Natural Language Processing tool to analyse social media sentiment on UBS announcements - Karl Merisalu

Background: There is a saying that a reputation can make of break a company. As such, these days, official communication from large corporations, including from UBS, is being thoroughly monitored.

There are many ways to say the same thing, so choosing words wisely to get the message across as intended, can make a significant difference for the public opinion and also for UBS' reputation.

Problem: Even if UBS always managed to get our messages across accurately and as intended, how would it know how the announcement was percieved by the public?

Being able to measure and predict social sentiment of UBS announcements would enable UBS to answer to the following sample questions:
1) Was the announcement/event perceived well or not so well?
2) Should UBS publish an announcement now or withold instead?
3) Given the need of cost cutting, should UBS consider: furlough, sabbatical or pay cut programs for it's employees from public opinion perspective?
4) How are the quarterly results perceived by the general public?
5) Has a large Community Affairs event improved UBS image in social media?
6) How can we immediately identify topics which threaten UBS from reputational perspective?

Task: In order to be able to shed more light on the questions mentioned above I will build a text analysis tool, which I will use to identify social media sentiment levels by analysing tweets from Twitter in real time.

I use data from Twitter, because it is one of the best places to get early information on events of interest. Due to the vast number of tweets, Twitter is hard to follow manually, so in this project we will create a live (real time) indicator of Twitter sentiment score, which will help us to determine automatically if there have been any sudden changes (be it very positive or negative) to the keyword "UBS".

In my role at UBS I work closely with our Comminications department. One of my tasks as a Business Manager is to act as a bridge between Communications department and all business functions to coordinate inputs for UBS quarterly message for our region. I then review these inputs and combine them into a complete "story".

The abovementioned tool will enable me to get immediate high level feedback on how the UBS' messages, I have contributed to, have been percieved after publishing.

Additionally, I could also offer the tool to my colleagues at Community Affairs department, who might find it beneficial in assessing public feedback against charitable spend for example, and then as a result possibly adjust their spending accordingly.

I will be using Python ecosystem for this task, because it is good in dealing with large quantities of unstructured data and given good results, the model would be easy to replicate by other departments such as by Community Affairs department, Strategy, Research and etc for an additional benefit.

Limitations: There are many factors that affect social sentiment score and there may also be malicious players involved in Twitter, who may want to disseminate false information to skew result in their favour. In addition to Twitter there are also newspapers and other publications that could be analysed for a more complete picture.

This is a big task to take on, but in this case I'm building a high level proxy tool to start with, which, if used over time and to measure the impact of larger announcements/events should give me a rough indication of change in real time.

Because I will be using Twitter's free developer tool, my hands are relatively tied compared to what could be done with full access. For example, the data available is limited and made available with a delay, however, I can still make relevant adjustments and build a proof of case project.

Given the success of the proof of case project, funding could be requested from UBS, to build a fully customisable research tool.

Summary of actions: To summarize, this is a Natural Language Processing / Text Analysis project in Python, where I'm going to measure the current social sentiment score of keyword "UBS" in Twitter. This project will be useful for my firm and myself, because it will help me to measure the impact of the company announcements I review and hopefully also help my firm to make better decisions in the future by including public sentiment factor into decision making. Finally, I will conclude by summarising results and making suggestions on how to improve the tool.

Following is a step by step guide on how I'm going to accomplish it:
1) Setting up the environment *(enables me to use pre-developed complex functions on the data)*
2) Defining function to clean the text *(cleaning, standardising and giving structure to natural language text)*
3) Using Twitter API
4) Building the live chart
5) Model validation *(comparing the model against an already existing similar application socialsentiment.net)*
5) Conclusion *(including suggestions for further research)*

1) Setting up the environment

I will create a live dashboard based on a loop function and I want the loop function to be efficient (so that it could be quick). In this first step I will set up the environment as much as possible to minimise the tasks to be executed by the loop.

I start with importing required libraries:

In [1]:

```
# For data and matrix manipulation
import pandas as pd
import numpy as np

# For visualisation
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import pylab as pl
from IPython import display
import time
import datetime

# For string manipulation
import re

# For text pre-processing
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer

# For assigning sentiment polarity scores
from textblob import TextBlob

# Necessary dependencies from NLTK
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\ulome\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ulome\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\ulome\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[1]:

```
True
```

2) Defining function to clean the text

Next I define a function which I will use to clean the messy ("natural language") tweets I am going to pull from Twitter:

In [2]:

```
# defining a function/method for data cleaning
def clean_text(sample_text):

    # Given a sample text (as a string), we first substitute a select few sybmols with
 white space
    sample_text = re.sub(r'[#|@|-|?|!]',r' ',sample_text)

    # We then strip extra white space
    sample_text = re.sub('\s+',' ', sample_text)

    # Then change everything to lower case
    sample_text = sample_text.lower()

    # Then lemmatize our words -- note,  stemming was deemed too crude here, and theref
ore not chosen
    sample_text = WordNetLemmatizer().lemmatize(sample_text)

