommend',
 'acting',
 'really',
 'never',
 'watching']
```

```
# tf_idf
```

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

```
hashing_vec = HashingTF(inputCol="refined_tokens", outputCol='tf_features')
```

```
hashing_df = hashing_vec.transform(refined_df)
```

```
hashing_df.select(['user_id', 'refined_tokens', 'tf_features']).show(4, False)
```

```
+-------+---------------------------------+-----------------------------------------------------+
|user_id|refined_tokens                   |tf_features                                          |
+-------+---------------------------------+-----------------------------------------------------+
|1      |[really, liked, movie]           |(262144,[99172,210223,229264],[1.0,1.0,1.0])         |
|2      |[recommend, movie, friends]      |(262144,[68228,130047,210223],[1.0,1.0,1.0])         |
|3      |[movie, alright, acting, horrible]|(262144,[95685,171118,210223,236263],[1.0,1.0,1.0,1.0])|
|4      |[never, watching, movie, ever]   |(262144,[63139,113673,203802,210223],[1.0,1.0,1.0,1.0])|
+-------+---------------------------------+-----------------------------------------------------+
```

```
tf_idf_vec = IDF(inputCol='tf_features', outputCol='tf_idf_features')
```

```
tf_idf_df = tf_idf_vec.fit(hashing_df).transform(hashing_df)
```

```
tf_idf_df.select(['user_id', 'tf_idf_features']).show(4, False)
```

```
+-------+----------------------------------------------------------------------------------------------------+
|user_id|tf_idf_features                                                                                     |
+-------+----------------------------------------------------------------------------------------------------+
|1      |(262144,[99172,210223,229264],[0.9162907318741551,0.0,0.9162907318741551])                          |
|2      |(262144,[68228,130047,210223],[0.9162907318741551,0.9162907318741551,0.0])                          |
|3      |(262144,[95685,171118,210223,236263],[0.9162907318741551,0.9162907318741551,0.0,0.9162907318741551])|
|4      |(262144,[63139,113673,203802,210223],[0.9162907318741551,0.9162907318741551,0.9162907318741551,0.0])|
+-------+----------------------------------------------------------------------------------------------------+
```

```
# classification
```

```
text_df = spark.read.csv('dbfs:/FileStore/shared_uploads/devjethva234@gmail.com/Movie_reviews.csv', inferSchema=True,
header=True,sep=',')
```

```
text_df.printSchema()
```

```
root
 |-- Review: string (nullable = true)
 |-- Sentiment: string (nullable = true)
```

```
text_df.count()
```

```
Out[36]: 7087
```

```
from pyspark.sql.functions import rand
```

```
text_df.orderBy(rand()).show(10,False)
```

```
+------------------------------------------------------------------------+---------+
|Review                                                                  |Sentiment|
+------------------------------------------------------------------------+---------+
|Harry Potter is AWESOME I don't care if anyone says differently!..     |1        |
|Harry Potter is AWESOME I don't care if anyone says differently!..     |1        |
|Always knows what I want, not guy crazy, hates Harry Potter..          |0        |
|I either LOVE Brokeback Mountain or think it's great that homosexuality|1        |
|dudeee i LOVED brokeback mountain!!!!                                  |1        |
|I liked the movie Brokeback Mountain.                                  |1        |
|I love Harry Potter.                                                   |1        |
|i hate brokeback mountain!!                                            |0        |
|Brokeback Mountain was an AWESOME movie.                               |1        |
|Harry Potter dragged Draco Malfoy ’ s trousers down past his hips and  |0        |
+------------------------------------------------------------------------+---------+
only showing top 10 rows
```

```
text_df= text_df.filter(((text_df.Sentiment =='1') | (text_df.Sentiment =='0')))
```

```
text_df.count()
```

```
Out[41]: 6990
```

```
text_df.groupBy("Sentiment").count().show()
```

```
+---------+-----+
|Sentiment|count|
+---------+-----+
|        0| 3081|
|        1| 3909|
+---------+-----+
```

```
text_df.printSchema()
```

```
root
 |-- Review: string (nullable = true)
 |-- Sentiment: string (nullable = true)
```

```
text_df = text_df.withColumn("Label", text_df.Sentiment.cast('float')).drop('Sentiment')
```

