NLP With Pyspark Project Report

Part 1: Large Dataset binary classification with Pyspark ML Lib.

Part 2: Attempt in Multi-Classification Problem with Pyspark Pipeline and BERT

Zhaopu Teng

Dec 5, 2020.

1 Report Part 1:Large Dataset binary classification with Pyspark ML Lib

1.1 Introduction

I get the dataset from https://www.kaggle.com/jmourad100/amazon-product-reviews-data-5core-2014?select=Books_5.json. The size of it is 9GB. It is about the book reviews on amazon. It has a score from 1 to 5. I define the score of 4, 5 is positive, and define them as class 1, while for score 1, 2, 3 is negative, and I define them as class 0. My research question is to predict the review if it is positive or negative. We are expect the models can give us a high accuracy of prediction (higher than 80%). We will evaluate the model by check the confusion matrix, f1 score and accuracy. We can access the correctness of my model by checking the .ipynb file or rerun .py on labtop or cluster. I expect that the model will work well by two part: high accuracy and low time consuming.

1.2 Methodology

On this dataset, I am going to apply three algorithms, they are:

1. Logistic Regression
2. SVM
3. Multilayer Perceptron Classifier

For all of them I am going to set the dictionary vocabulary to 20K and calculate the TFidf matrix for the prediction. The train test split is 7:3, and they all have a max iteration equals to 100.

Especially, in the MLPC model, I have four layer in total. They are input layer with 20K dimensions, 2 hidden layer and each of them has 128 dimensions, and output layer with 2 dimensions.

1.3 Simulation

1.3.1 Perprocessing

The dataset has total 889K lines.

```
File = 'Books_5.json'
df = spark.read.json(File)
```

```
df.count()
```

```
8898041
```

And I put them into the Spark DataFrame and do the calculations. And define score 4 and 5 to be positive and score 1, 2 and 3 to be negative.

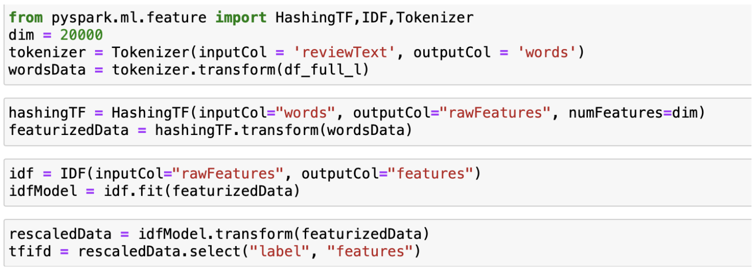
```
pn_udf = udf(lambda x: 1 if x >=4 else 0, IntegerType())
df_full_l = df_full.withColumn('label',pn_udf('overall')).drop('overall')
```

```
df_full_l.show()
```

```
+--------------------+-----+
|          reviewText|label|
+--------------------+-----+
|Spiritually and m...|    1|
|This is one my mu...|    1|
|This book provide...|    1|
|I first read THE ...|    1|
|A timeless classi...|    1|
|Reading this made...|    1|
|As you read, Gibr...|    1|
|Deep, moving dram...|    1|
|This is a timeles...|    1|
|An amazing work. ...|    1|
|I LOVE this book....|    1|
|This book has bee...|    1|
|This book has so ...|    1|
|When I was in col...|    1|
|I discovered The ...|    1|
|Can't say enough ...|    1|
|I have the 1972 v...|    1|
|Anything I've rea...|    1|
|Cool book, I real...|    1|
|This book is ever...|    1|
+--------------------+-----+
only showing top 20 rows
```

1.3.2 TFidf Calculation

Instead of using the method I used in the assignment, I researched and learnt a new approach to calculate TFidf matrix. There are modules in pyspark.ml.feature we can use to find the TFidf matrix.

