Capstone Project

By Barrett Zhang submitted: 3/2/2023

Application: Fintech

Domain: NLP

Descripton: Find Cryptocurrency mentions in reddit comments. Calculate a sentiment score from the comments to determine positive, negative, or neutral sentiment.

Dataset: Reddit comments sourced from multiple available methods, including Reddit API, pushshift.io, PMAW, SurgeHQ.ai, and Kaggle.

Project Thesis

Crypto has proven to be extremely influenced by social media and influencer opinion. Take from the exmaple where Dogecoin rose 9840% during 2021, due to Elon Musk's involvement. From this a conjecture can be raised about the link between public sentiment and Cryptocurrency value. From what it seems, public sentiment has a direct impact on the value and a resulting effect can be seen on the stock market. There are hundreds of thousands of comments related to crypto posted through Reddit daily, and through this simple pattern we will be utilizing these comments to identify trends in the Cryptocurrency market. This project will use Natural Language Processing to classify a comment as positive, negative, or neutral sentiment. A method to scan reddit comments for Cryptocurrency related information and an API to return the sentiment of the comments will be developed.

Dataset

Reddit comments are avaiable through:

- https://pushshift.io/
- PMAW Python API
- https://www.kaggle.com/datasets/leukipp/reddit-crypto-data

The following subreddits have been scanned:

- r/cryptocurrency
- r/wallstreetbets

Model

The BERT model is a NLP model built for many variations of language needs. It also has functionality in assessing the sentiment of a comment as positive or negative. The question with it is how well trained is it for social media posts, and how much it can be improved through fine-tuning and retraining. A small sample of data was utilized to test the effectiveness of the model.

Model Evaluation Results

Pretrained BERT

BERT is an NLP model for sentiment analysis, short for Bidirectional Encoder Representations from Transformers. Huggingface and transformers has many modules that allow easy utilization of the model. The model is pre-trained from unlabeled text and can be fine-tuned for specific tasks such as this one. Documentation can be found at the following link:

https://huggingface.co/docs/transformers/model_doc/bert

We use the following Python code to read the labeled prediction data into a Pandas dataframe and then use sklearn to generate a confusion matrix and classification report. The model demonstrated was not retrained for our specific task.

Python Code:

```
import pandas as pd
from sklearn import metrics

df = pd.read_csv('Crypto_com.csv')

y_true = df['LABEL_COLUMN']
y_pred = df['predicted']
y_true.value_counts()

print(metrics.accuracy_score(y_true, y_pred))
print(metrics.confusion_matrix(y_true, y_pred, labels = [0, 1, 2]))
print(metrics.classification_report(y_true, y_pred, labels =[0, 1, 2]))
```

```
0.2541750167000668
[[ 463  141   77]
 [1050  228  145]
 [ 604  216   70]]
              precision    recall  f1-score   support

           0       0.22      0.68      0.33       681
           1       0.39      0.16      0.23      1423
           2       0.24      0.08      0.12       890

    accuracy                           0.25      2994
   macro avg       0.28      0.31      0.23      2994
weighted avg       0.31      0.25      0.22      2994
```

In this case 0 represents Negative sentiment, 1 represents Neutral, and 2 represents Positive.

As can be seen we have a model that is poor at predicting overall sentiment but has a higher recall with positive sentiment (68%). The negative predictions and postive predictions are especially poor with precision of 22% and recall of 68% and precision of 24% and recall of 8% respectively. The neutral predictions come at a slightly higher precision of 39% but a low recall of 16%

As and overall assessment we can see that training is necessary to improve predictions for all categories.

Finetuning and Retraining BERT

BERT can be retrained to improve accuracy. The training procedure varies but the procedure used in this case can be seen showcased at:

https://towardsdatascience.com/sentiment-analysis-in-10-minutes-with-bert-and-hugging-face-294e8a04b671

Preparing for training

To prepare for training we must label data that we have extracted. The above metrics were conducted with around 3000 data points that we have labeled.

Here is an example of the first 20 rows, the data has been pre-processed for tokenization:

```
df[['DATA_COLUMN', 'LABEL_COLUMN']].head(20)
```

	DATA_COLUMN	LABEL_COLUMN
0	dad works bitcoin told bitcoin missed earnings	0.0
1	gt bitcoins fixed supply led fans consider aki...	1.0
2	bitcoin printer goes brrrrrrr	1.0
3	twitter enable bitcoin solana xrp crypto payme...	2.0
4	looks like read bunch apologia shady latin ame...	0.0
5	damn crypto like internet 90s mean 30 years bi...	2.0
6	general concepts simple ordinary income capita...	1.0
7	top bitcoin	1.0
8	around 20 bear stearns lehman brothers collaps...	0.0
9	one thing buttcoiners rcc common bitcoin still...	0.0
10	save money go vegan buy bitcoin	2.0
11	bitcoin fine anything used make even secure ba...	2.0
12	work life skills young age not come twice spar...	2.0
13	well bitcoin hit time high russia war started ...	2.0
14	ive looking golden bitcoin finally win life ti...	2.0
15	print money gt inflation gt print money battle...	2.0

We then split the dataframe into 2500 and 494 for training and test respectively.