```
text_df.orderBy(rand()).show(10,False)
```

```
+------------------------------------------------------------------------+-----+
|Review                                                                  |Label|
+------------------------------------------------------------------------+-----+
|I either LOVE Brokeback Mountain or think it's great that homosexuality |1.0  |
|Da Vinci Code sucks.                                                    |0.0  |
|we're gonna like watch Mission Impossible or Hoot.(                     |1.0  |
|I love Brokeback Mountain....                                           |1.0  |
|I would give you more Kudos on this blog but you had to go and talk abou|0.0  |
|So Brokeback Mountain was really depressing.                            |0.0  |
|No matter how much I love Brokeback Mountain, Crash definitely deserved |1.0  |
|I think I hate Harry Potter because it outshines much better reading mat|0.0  |
|friday hung out with kelsie and we went and saw The Da Vinci Code SUCKED|0.0  |
|da vinci code sucks...                                                  |0.0  |
+------------------------------------------------------------------------+-----+
only showing top 10 rows
```

```
text_df.groupBy('label').count().show()
```

```
+-----+-----+
|label|count|
+-----+-----+
|  1.0| 3909|
|  0.0| 3081|
+-----+-----+
```

```
# Add length of the dataframe
from pyspark.sql.functions import length
```

```
text_df = text_df.withColumn('length', length(text_df['Review']))
```

```
text_df.orderBy(rand()).show(10,False)
```

```
+------------------------------------------------------------------------+-----+------+
|Review                                                                  |Label|length|
+------------------------------------------------------------------------+-----+------+
|Da Vinci Code = Up, Up, Down, Down, Left, Right, Left, Right, B, A, SUCK|0.0  |72    |
|I love The Da Vinci Code...                                             |1.0  |27    |
|Combining the opinion / review from Gary and Gin Zen, The Da Vinci Code |0.0  |71    |
|The Da Vinci Code was absolutely AWESOME!                               |1.0  |41    |
|You know, the Harry Potter books are decent enough, and I ’ m glad the  |1.0  |70    |
|Brokeback mountain was beautiful...                                     |1.0  |35    |
|The Da Vinci Code was absolutely AWESOME!                               |1.0  |41    |
|I want to be here because I love Harry Potter, and I really want a place|1.0  |72    |
|I love Brokeback Mountain....                                           |1.0  |29    |
|"I liked the first "" Mission Impossible."                              |1.0  |42    |
+------------------------------------------------------------------------+-----+------+
only showing top 10 rows
```

```
text_df.groupBy('Label').agg({'Length': 'mean'}).show()
```

```
+-----+-----------------+
|Label|      avg(Length)|
+-----+-----------------+
|  1.0|47.61882834484523|
|  0.0|50.95845504706264|
+-----+-----------------+
```

```
# data Cleaning
```

```
tokenization= Tokenizer(inputCol='Review', outputCol='tokens')
```

```
tokenized_df= tokenization.transform(text_df)
```

```
tokenized_df.show()
```

```
+--------------------+-----+------+--------------------+
|              Review|Label|length|              tokens|
+--------------------+-----+------+--------------------+
|The Da Vinci Code...|  1.0|    39|[the, da, vinci, ...|
|this was the firs...|  1.0|    72|[this, was, the, ...|
|i liked the Da Vi...|  1.0|    32|[i, liked, the, d...|
|i liked the Da Vi...|  1.0|    32|[i, liked, the, d...|
|I liked the Da Vi...|  1.0|    72|[i, liked, the, d...|
|that's not even a...|  1.0|    72|[that's, not, eve...|
|I loved the Da Vi...|  1.0|    72|[i, loved, the, d...|
|i thought da vinc...|  1.0|    57|[i, thought, da, ...|
|The Da Vinci Code...|  1.0|    45|[the, da, vinci, ...|
|I thought the Da ...|  1.0|    51|[i, thought, the,...|
|The Da Vinci Code...|  1.0|    68|[the, da, vinci, ...|
|The Da Vinci Code...|  1.0|    62|[the, da, vinci, ...|
|then I turn on th...|  1.0|    66|[then, i, turn, o...|
|The Da Vinci Code...|  1.0|    34|[the, da, vinci, ...|
|i love da vinci c...|  1.0|    24|[i, love, da, vin...|
|i loved da vinci ...|  1.0|    23|[i, loved, da, vi...|
|TO NIGHT:: THE DA...|  1.0|    52|[to, night::, the...|
|THE DA VINCI CODE...|  1.0|    40|[the, da, vinci, ...|
```

