Capstone Project	Submitted By	Robert Morrison
	Date	4/26/2021

Application: Fintech
Domain: NLP
Description: Find crypto coin mentions in social media comments. Calculate a sentiment score from the comments to determine positive, negative or neutral outlook.
Dataset: Reddit comments are available through several methods, including Reddit API, pushshift.io, PRAW, PSAW and Kaggle

1. Project Thesis

Recent crypto coin memes on social media show how retail investors can band together in large numbers to influence crypto coin prices. This is particularly evident in Dogecoin (DOGE), which has risen 9840% YTD (as of May 12, 2021), due in part to tweets from Elon Musk. Users post thousands of crypto related comments every day on the Reddit platform. How do we identify crypto coins with the most potential to rise in value? This seems like a job ideally suited to machine learning. Sites such as Reddit offer an API for retrieving comments, therefore datasets are readily available. A simple pattern matching algorithm can be used to identify comments that mention crypto coins. This project will use natural language parsing to classify a comment as positive, negative or neutral and assign a "sentiment score". A service will be developed that scans Reddit comments in real-time and an API to list popular crypto coins, based on sentiment score. Candidate models for sentiment analysis are BERT and XLNet.

2. Dataset

Reddit comments are available through:

- https://www.reddit.com/dev/api/
- https://pushshift.io/
- PRAW, PSAW Python API
- https://www.kaggle.com/gpreda/reddit-wallstreetsbets-posts

The following sub-reddits will be scanned:

- /r/cryptomarkets
- /r/cryptocurrency
- /r/cryptocurrencies
- /r/cryptomoonshots
- r/satoshistreetbets

A proof of concept using the Reddit API to retrieve articles and comments and perform basic word frequency analysis is available here:
https://github.com/rrmorris2102/ucsd-mle/tree/main/reddit

3. Model

The VADER model (Valence Aware Dictionary and sEntiment Reasoner) is ideal for assessing the sentiment of each comment as positive or negative. This is a rules-based model that is pre-trained on social media data. The question arises, does the VADER model require supervised learning to tune for Reddit posts? Or is pre-trained social media data good enough? VADER is a rules-based NLP, so doesn't require re-training. It is sufficient to provide a curated list of positive and negative word labels. A small sample of labeled data should be sufficient for testing the effectiveness of the model.

FinBERT is another model designed for sentiment analysis. FinBERT is a natural language processing model based on BERT and trained for the financial domain.

4. Model Evaluation Results

4.1. Pre-trained FinBERT

FinBERT is an NLP model for financial sentiment analysis based on BERT. The model is pre-trained and tuned on financial text. Following are classification results for 1000 Reddit cryptocurrency comments using the FinBERT model. The model was not re-trained. The model classifies sentences into three classes: positive, negative and neutral. We find that training is required to improve predictions for positive and negative sentiment.

Sources:
Model: https://github.com/ProsusAI/finBERT
Dataset:
https://github.com/rrmorris2102/ucsd-mle/raw/main/reddit/sentiment_labels_predictions.csv

We use the following Python code to read the labeled prediction data into a Pandas dataframe, then use sklearn to generate the confusion matrix and classification report.

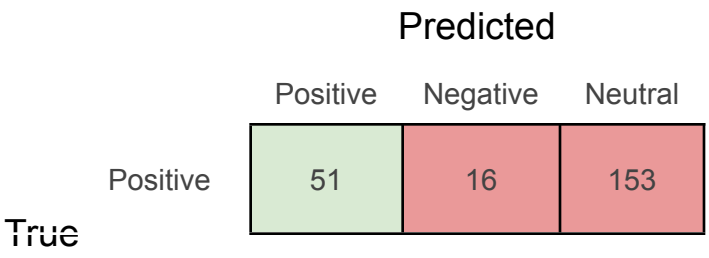
Python Code:

```
import pandas as pd
```

```
from sklearn import metrics

df = pd.read_csv('sentiment_labels_predictions.csv', index_col=0,
low_memory=False)

y_true = df['sentiment']
y_pred = df['prediction']
y_true.value_counts()

print(metrics.accuracy_score(y_true, y_pred))
print(metrics.confusion_matrix(y_true, y_pred, labels=['positive',
'negative', 'neutral']))
print(metrics.classification_report(y_true, y_pred, labels=['positive',
'negative', 'neutral']))
```

Output:

```
0.721
[[ 51  16 153]
 [  0  40  66]
 [ 15  29 630]]
              precision    recall  f1-score   support

    positive       0.77      0.23      0.36       220
    negative       0.47      0.38      0.42       106
     neutral       0.74      0.93      0.83       674

    accuracy                           0.72      1000
   macro avg       0.66      0.51      0.53      1000
weighted avg       0.72      0.72      0.68      1000
```



Negative	0	40	66
Neutral	15	29	630

Insights

Positive Sentiment:
The model is poor overall at identifying comments with positive sentiment (23% recall.) When the model predicts positive sentiment, most of the instances are correctly identified (77% precision.)

Negative Sentiment:
The model is poor overall at identifying comments with negative sentiment (38% recall.) When the model predicts negative sentiment, more than half of the instances are incorrectly identified (47% precision.)

Neutral Sentiment:
The model is excellent overall at identifying comments with neutral sentiment (93% recall.) When the model predicts neutral sentiment, most of the instances are correctly identified (74% precision.)

Overall assessment:
Training is needed to improve predictions for positive and negative sentiment.