The train will be used to fine-tune the model while 494 will be used as a validation set.

18	already paid bitcoin rose 100000 christmas las...	2.0

Retrained Model and Results

```
Finally we used the berttest.ipynb notebook to re-train the model. The model is
then saved to the hugging face hub under baz08/crypto-Bert-test folder for
future use.  The model can be found at :

https://huggingface.co/baz08/crypto-Bert-test

and the hyperparameters are as follows:

optimizer: {'name': 'Adam', 'weight_decay': None, 'clipnorm': 1.0,
'global_clipnorm': None, 'clipvalue': None, 'use_ema': False, 'ema_momentum': 0.
99, 'ema_overwrite_frequency': None, 'jit_compile': True,
'is_legacy_optimizer': False, 'learning_rate': 2e-05, 'beta_1': 0.9, 'beta_2':
0.999, 'epsilon': 1e-08, 'amsgrad': False}
training_precision: float32
```

Finally we used the berttest.ipynb notebook to re-train the model. The model is then saved to the hugging face hub under baz08/crypto-Bert-test folder for future use. The model can be found at :

https://huggingface.co/baz08/crypto-Bert-test

and the hyperparameters are as follows:

optimizer: {'name': 'Adam', 'weight_decay': None, 'clipnorm': 1.0, 'global_clipnorm': None, 'clipvalue': None, 'use_ema': False, 'ema_momentum': 0.99, 'ema_overwrite_frequency': None, 'jit_compile': True, 'is_legacy_optimizer': False, 'learning_rate': 2e-05, 'beta_1': 0.9, 'beta_2': 0.999, 'epsilon': 1e-08, 'amsgrad': False} training_precision: float32

```
import pandas as pd
from sklearn import metrics

df = df.iloc[2500:2994, :]

y_true = df['LABEL_COLUMN']
y_pred = df['train_pred']
y_true.value_counts()

print(metrics.accuracy_score(y_true, y_pred))
print(metrics.confusion_matrix(y_true, y_pred, labels = [0, 1, 2]))
print(metrics.classification_report(y_true, y_pred, labels =[0, 1, 2]))
```

```
    0.6619433198380567
    [[ 18  17   3]
     [ 43 275  50]
     [ 17  37  34]]
                  precision    recall  f1-score   support

               0       0.23      0.47      0.31        38
               1       0.84      0.75      0.79       368
               2       0.39      0.39      0.39        88

        accuracy                           0.66       494
       macro avg       0.49      0.54      0.50       494
    weighted avg       0.71      0.66      0.68       494
```

```
df = pd.read_csv('Crypto_com.csv')

y_true = df['LABEL_COLUMN']
y_pred = df['train_pred']
y_true.value_counts()

print(metrics.accuracy_score(y_true, y_pred))
print(metrics.confusion_matrix(y_true, y_pred, labels = [0, 1, 2]))
print(metrics.classification_report(y_true, y_pred, labels =[0, 1, 2]))
```

```
    0.8386773547094188
    [[ 557   95   29]
     [  60 1289   74]
     [  36  189  665]]
                  precision    recall  f1-score   support

               0       0.85      0.82      0.84       681
               1       0.82      0.91      0.86      1423
               2       0.87      0.75      0.80       890

        accuracy                           0.84      2994
       macro avg       0.85      0.82      0.83      2994
    weighted avg       0.84      0.84      0.84      2994
```

Set	Accuracy Score	Negative Recall	Neutral Recall	Positive Recall
Original BERT(full 3000 dataset)	0.25	0.68	0.16	0.08
Retrained BERT(test 494 dataset)	0.66	0.47	0.75	0.39
Retrained BERT(full 3000 dataset	0.84	0.82	0.91	0.75

As can be seen from the tabel above, the overall accuracy went from 25 percent to 66 percent between the original BERT and the Retrained BERT tested on the validation data. In terms of recall, while the negative recall did decrease from 68 percent to 47. The neutral and positive recall drastically increased going from 16 percent to 75 percent and 8 percent to 39 percent respectively.