Text Analytics and Machine Learning with Spark

Submitted to-
Praxis Business School
Dr. Prithwis Mukerjee

Submitted by-
Aditya Pathania
A21002

Idea behind the problem

The project of NLP was done , while I was studying for the Text analytics , I had done the project in the python after getting into the spark world , I tried to solve the problem in the spark itself

The idea of using NLP and the text , Is different than the other machine learning models

The text analytics , can become a little complicated in times

The idea of the project was to identify the authors that might have written the text

We will follow the approach of separating the words, tokens , stop words and then build our specific model

Then we will check our accuracy for our model with the test set

Tools and techniques used

The tools used for the project are -

- Spark
- Python
- Google colab
- Matplotlib
- NLP: text analysis
- SQL context

Spark is the platform we chose to do the project we had installed the spark and started the spark session on the python environment.

We used matplot to visualize the results

Tools and techniques used

We used the google colab platform to do the project

We used the NLP , as we are doing the text analytics

We also used the SQL context to work on the tables

And then for machine learning we used the logistic regression , which gave us good results being a 3 class data

The test data and the train data has been shared as well with the project as well

Problem Statement

In this notebook, we tried to predict the author of excerpts from horror stories by Edgar Allan Poe, Mary Shelley, and HP Lovecraft. The goal of the notebook is not to get too fancy with the choice of the Algorithms but its more on how can you use Spark to achieve or at least try to achieve what you could do using scikit-learn and pandas. The dataset can be downloaded from the Kaggle competition page.According to the dataset there are three distinct author initials we have already been provided with a mapping of these initials to the actual author which is as follows:

EAP - Edgar Allen Poe

HPL - HP Lovecraft

MWS - Mary Shelley

Description of the Problem

In the dataset we had been given text from some authors , that we need to analyse and then use some modelling technique that can be used to predict what texts are written by which user

We had been given 2 datasets , the training dataset and the testing dataset

We tried to analyse the text , create count vectorizer , token vectorizer , transformed the text into vector and then tried to analyse and predict from the token words who the author of the text can be.

Approach to the Problem

Steps that were followed-

1- Creating a spark session and installing all related packages

2- Importing libraries

3- EDA on the dataset

4- Creating different token and word vectorizer

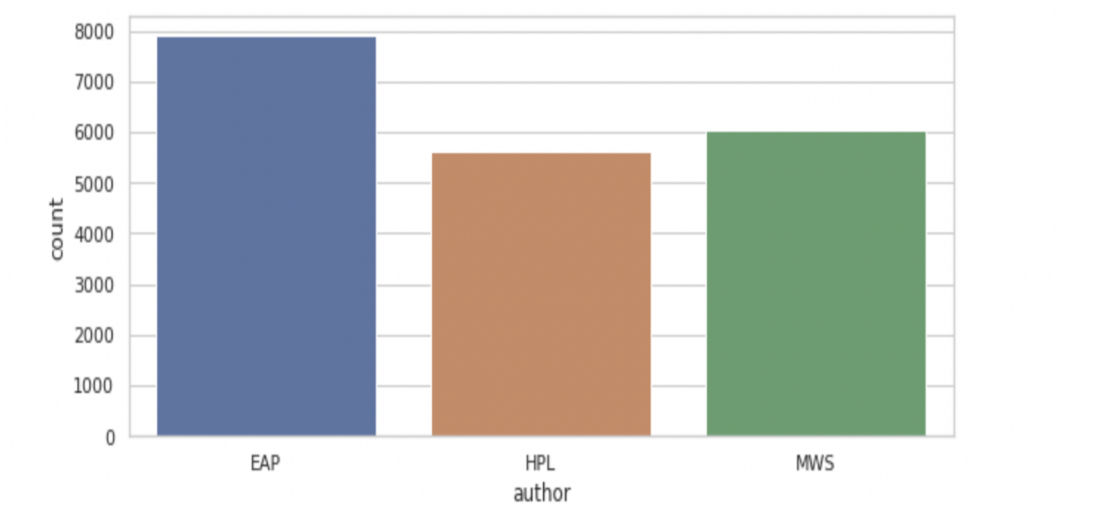
5- Creating hashed words

6- Preparing dataset for the modelling

7- Machine learning

Exploratory Data Analysis-

We started with checking the label , as in how many unique authors we had



As we see there are 3 authors and most of the text count comes from EAP

The lowed count of words are from HPL

We then tried to check some word tokenizers with the Spark

Tokenizer and NLP Transformer

The Idea behind Tokenizer and NLP transformer is that

- It helps to analyse the count of the words
- It picks up the stop words
- It picks up the token words
- It helps to transform the text into word vector

