Creating a Reddit Sentiment Analysis Model using Python, the PRAW Reddit API Wrapper, VADER-Sentiment Tools`1

By Daniel Rusinek

Introduction:

We've all heard it by now... "Bitcoin reaches 50K!", "Cryptocurrency is the future!". But what exactly keeps driving the price of Cryptocurrency? Many have tried to answer this question and in this article we will discuss how to create a simple yet effective sentiment analysis model using Python, the PRAW Python Reddit API Wrapper, and MIT's VADER-Sentiment unsupervised sentiment analysis algorithm to attempt to answer that very question.

Hopefully, after reading this article you will have learned new natural language processing (NLP) skills, learned how to access any desired subreddit via the PRAW Reddit API Wrapper, and display results. It is my goal that anyone, from an advanced Machine learning (ML) and Python user, to a beginner can follow along.

Note: For those already familiar with PRAW API wrapper feel free to skip down to **Filtering Comments** section.

What is Sentiment Analysis?

First off, what exactly is sentiment analysis and why is it useful in cryptocurrency data analysis? Sentiment analysis is the use NLP, and ML (both supervised and unsupervised) methods, in order to identify and quantify textual information on social media and other platforms. It has gained widespread popularity as a way to determine everything from popular topics being discussed on social media, to survey response, healthcare applications, and so on.

In this digital age, it is of the utmost importance to track online sentiment in order to gain industry insight, gain information to improve services and provide an overall improved User Experience (UX).

Why VADER-Sentiment-Analysis?

According the VADER (Valence Aware Dictionary and sEntiment Reasoner) GitHub page, it is "lexicon and rule based unsupervised sentiment analysis tool" available for free through an MIT License (Hutto, & Gilbert, 2014). See https://github.com/cjhutto/vaderSentiment. So why should you choose this model?:

- VADER-Sentiment is an unsupervised model

Because this model is unsupervised it does not require previous training, and can handle textual data in any language, making it ideal for use in social media sentiment analysis in our increasingly connected world. A supervised model such as a Neural Network, or the popular BERT model, would require a large pre-classified dataset . For example:

Comment	0 (negative comment) 1 (positive comment)
"Bitcoin is awesome!"	1
"Dogecoin sucks!"	0
"Polkadot is my favorite color"	1
"Ethereum is a type of crypto-currency"	0

In order to properly train a classified sentiment analysis model, one would require a large dataset with hundreds of thousands to potentially millions of data points like the example above. The next step would be to vectorize each comment into a numerical representation, feed it through the model, which would then ideally recognize patterns that give the optimum results based on the given training set.

NOTE the third comment. While it does mention the word Polkadot, the context of the comment is off-topic, but the model may mistake it for a valid data-point. Also note how the fourth row is a fairly neutral comment which does not give much meaningful information, but could be misclassified as a positive or neutral comment depending on the methods used to create the data set. Let's try that again...

Comment	0 (negative comment) ,1 (positive comment), 2 (neutral comment)
"Bitcoin is awesome!"	1
"Dogecoin sucks!"	0
"Polkadot is my favorite color" (Not meaningful)	1
"Ethereum is a type of crypto-currency"	2

Moreover, as we pointed out in the third entry, the model can get tripped up if the training set is not classified properly. Some companies can get around this by leveraging their existing user base to classify data for them (You ever wondered why Google always forces you to look for the picture that does not have a stop sign in it?)

Unfortunately, most of use do not have access to those type of resources. This is where Vader-Sentiment saves the day. Vader-Sentiment classifies textual information on a scale between 0-1 with a score between -0.05 - 0.05 classified as a neutral comment (Hutto & Gilbert, 2014).

- VADER-Sentiment is easy to install and intuitive

VADER Sentiment can be installed as easy as typing:

```
pip install vaderSentiment
```

and who doesn't like easy?

Now that some background is out of the way, lets talk accessing PRAW, collecting comments, and obtaining results.

Using the PRAW Reddit API Wrapper

Check here for instructions on installing the Wrapper https://praw.readthedocs.io/en/v4.0.0rc3/getting_started/installation.html.

Once it is installed you will need to create an account on reddit if you don't already have one. It's free and easy. Next, you need to create an authorized application on Reddit so you can make requests to the API. You can do this by going to Reddit preferences and click the "app" tab, or more easily just google "Reddit create an app" and click on the create app bottom at the bottom of the screen.