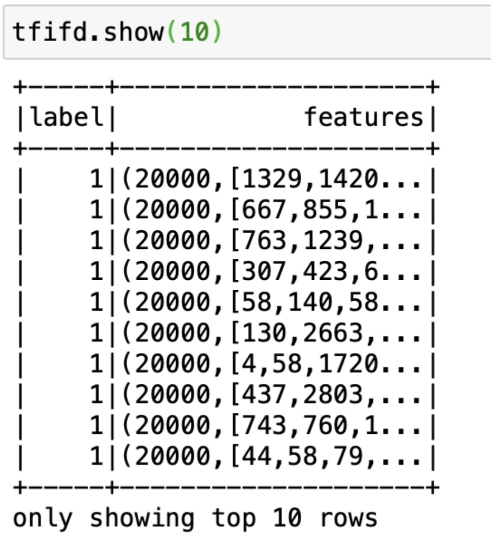


```python
from pyspark.ml.feature import HashingTF,IDF,Tokenizer
dim = 20000
tokenizer = Tokenizer(inputCol = 'reviewText', outputCol = 'words')
wordsData = tokenizer.transform(df_full_l)
```

```python
hashingTF = HashingTF(inputCol="words", outputCol="rawFeatures", numFeatures=dim)
featurizedData = hashingTF.transform(wordsData)
```

```python
idf = IDF(inputCol="rawFeatures", outputCol="features")
idfModel = idf.fit(featurizedData)
```

```python
rescaledData = idfModel.transform(featurizedData)
tfifd = rescaledData.select("label", "features")
```

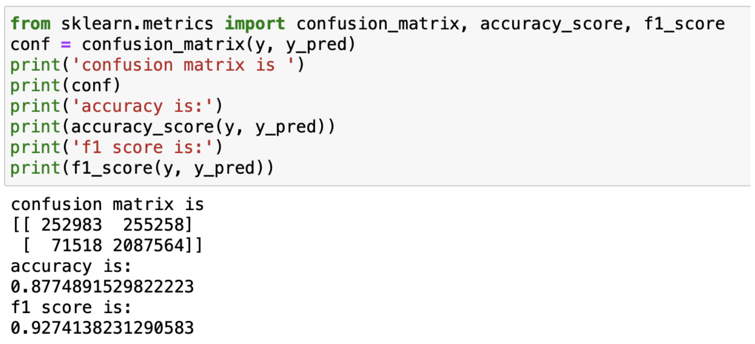


```python
tfifd.show(10)
```

```
+-----+--------------------+
|label|            features|
+-----+--------------------+
|    1|(20000,[1329,1420...|
|    1|(20000,[667,855,1...|
|    1|(20000,[763,1239,...|
|    1|(20000,[307,423,6...|
|    1|(20000,[58,140,58...|
|    1|(20000,[130,2663,...|
|    1|(20000,[4,58,1720...|
|    1|(20000,[437,2803,...|
|    1|(20000,[743,760,1...|
|    1|(20000,[44,58,79,...|
+-----+--------------------+
only showing top 10 rows
```

1.3.3 Logistic Regression

For Logistic Regression, the accuracy is .877, f1 score is .927.

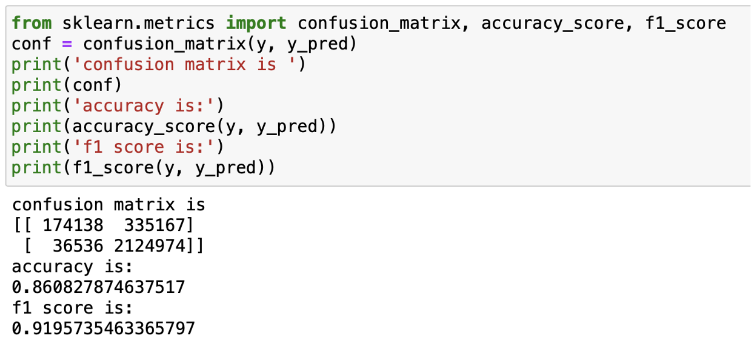


```
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score
conf = confusion_matrix(y, y_pred)
print('confusion matrix is ')
print(conf)
print('accuracy is:')
print(accuracy_score(y, y_pred))
print('f1 score is:')
print(f1_score(y, y_pred))
```

```
confusion matrix is
[[ 252983  255258]
 [  71518 2087564]]
accuracy is:
0.8774891529822223
f1 score is:
0.9274138231290583
```

1.3.4 SVM

For SVM(linear), the accuracy is .861, f1 score is .920.