I tried to analyse these libraries on the purpose of learning and analysing how with these work with the Spark session

Now as we are using tokenizer and vectorizer , we will need to also check how the word cloud of the stopwords / tokens for each user will look like

Word cloud / each author

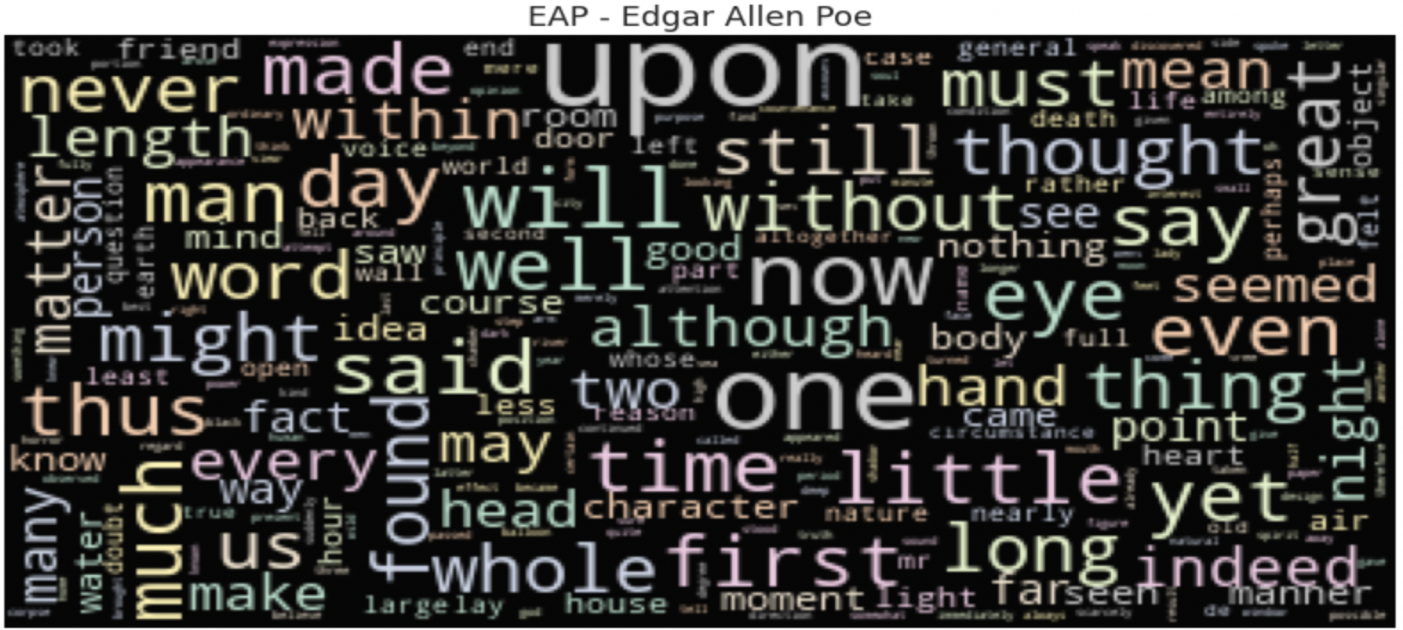
Now as we know we are looking to work for the word cloud with the token vector created but the main issue is -

We need to apply a series of Spark functions to get all the tokens for a given author:Each row in the Dataframe is a token array. So, when we group by each author, we collect tokens as an array of arrays using collect_list function for each author. And, then finally flatten out the array of arrays into one array

```
+------+--------------------------------------------------------------------------+
|author|                                                                all_tokens|
+------+--------------------------------------------------------------------------+
|   MWS|[how, lovely, is, spring, as, we, looked, from, windsor, terrace, on, th...|
|   HPL|[it, never, once, occurred, to, me, that, the, fumbling, might, be, a, m...|
|   EAP|[this, process,, however,, afforded, me, no, means, of, ascertaining, th...|
+------+--------------------------------------------------------------------------+
```

This is how group by was used to gather all the array of tokens by each user then the main step was to create a word cloud