Sample Calculation - Positive Sentiment

```
TP = 51
TN = (40 + 66 + 29 + 630) = 765
FP = (0+15) = 15
FN = (16+153) = 169

Precision = TP / (TP + FP) = 51 / (51 + 15) = 0.77
Recall = TP / (TP + FN) = 51 / (51 + 159) = 0.23
F1-Score = 2 * (precision * recall) / (precision + recall) = 2 * (0.77 *
0.23) / (0.77 + 0.23) = 0.35
```

4.2. Retraining FinBERT

The FinBERT model can be retrained to improve accuracy. The training procedure is available at:
https://github.com/ProsusAI/finBERT/blob/master/notebooks/finbert_training.ipynb

4.2.1. Prepare Labeled Data

First, we prepare the file 'train.csv' with labeled data for 1000 sentences. Example (first 20 results):

I have just encountered a Bitcoin scam on a different level,	neutral
Maybe theyll rename themselves once theyve amassed enough BTC	neutral
- Part of NFT sale royalties will be swapped for $UBA on its Uniswap market creating regular buy pressure.	neutral
this clown still buying machine slave produced techno terror btc?	negative
I see almost no justification or explanation of any kind, including the top advice to hold bitcoin.	negative
"Crypto Analyst says bitcoin will be $100k/$300k/$1m/$10m" "Ethereum is the future" "Tron is a scam!!"	positive
"cute animal faces" im looking at you Doge	positive
"During that BTC run" most altcoins will also perform well, the goal in trading is to have a good entry and maximise the return for the risk you take.	neutral
"In the end, it doesnt matter if you have a scarce token when it cannot do much", didnt you just describe bitcoin?	neutral
"Rich Dad Poor Dad" Author Predicts That Bitcoin Could Hit $1.2 Million.	positive

4.2.2. Prepare train/test/split Data

Next we prepare the train/test/split data using the finBERT scripts/datasets.py script:

```
$ python3 scripts/datasets.py  --data_path ./train.csv
```

This creates three files in finBERT's data/sentiment_data folder:

- data/sentiment_data/train.csv
- data/sentiment_data/validation.csv
- data/sentiment_data/test.csv

4.2.3. Retrain Model

Finally we use the finbert_training.ipynb notebook to re-train the model. The re-trained model is written to the ./models/classifier_model/finbert-sentiment folder:

- ./models/classifier_model/finbert-sentiment/config.json
- ./models/classifier_model/finbert-sentiment/pytorch_model.bin

The labels in config.json should also be adjusted as follows:

```
"id2label": {
   "0": "positive",
   "1": "negative",
   "2": "neutral"
 },

 "label2id": {
   "positive": 0,
   "negative": 1,
   "neutral": 2
 },
```

4.2.4. Retrain Results

The overall accuracy score decreased from 0.721 to 0.64. The recall score for positive and negative sentiment improved greatly. The recall score for neutral feedback decreased.

	Original FinBERT (Full 1000 dataset)	Retrained FinBERT (Test 200 dataset)
Accuracy Score	0.721	0.64
Positive Recall	0.23	0.62
Negative Recall	0.38	0.86
Neutral Recall	0.92	0.61

```
Loss:0.66
Accuracy:0.64

Classification Report:
              precision    recall  f1-score   support

           0       0.45      0.62      0.52        45
           1       0.43      0.86      0.58        22
           2       0.86      0.61      0.71       133

    accuracy                           0.64       200
   macro avg       0.58      0.70      0.60       200
weighted avg       0.72      0.64      0.66       200
```

4.3. XLNet

XLNet is an NLP model based on BERT that features a generalized autoregressive (AR) pre-training method. By contrast, BERT is an autoencoder (AE) language model.

For this exercise we adapted the XLnet implementation from Shanay Ghag described at https://medium.com/swlh/using-xlnet-for-sentiment-classification-cfa948e65e85.

A Python module for XLNet is provided by HuggingFace at https://huggingface.co/transformers/model_doc/xlnet.html.

The following changes were made to the original implementation from Shanay Ghag to match the settings of FinBERT:

1. Expand from 2 classes (positive, negative) to 3 classes (positive, negative, neutral)
2. Shorted max sequence length from 512 to 64
3. Batch size increased from 4 to 32
4. Training epochs increased from 3 to 10

In addition, the code was restructured as two Python classes: XLNetSentiment and XLNetSentimentTrain.

The modified version is published here:
https://github.com/rrmorris2102/ucsd-mle/tree/main/xlnet

Following are the training results from the dataset previously used for FinBERT:
https://github.com/rrmorris2102/ucsd-mle/raw/main/reddit/sentiment_labels_predictions.csv

```
              precision    recall  f1-score   support

    positive       0.67      0.60      0.64        48
    negative       0.59      0.42      0.49        24
     neutral       0.79      0.87      0.83       120

    accuracy                           0.74       192
   macro avg       0.68      0.63      0.65       192
weighted avg       0.73      0.74      0.74       192
```

FinBERT vs XLNet Training Results (better results highlighted in yellow) :

		FinBERT	XLNet
Overall	**Accuracy**	0.64	0.74
Positive	**Precision**	0.45	0.67
	Recall	0.62	0.60
	f1-score	0.52	0.64
Negative	**Precision**	0.43	0.59
	Recall	0.86	0.42
	f1-score	0.58	0.49
Neutral	**Precision**	0.86	0.79
	Recall	0.61	0.87
	f1-score	0.71	0.83

XLNet provides better accuracy compared to FinBERT (0.74 vs 0.64.) Predictions for positive and neutral sentiment are better with XLNet, however negative predictions are better with FinBERT.