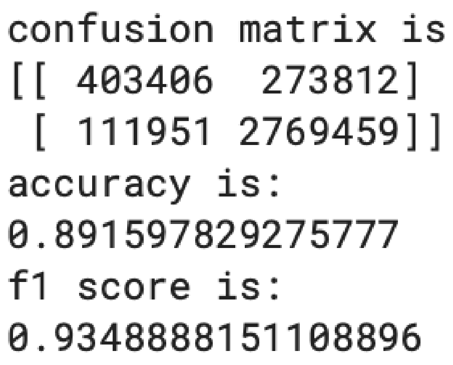


```
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score
conf = confusion_matrix(y, y_pred)
print('confusion matrix is ')
print(conf)
print('accuracy is:')
print(accuracy_score(y, y_pred))
print('f1 score is:')
print(f1_score(y, y_pred))
```

```
confusion matrix is
[[ 174138  335167]
 [  36536 2124974]]
accuracy is:
0.860827874637517
f1 score is:
0.9195735463365797
```

1.3.5 MLPC

For MLPC, the accuracy is .892, f1 score is .935.



```
confusion matrix is
[[ 403406  273812]
 [ 111951 2769459]]
accuracy is:
0.891597829275777
f1 score is:
0.9348888151108896
```

1.4 Conclusion and Discussion

We notice that the MLPC model gives us the performance. However the time consuming is a big problem for the MLPC model. It takes more than 12 hours to calculate the result on the cluster. When we apply the model in the practical problems, we need to balance the importance between the time consuming and performance.

2 Report Part 2: Attempt in Multi-Classification Problem with Pyspark Pipeline and BERT

2.1 Introduction

2.1.1 Sparknlp

Spark-nlp is developed by John Sonw LABS. It is a NLP library built on top of Apache Spark ML. It provides simple, performant accurate NLP annotations for machine learning pipelines, that scale easily in a distributed environment.

2.1.2 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a Transformer-based machine learning technique for natural language processing (NLP) pre-training developed by Google. BERT was created and published in 2018 by Jacob Devlin and his colleagues from Google. As of 2019, Google has been leveraging BERT to better understand user searches.

2.2 Methodology

The techniques and algorithms which are used here shows as below:

1. Sentence preprocessing(such as: Tokenizer, StopWordsCleaner)
2. Word Embedding(BERT model is used)
3. Sentence Embedding
4. Classifier DLApproach
5. Spark Pipeline

For sparknlp library, many classification algorithm we can choose, such as: ClassifierDL, MultiClassifierDL, SentimentDL(max class equal 3), etc. I will choose one of the algorithm with the best performance to show my simulation in next section.

2.3 Simulation for BBC News Dataset

I use a small dataset to test, since it is a very time consuming task. What's more, because of the version of the two libraries, my code right now can only run on Colab. Here is the sample of my data:

```
Dataset.show(10)
```

```
+-------------+--------------------+
|     category|                text|
+-------------+--------------------+
|         tech|tv future in the ...|
|     business|worldcom boss  le...|
|        sport|tigers wary of fa...|
|        sport|yeading face newc...|
|entertainment|ocean s twelve ra...|
|     politics|howard hits back ...|
|     politics|blair prepares to...|
|        sport|henman hopes ende...|
|        sport|wilkinson fit to ...|
|entertainment|last star wars  n...|
+-------------+--------------------+
only showing top 10 rows
```

Next step would build up our pipeline, the pipeline stages shows as below:

```
bert_pipeline = Pipeline(
    stages=[document_assembler,
            tokenizer,
            normalizer,
            stopwords_cleaner,
            lemma,
            word_embeddings,
            embeddingsSentence,
            classifierdl,
    ]
)
```

Then we can train this model by running this pipeline.

2.4 Result

```
from sklearn.metrics import classification_report, accuracy_score
df = bert_pipelineModel.transform(df_test).select('category','text','class.result').toPandas()
df['result'] = df['result'].apply(lambda x: x[0])

print(classification_report(df.category, df.result))
print(accuracy_score(df.category, df.result))
```