```
stopword_removal = StopWordsRemover(inputCol='tokens', outputCol='refined_tokens')
```

```
refined_text_df = stopword_removal.transform(tokenized_df)
```

```
refined_text_df.show()
```

```
+--------------------+-----+------+--------------------+--------------------+
|              Review|Label|length|              tokens|      refined_tokens|
+--------------------+-----+------+--------------------+--------------------+
|The Da Vinci Code...|  1.0|    39|[the, da, vinci, ...|[da, vinci, code,...|
|this was the firs...|  1.0|    72|[this, was, the, ...|[first, clive, cu...|
|i liked the Da Vi...|  1.0|    32|[i, liked, the, d...|[liked, da, vinci...|
|i liked the Da Vi...|  1.0|    32|[i, liked, the, d...|[liked, da, vinci...|
|I liked the Da Vi...|  1.0|    72|[i, liked, the, d...|[liked, da, vinci...|
|that's not even a...|  1.0|    72|[that's, not, eve...|[even, exaggerati...|
|I loved the Da Vi...|  1.0|    72|[i, loved, the, d...|[loved, da, vinci...|
|i thought da vinc...|  1.0|    57|[i, thought, da, ...|[thought, da, vin...|
|The Da Vinci Code...|  1.0|    45|[the, da, vinci, ...|[da, vinci, code,...|
|I thought the Da ...|  1.0|    51|[i, thought, the,...|[thought, da, vin...|
|The Da Vinci Code...|  1.0|    68|[the, da, vinci, ...|[da, vinci, code,...|
|The Da Vinci Code...|  1.0|    62|[the, da, vinci, ...|[da, vinci, code,...|
|then I turn on th...|  1.0|    66|[then, i, turn, o...|[turn, light, rad...|
|The Da Vinci Code...|  1.0|    34|[the, da, vinci, ...|[da, vinci, code,...|
|i love da vinci c...|  1.0|    24|[i, love, da, vin...|[love, da, vinci,...|
|i loved da vinci ...|  1.0|    23|[i, loved, da, vi...|[loved, da, vinci...|
|TO NIGHT:: THE DA...|  1.0|    52|[to, night::, the...|[night::, da, vin...|
```

```
from pyspark.sql.functions import udf
from pyspark.sql.types import IntegerType
from pyspark.sql.functions import *
```

```
len_udf = udf(lambda s: len(s), IntegerType())

refined_text_df = refined_text_df.withColumn("token_count", len_udf(col('refined_tokens')))
```

```
refined_text_df.orderBy(rand()).show(10)
```

```
+--------------------+-----+------+--------------------+--------------------+-----------+
|              Review|Label|length|              tokens|      refined_tokens|token_count|
+--------------------+-----+------+--------------------+--------------------+-----------+
|The Da Vinci Code...|  1.0|    30|[the, da, vinci, ...|[da, vinci, code,...|          4|
|i heard da vinci ...|  0.0|    53|[i, heard, da, vi...|[heard, da, vinci...|          9|
|Which is why i sa...|  1.0|    72|[which, is, why, ...|[said, silent, hi...|          8|
|I used to hate Ha...|  0.0|    28|[i, used, to, hat...|[used, hate, harr...|          4|
|I finished The Da...|  1.0|    54|[i, finished, the...|[finished, da, vi...|          6|
|Then snuck into B...|  0.0|    72|[then, snuck, int...|[snuck, brokeback...|          5|
|""" I hate Harry ...|  0.0|    25|[""", i, hate, ha...|[""", hate, harry...|          4|
|Brokeback Mountai...|  0.0|    37|[brokeback, mount...|[brokeback, mount...|          3|
|we're gonna like ...|  1.0|    51|[we're, gonna, li...|[gonna, like, wat...|          6|
|Brokeback Mountai...|  0.0|    30|[brokeback, mount...|[brokeback, mount...|          3|
+--------------------+-----+------+--------------------+--------------------+-----------+
only showing top 10 rows
```

```
count_vec = CountVectorizer(inputCol='refined_tokens', outputCol='features')
```