Word cloud-EAP



This is the word cloud for EAP -

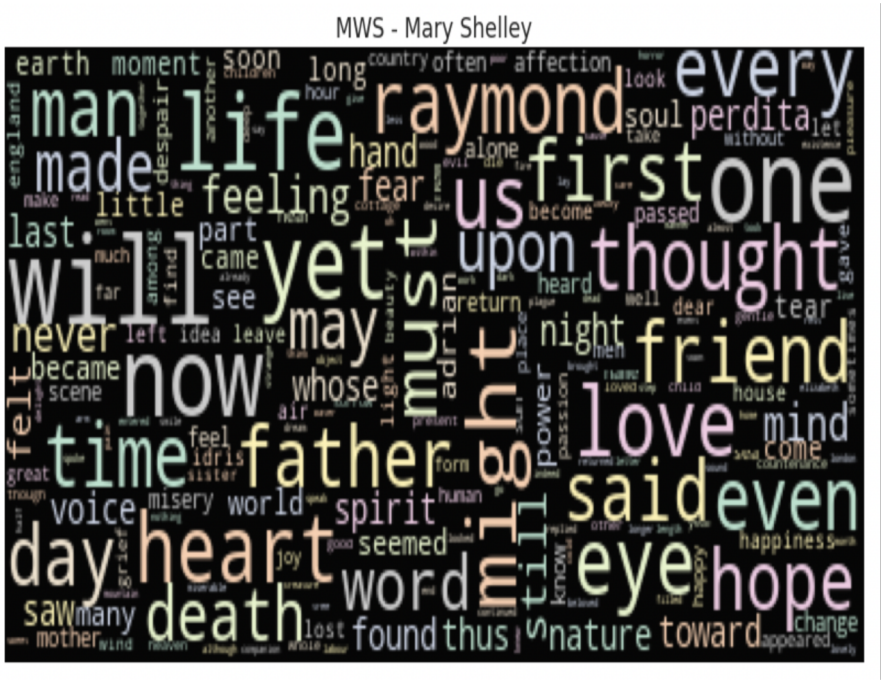
His works revolved around tales of mystery and the grisly and the grim. We can see the usage of body, head, heart in his works.

Word cloud-HPL

This is what the word cloud for HPL , looks like

HP Lovecraft's works revolved around fictional mythology. We can see the usage of dream, eye, strange in his works.

Word cloud - MWS



This is how the word cloud for MWS looks like-

Mary Shelley seemed to be a versatile author. I do not find any distinctive choice of words.

Counting the tokens into hashed feature

The main idea behind this was-

- To identify the tokens and the stop words for each user
- Once identified count the number of tokens
- The tokens count will only help us in the machine learning process
- Once have the count of tokens we save a new feature

We first did it manually and then created a vectorizer

- The main vectorizer used in NLP is tfidf vectorizer

```
+-------+------+------------------------------+------------------------------+------------------------------+
|     id|author|                        tokens|              hashed_features|                tfidf_features|
+-------+------+------------------------------+------------------------------+------------------------------+
|id26305|   EAP|[this, process,, however,, ...|(262144,[239,19036,27576,34...|(262144,[239,19036,27576,34...|
|id17569|   HPL|[it, never, once, occurred,...|(262144,[24284,27576,30950,...|(262144,[24284,27576,30950,...|
|id11008|   EAP|[in, his, left, hand, was, ...|(262144,[25217,28338,42404,...|(262144,[25217,28338,42404,...|
|id27763|   MWS|[how, lovely, is, spring, a...|(262144,[19245,29423,42404,...|(262144,[19245,29423,42404,...|
|id12958|   HPL|[finding, nothing, else,, n...|(262144,[2128,32983,33917,3...|(262144,[2128,32983,33917,3...|
+-------+------+------------------------------+------------------------------+------------------------------+
only showing top 5 rows
```

Advanced tokenizer

Instead of manually doing the tokens , we also have an option to build a custom word tokenizer it includes.

- creating the word tokens
- lemmetize the words
- remove stop words and punctuation
- strip of html and digits.
- count the number of punctuation in the sentence
- count the number of words in the sentence

Advanced tokenizer output

```
def lemmatize(text):
    """
    param: sentence
    return: tokens, tokens count, punctuation count
    """
    wnl = nltk.stem.WordNetLemmatizer()
    stopwords = set(nltk.corpus.stopwords.words('english'))
    list_punct=set(string.punctuation)

    punct_count = sum([1 for ch in text if ch in list_punct])

    text = (unicodedata.normalize('NFKD', text)
            .encode('ascii', 'ignore')
            .decode('utf-8', 'ignore')
            .lower())
    # remove urls
    text = re.sub(r'https?.+|[^(a-zA-Z)(0-9)\s]', ' ', text)
    # remove numbers
    text = re.sub(r'\d+', ' ', text)

    words = text.split()
    # remove stopwords and strings of length <= 2
    words = [wnl.lemmatize(word) for word in words if word not in stopwords and len(word) > 2]
    word_count = len(words)

    return words, word_count, punct_count
```

This is how a custom function was build to separated out all the features mentioned in the previous slide

The features separated will be used for the machine learning process

The hashed features created were applied to test and train both , and then we used the machine learning for the modelling.