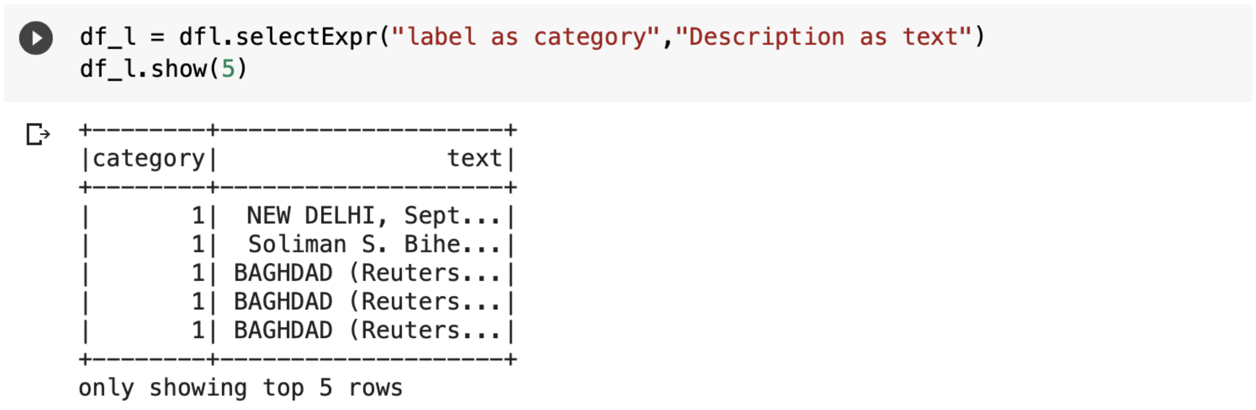
```
               precision    recall  f1-score   support

     business       0.96      0.96      0.96       153
entertainment       0.93      0.94      0.94       109
     politics       0.94      0.94      0.94       109
        sport       0.98      0.99      0.99       153
         tech       0.96      0.92      0.94       116

     accuracy                           0.96       640
    macro avg       0.95      0.95      0.95       640
 weighted avg       0.96      0.96      0.96       640

0.95625
```

We can check the accuracy and f1 score with the table above.

2.5 Simulation for AG News Classification Dataset

I get the data from https://www.kaggle.com/amananandrai/ag-news-classification-dataset?select=train.csv. It has 120K lines, and has four types of a article they are:

1. World
2. Sports
3. Business
4. Sci/Tech



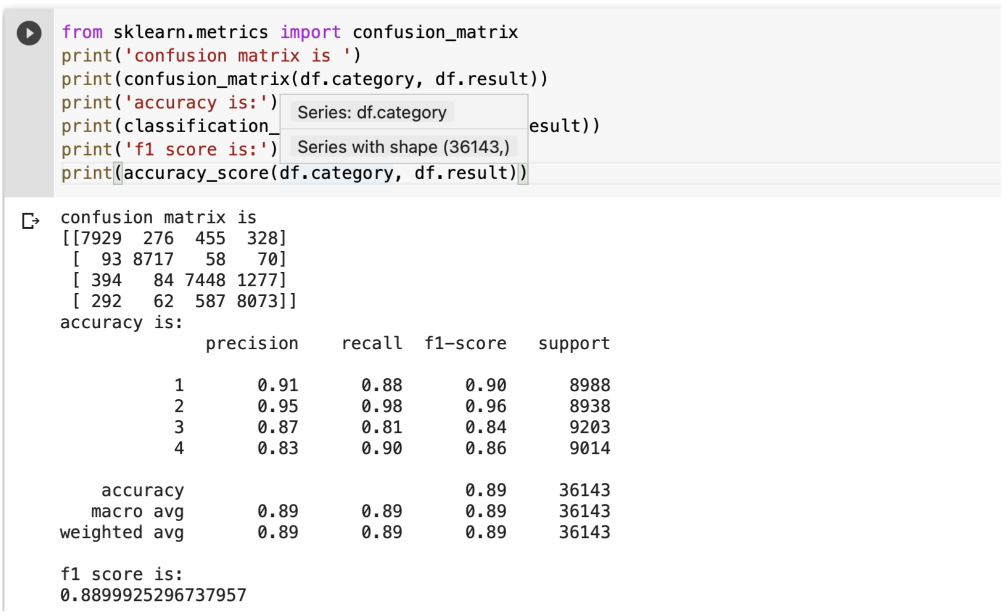
```
df_l = dfl.selectExpr("label as category","Description as text")
df_l.show(5)
```

```
+--------+--------------------+
|category|                text|
+--------+--------------------+
|       1| NEW DELHI, Sept...|
|       1|  Soliman S. Bihe...|
|       1|BAGHDAD (Reuters...|
|       1|BAGHDAD (Reuters...|
|       1|BAGHDAD (Reuters...|
+--------+--------------------+
only showing top 5 rows
```