- Click on "Are you a developer? create an app..."
- You can leave "description" and "about url" blank, and put any url in redirect uri (`https://google.com` for example). Choose the "script" option for now.

You now see that there is a code underneath the words "personal use script". This is your public key. The other code that is important to note is your secret key. **Keep it secret. Keep it safe** .

You will need both keys in order to access the API and make requests.

NOTE it appears like the key expires after a certain amount of time so be aware you may need to renew it or make a new one.

Accessing the API

Now we get to start do some real data extraction from PRAW. I recommend saving your keys in a .config file and reading that into a Python file for privacy, especially if you are collaborating with other developers. You can access the Reddit API as follows:

```
    reddit = praw.Reddit(client_id=client_id,
                         client_secret=client_secret,
                         password=password,
                         user_agent=user_agent,
                         username=username)
```

NOTE: "password" is your reddit user account password.

This creates an object instance which you can now use to access any subreddit you want. Let's try accessing /r/CryptoCurrency!

```
subreddit = reddit.subreddit('CryptoCurrency')
hot_list = subreddit.hot(limit=num_posts)
```

The hot method allows you to access a specified number of posts from the subreddits "Hot" posts list. Now we need to be able to parse these posts and extract their raw text.

```
def get_raw_comments(hot):
    #Return a list of lists representing the hot list
    #with each post being its own list of comments

    #List of lists containing lists with comment text
    hot_list = []
    index = 0
    for submission in hot:
        print('submission', submission, 'index:', index)
        if not submission.stickied:
            try:

                #print("Title: {}, ups: {}, downs: {},
".format(submission.title,
                #                                               submission.ups,
                #
submission.downs))

                submission_comments = []
                submission.comments.replace_more(limit=0)
                for comment in submission.comments.list():
                    try:
                        now = datetime.utcnow().timestamp()
                        commentTime = comment.created_utc

                        delta = now - commentTime
                        if delta <= TIMELIMIT:
                            submission_comments.append(comment.body)

                    except UnicodeEncodeError:
                        pass

                hot_list.append(submission_comments)
            except UnicodeEncodeError:
                pass
        index += 1
    print('submission comments parsed, returning...')
    return hot_list
```

You will most likely want to add a condition to check if the post is stickied or not as stickied posts will complicate things. This can be done the `stickied` method for each post object in the hot list. Then each comment's raw text can be obtained with the `comment.body` method and appended onto a list.

Voila! Easy as that! Now for the fun part....

Filtering Comments

Before we can do any ML with this data we need to filter the comments before feeding them into the VADER-Sentiment Tool.

1. Many words used in every day speech are not useful for algorithms and may lead to detrimental results. These are called stop words and we need to remove them.
2. We need to standardize all the letter casing.
3. We need to remove emojis and non-alpha-numeric characters
4. We need to tokenize words in each comment, i.e. parse them into a list where each element being a word.

Lucky for us Python has many easy to use NLP tools to accomplish this task. First let us import a few modules

```
from nltk import word_tokenize
from nltk.corpus import stopwords
import re #Regex for those not familiar
```

You can write a simple function like the one below to filter each comment.

```
def filter_comments(txt):
    """
    This function will preform basic filtering on a list
    of lists containing txt and return filtered txt.
    """
    #Remove stop words
    filt_comments = []
    for comment in txt:
        words = word_tokenize(comment)
        stop_words = set(stopwords.words("english"))

        #filter out stop words
        filt_words = [w for w in words if w not in stop_words]

        #Remove emojis
        no_emoji = " ".join(c for c in filt_words if c <= '\uFFFF')

        #Remove non alpha num
        S = re.sub(r'[^A-Za-z0-9 ]+', '', no_emoji)

        filt_comments.append(" ".join(S.split()))
    return filt_comments
```

Easy right?!