Machine learning

Once we have our hashed features separated as the new column in the table then -

- applied the transformation to the train the test set
- created the labels for the authors using String Indexer

0- EAP , 1- MWS, 2 - HPL

Once we have our labels separated

- We created the table separately

Machine learning

text	author	tfidf_features	label
This process, however, afforded me no means of ...	EAP	(262144,[239,19036,27576,34188,42924,50001,5191...	0.0
It never once occurred to me that the fumbling ...	HPL	(262144,[24284,27576,30950,39275,48448,50001,95...	2.0
In his left hand was a gold snuff box, from whi...	EAP	(262144,[25217,28338,42404,48648,49120,55559,57...	0.0
How lovely is spring As we looked from Windsor ...	MWS	(262144,[19245,29423,42404,62400,67416,95889,10...	1.0
Finding nothing else, not even gold, the Superi...	HPL	(262144,[2128,32983,33917,37673,49120,64760,723...	2.0
A youth passed in solitude, my best years spent...	MWS	(262144,[7075,13938,15494,19036,24980,27576,309...	1.0
The astronomer, perhaps, at this point, took re...	EAP	(262144,[51823,95513,95889,99211,100920,108541,...	0.0
The surcingle hung in ribands from my body.	EAP	(262144,[43534,95889,101169,115183,148970,19125...	0.0
I knew that you could not say to yourself 'ster...	EAP	(262144,[991,7014,7608,8449,19036,24980,27576,3...	0.0
I confess that neither the structure of languag...	MWS	(262144,[2564,19036,40789,48448,52105,59554,932...	1.0

only showing top 10 rows

This is how our final dataset looked like , where we had text , author , tfidf_feautes , as discussed earlier the tfidf was used to count and separate the words

Machine learning

- Then we separated the test and the train set in the ratio-75:25
- The authors per each set had the ratio of

```
(labeled_train_df
 .groupBy('label')
 .count()
 .withColumn('%age', F.round(F.col('count') / labeled_train_df.count(), 2))
 .show())
```

```
+-----+-----+----+
|label|count|%age|
+-----+-----+----+
|  0.0| 3288| 0.4|
|  1.0| 2546|0.31|
|  2.0| 2361|0.29|
+-----+-----+----+
```

- The ratio for the labels were checked and used according to the validation set as well

Machine learning

After all the data was prepared we then used the logistic regression model

```
lr = LogisticRegression(featuresCol='tfidf_features',
                        labelCol='label',
                        predictionCol='prediction',
                        probabilityCol='probability',
                        maxIter=20,
                        regParam=0.3,
                        elasticNetParam=0)
```

We took the label as the response variable , and the vectorized transformed input as tfidf_feautes for the x variable

Machine learning

Along with the prediction we also check the probability of text , that these count of the words can appear with such probability

We are getting the predictions with the accuracy of 99 % at the training level of the data , our model has

```
+--------------------+------+--------------------+-----+--------------------+----------+
|                text|author|     tfidf_features|label|         probability|prediction|
+--------------------+------+--------------------+-----+--------------------+----------+
|At such times, al...|   EAP|(262144,[10077,19...|  0.0|[0.88508120206136...|       0.0|
|The great clock o...|   EAP|(262144,[10228,95...|  0.0|[0.85464774389614...|       0.0|
|We kept track of ...|   HPL|(262144,[2622,394...|  2.0|[0.17520778912928...|       2.0|
|There are six can...|   EAP|(262144,[46923,58...|  0.0|[0.82076268911299...|       0.0|
|It was firmly sec...|   EAP|(262144,[11694,30...|  0.0|[0.88749179149987...|       0.0|
+--------------------+------+--------------------+-----+--------------------+----------+
only showing top 5 rows
```

Actually learned well in the terms of training set

We will check this on our testing now .

Machine learning

```
[ ] print('train_f1_score',train_f1_score,'validation_f1_score',validation_f1_score)

    train_f1_score 0.9964611893909074 validation_f1_score 0.7677756780042333
```

We used a mutliclass evaluator for the results and checked the f1 score

We got the f1 score of 99% at the training set and the score of 77 % the testing set

Out model has performed well on the terms of multi class in the dataset

COMMENTS

In the start of the project we discussed our goal for the project , to learn the tokens words , count the token words and predict the author for the text with the count and words vector

We then started our process to visualize the results of the words and the token

We created the count of the tokens , removed stop words

We create tokenizer manually to improve the accuracy of the vectorizer

We then transformed the features into the hashed vector feature

And then we started our process of machine learning

COMMENTS

At the machine learning stage our main focus was to predict the author for the text based on the token and the words count

We also find the probability for the text to appear

Finally we were able to do so with the f1 score of 77 % at the validation set

The idea of starting the project went successfully with the good results