```
cv_text_df = count_vec.fit(refined_text_df).transform(refined_text_df)
```

```
cv_text_df.select(['refined_tokens', 'token_count','features','Label']).show(10)
```

```
+--------------------+-----------+--------------------+-----+
|      refined_tokens|token_count|            features|Label|
+--------------------+-----------+--------------------+-----+
|[da, vinci, code,...|          5|(2302,[0,1,4,43,2...|  1.0|
|[first, clive, cu...|          9|(2302,[11,51,229,...|  1.0|
|[liked, da, vinci...|          5|(2302,[0,1,4,52,3...|  1.0|
|[liked, da, vinci...|          5|(2302,[0,1,4,52,3...|  1.0|
|[liked, da, vinci...|          8|(2302,[0,1,4,52,7...|  1.0|
|[even, exaggerati...|          6|(2302,[46,229,272...|  1.0|
|[loved, da, vinci...|          8|(2302,[0,1,22,30,...|  1.0|
|[thought, da, vin...|          7|(2302,[0,1,4,228,...|  1.0|
|[da, vinci, code,...|          6|(2302,[0,1,4,33,2...|  1.0|
|[thought, da, vin...|          7|(2302,[0,1,4,223,...|  1.0|
+--------------------+-----------+--------------------+-----+
only showing top 10 rows
```

```
# select data for building work
model_text_df = cv_text_df.select(['features', 'token_count','Label'])
```

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

```
df_assembler = VectorAssembler(inputCols=['features', 'token_count'], outputCol='features_vec')
model_text_df =df_assembler.transform(model_text_df)
```

```
model_text_df.printSchema()
```

```
root
 |-- features: vector (nullable = true)
 |-- token_count: integer (nullable = true)
 |-- Label: float (nullable = true)
 |-- features_vec: vector (nullable = true)
```

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

```
# splitting tha train data
training_df , test_df = model_text_df.randomSplit([0.75, 0.25])
```

```
training_df.groupBy('Label').count().show()
```

```
+-----+-----+
|Label|count|
+-----+-----+
|  1.0| 2916|
|  0.0| 2296|
+-----+-----+
```

```
test_df.groupBy('Label').count().show()
```

```
+-----+-----+
|Label|count|
+-----+-----+
|  1.0|  993|
|  0.0|  785|
+-----+-----+
log_reg = LogisticRegression(featuresCol = 'features_vec', labelCol = 'Label').fit(training_df)
```

```
results = log_reg.evaluate(test_df).predictions
```

```
results.show()
```

```
+--------------------+-----------+-----+--------------------+--------------------+--------------------+----------+
|            features|token_count|Label|        features_vec|       rawPrediction|         probability|prediction|
+--------------------+-----------+-----+--------------------+--------------------+--------------------+----------+
|(2302,[0,1,4,5,30...|          5|  1.0|(2303,[0,1,4,5,30...|[-20.033223790660...|[1.99379935936333...|       1.0|
|(2302,[0,1,4,5,36...|          5|  1.0|(2303,[0,1,4,5,36...|[-36.496611275475...|[1.41163726618194...|       1.0|
|(2302,[0,1,4,5,75...|          5|  1.0|(2303,[0,1,4,5,75...|[-24.189057523923...|[3.12482567921014...|       1.0|
|(2302,[0,1,4,5,10...|          6|  1.0|(2303,[0,1,4,5,10...|[-24.600819449628...|[2.07014069441782...|       1.0|
|(2302,[0,1,4,12,1...|         10|  1.0|(2303,[0,1,4,12,1...|[-25.485560392607...|[8.54597766156175...|       1.0|
|(2302,[0,1,4,12,1...|          5|  1.0|(2303,[0,1,4,12,1...|[-30.350012582238...|[6.59412248487400...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
```

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
```

```
# confusion matrix
true_positives = results[(results.Label == 1) & (results.prediction == 1)].count()
true_negatives = results[(results.Label == 0) & (results.prediction == 0)].count()
false_positives = results[(results.Label == 0) & (results.prediction == 1)].count()
false_negatives = results[(results.Label == 1) & (results.prediction == 0)].count()
```

```
recall = float(true_positives)/(true_positives + false_negatives)
print(recall)
```

```
0.9798590130916415
```

```
precision = float(true_positives)/(true_positives + false_positives)
print(precision)
```

```
0.9778894472361809
```

```
0.9763779527559056
```