The pipeline stages show as below:

```
pipeline = Pipeline(
    stages = [
        document,
        tokenizer,
        normalizer,
        stopwords_cleaner,
        word_embeddings,
        use,
        classsifierdl
    ])
```

2.6 Result



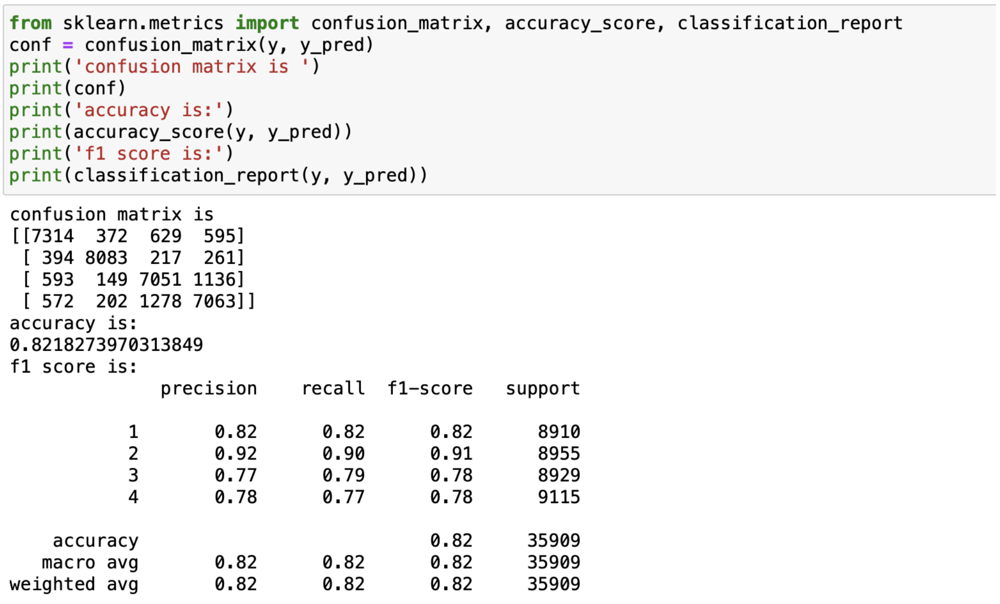
```
from sklearn.metrics import confusion_matrix
print('confusion matrix is ')
print(confusion_matrix(df.category, df.result))
print('accuracy is:')
print(classification_                          esult))
print('f1 score is:')
print(accuracy_score(df.category, df.result))
```

```
confusion matrix is
[[7929  276  455  328]
 [  93 8717   58   70]
 [ 394   84 7448 1277]
 [ 292   62  587 8073]]
accuracy is:
              precision    recall  f1-score   support

           1       0.91      0.88      0.90      8988
           2       0.95      0.98      0.96      8938
           3       0.87      0.81      0.84      9203
           4       0.83      0.90      0.86      9014

    accuracy                           0.89     36143
   macro avg       0.89      0.89      0.89     36143
weighted avg       0.89      0.89      0.89     36143

f1 score is:
0.8899925296737957
```

We can check the accuracy and f1 score with the table above.

2.7 Comparison with Logistic Regression

I use the same dataset under Logistic Regression implement by Pyspark environment. And I get the following result for testing dataset:



```
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
conf = confusion_matrix(y, y_pred)
print('confusion matrix is ')
print(conf)
print('accuracy is:')
print(accuracy_score(y, y_pred))
print('f1 score is:')
print(classification_report(y, y_pred))
```

```
confusion matrix is
[[7314  372  629  595]
 [ 394 8083  217  261]
 [ 593  149 7051 1136]
 [ 572  202 1278 7063]]
accuracy is:
0.8218273970313849
f1 score is:
              precision    recall  f1-score   support

           1       0.82      0.82      0.82      8910
           2       0.92      0.90      0.91      8955
           3       0.77      0.79      0.78      8929
           4       0.78      0.77      0.78      9115

    accuracy                           0.82     35909
   macro avg       0.82      0.82      0.82     35909
weighted avg       0.82      0.82      0.82     35909
```

2.8 Conclusion and Discussion

We can see that for the multi-classification problems, by using BERT which is a Transformer-based machine learning technique, it will give the better accuracy. However, it takes a really long time to run on Colab and since I am still trying to find a way run this on Google cluster, I am concerned about the time consuming problem. To be continue, we can work on finding a better time consuming way to run this code.