Analyzing Sentiment

Now that we have all our comments filtered it's finally time to start doing some ML. First, import the VADER-Sentiment Tool

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
```

Now there are many ways that one could quantify sentiment using this model. Below is a simply approach.

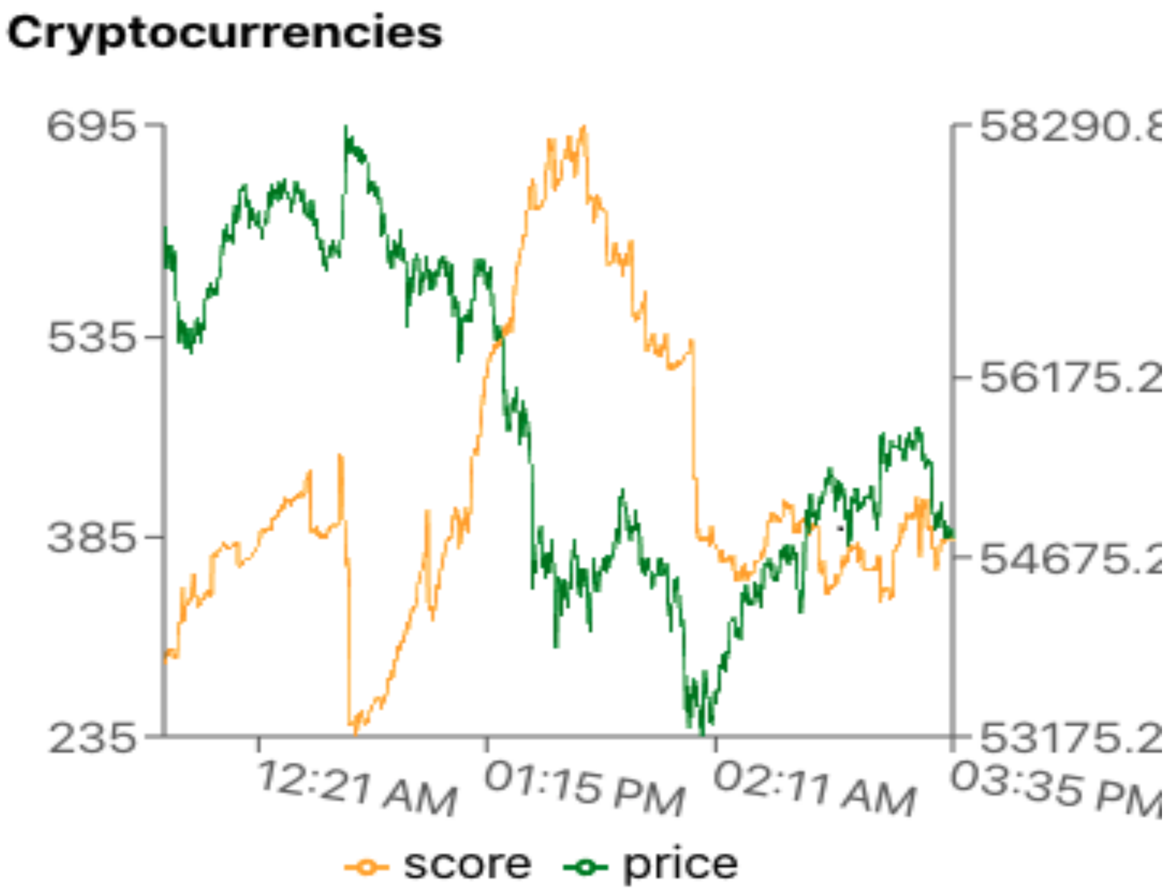
```
   for idx, post in enumerate(filtered_txt):
        for comment in post:
            sentiment = analyzer.polarity_scores(comment)['compound']
            score = 0
            if sentiment < -0.05:
                score = -1
            elif -0.05 <= sentiment <= 0.05:
                score = 0
            else:
                score = 1

            txt = word_tokenize(comment)

            for coin in COINS:
                for word in txt:
                    if word.lower() == coin['name'] or word.lower() ==
 coin['ticker']:
                        if tally[coin['name']] == -1:
                            tally[coin['name']] = 1
                            scoreKeeper[coin['name']] = score
                        else:
                            tally[coin['name']] += 1
                            scoreKeeper[coin['name']] += score
        print('post', idx, 'finished')
```

As you can see, each comment is scored by the VADER-sentiment tool and an appropriate value is added to a score keeper dictionary. A negative comment will add a -1 score, a positive will add a +1 score, and a neutral comment will add 0.

Putting it all together

If you're still following along... congratulations. You now have a quantified measure of sentiment for Bitcoin on /r/CryptoCurrency. We can repeat this process to scrape data at regular intervals to create a timeseries. Not too hard huh?

It is my hope that this article was useful to our readers and that it provides value to data scientists, cryptocurrency traders, and hobbyists alike.



References:

Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.