    # Now that we transformed our text, we need to tokenize it. Let's treat each word a
s a token.
    words = word_tokenize(sample_text)

    # As we now have a list of words,  we can go ahead and find and remove those words
 that also belong to the
    # stopwords list from the NLTK corpus
    words = [w for w in words if w not in stopwords.words('english')]

    # We then proceed to joining those list of words, back to 'free text'  or string fo
rmat
    text = ' '.join(words)

    return text
```

3) Using Twitter API

Because I want to build a live dashboard, I need access to live data.

UBS has 4 data confidentiality classes:
- Strictly confidential – not to be further distributed
- Confidential – only for targeted audience
- Internal – can be shared with colleagues internally
- Public – can be shared internally and externally

This Twitter data qualifies as public data, so there is no need to hide/transform any of the data

Getting live data can be arranged with an API (application programming interface). I'm going to use Twitter's Standard Developer API, which is somewhat restricted (allows to pull 100 tweets at a time, 1h delay in tweets and a restricted number of pull requests allowed per 15min), but is suitable for this purpose to showcase the live chart.

In order to use Twitter API I need to register an account on their developer site and get credentials that we use to verify our API get requests. If you're new to this, one good place to start is here: https://stackabuse.com/accessing-the-twitter-api-with-python/ (https://stackabuse.com/accessing-the-twitter-api-with-python/)

As per the instructions above, I have chosen to use Twython python library to use Twitter API. Below I import the library and open Twitter credentials from the json file that I have created earlier.

In [3]:

```
# Import the Twython class
from twython import Twython
import json

# Load credentials from json file - these credentials in twitter_credentials.json were
 saved previously to hide them from
# showing in public
with open("twitter_credentials.json", "r") as file:
    creds = json.load(file)

# Instantiate an object
python_tweets = Twython(creds['CONSUMER_KEY'], creds['CONSUMER_SECRET'])
```

Next, I need to define what keywords ('q') I'm looking to monitor in Twitter. In this case I will look to pull tweets referencing: 'UBS'. Unfortunately, Twitter allows me to pull only 100 tweets at the time, so I will go for the maximum of 100. I will request the most recent 100 tweets referencing 'UBS'.

In [4]:

```
# Create our query
query = {'q': 'UBS', # insert search term here
        'count': 100, # pulling the maximum of 100 tweets
        'result_type': 'recent', # pulling most recent tweets (also possible to pull by
popularity, etc.)
        'lang': 'en', # pulling tweets in english language
        }

# Creating an empty dataframe where we will be saving (appending) results that will be
 dispalyed on the live chart
dfrolling = pd.DataFrame()
```

4) Building the live chart

1) First I use Twython API method to pull tweets as already explained above (and here: https://stackabuse.com/accessing-the-twitter-api-with-python/ (https://stackabuse.com/accessing-the-twitter-api-with-python/))

2) In order to make it "live" to continuously update the status with most recent sentiment, I will build a loop around the API request.

3) Inside the loop, once I have the tweets: I clean the text, then process for sentiment value and finally plot the value on a chart.

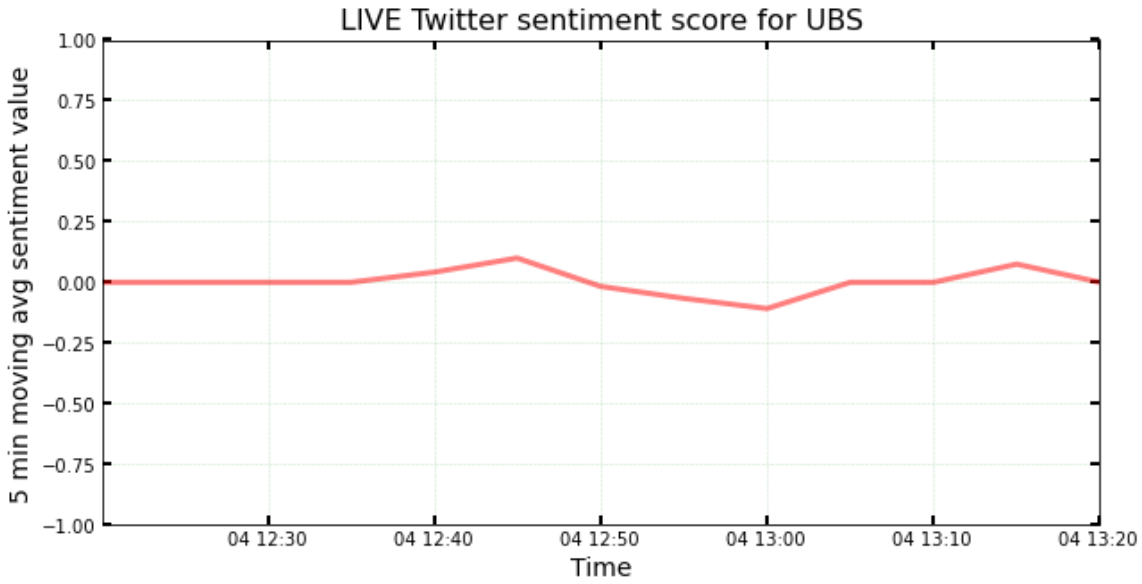
Because (almost) everything below will be in a loop and needs to work continuously, it will all need to be in one jupyter notebook cell. **Hence, further explanations are provided in the code:**

In [5]:

```
# 1st search for tweets (as per Twython guidelines)
dict_ = {'user': [], 'date': [], 'text': [], 'favorite_count': []}
for status in python_tweets.search(**query)['statuses']:
    dict_['user'].append(status['user']['screen_name'])
    dict_['date'].append(status['created_at'])
    dict_['text'].append(status['text'])
    dict_['favorite_count'].append(status['favorite_count'])

# Structuring data in a pandas DataFrame for easier manipulation - the purpose of this
 is to create a dataframe
# that I can start "appending" with the loop below.
df = pd.DataFrame(dict_)

# Start of the loop: inserting range value to how many times I want to run the loop (1
3)
for i in range(13):
    dict_ = {'user': [], 'date': [], 'text': [], 'favorite_count': []}
    for status in python_tweets.search(**query)['statuses']:
        dict_['user'].append(status['user']['screen_name'])
        dict_['date'].append(status['created_at'])
        dict_['text'].append(status['text'])
        dict_['favorite_count'].append(status['favorite_count'])

    # Structuring data in a pandas DataFrame for easier manipulation as above (now just
in the loop). Saving newly
    # pulled tweets into 'df2'
    df2 = pd.DataFrame(dict_)

    # Adding df2 results to the dataframe 'df' which I created before the loop note tha
t 'df2' will be overwritten
    # with new tweets with every loop, but 'df' will contain all tweets/ act as a datab
ase
    df = pd.concat([df,df2], ignore_index=True)
    df.sort_values(by='text', inplace=True, ascending=False)

    # The algorithm will pull 100 most recent tweets, but if I don't have 100 new tweet
s in every cycle then
    # duplicates will occur to fill the void, so will need to remove duplicates
    df.drop_duplicates(subset = 'text', inplace = True)

    # In order not to modify the master database 'df' we add only necessary information
(time and tweet text columns)
    # from 'df' into a new dataframe 'tweets'
    tweets = df[['text','date']]

    # Formatting tweets 'date' column into datetime and setting 'date' column as index
 for time series chart later on
    # '%a %b %d %H:%M:%S +0000 %Y' <-- encoding Twitter time into Python datetime under
standable date format
    tweets['date'] = pd.to_datetime(tweets['date'], format = '%a %b %d %H:%M:%S +0000 %
Y')
    tweets.set_index('date', inplace = True)

    # Sorting tweets by time
    tweets = tweets.sort_index(ascending=0)

    # Speficication: I want to display on the chart most recent 5min moving average sen
timent value of tweets with
    # our keyword. I also want the status to refresh every 300 seconds
```

```
    # Since standard developer (free) twitter API access has a 60min delay built in, I
 won't be able to get real live tweets.
    # I will therefore specify that I want to analyse only tweets from now until 65min
 in the past (1h delay + 5min). If
    # you have access to real live tweets, change it to 5min instead of 1h + 5min
    movavg = tweets[datetime.datetime.now():(datetime.datetime.now() - datetime.timedel
ta(minutes=65))]

    # I will apply the text cleaning function that we defined above to the most recent
 5min of tweets
    movavg = movavg['text'].apply(lambda x: clean_text(x))

    # Creating a new array where we save sentiment values. it's also important to creat
e this/clear it with
    # every loop because I only want the latest data, and not to add new data to old da
ta.
    sentiments = []

    # I use TextBlob (which includes a dictionary with polarity scores for each word, s
ome phrases among other things) to get
    # sentiment values for each (cleaned) tweet
    for tweet in movavg:
        analysis = TextBlob(tweet)
        sentiments.append(analysis.sentiment.polarity)

    # Converting movavg from time series to a dataframe
    movavg = pd.DataFrame(movavg)

    # Adding sentiment values to moving average tweets (movavg)
    movavg['sentiments'] = sentiments

    # Creating a new dataframe where I add current time and the mean of latest tweet se
ntiments. This one will be
    # displayed on the chart and I want it to update (append) with new data while also
 maintaining old data
    dfrolling = dfrolling.append({'Time': datetime.datetime.now(), 'movavg': movavg["se
ntiments"].mean()}, ignore_index=True)

    # Plotting the live chart: I choose the chart to show the most recent 60 minutes of
sentiment values. Since I decided to
    # refresh at 300sec and I run the loop for 13 times, I should be just about to be a
ble to fill the full chart in 1h
    axes = plt.gca()
    axes.set_xlim([(datetime.datetime.now() - datetime.timedelta(minutes=60)),datetime.
datetime.now()])

    # Setting the sentiment limits (y-axis) between -0.4 and 0.4 to be able to see chan
ges better
    axes.set_ylim(-1,1)

    # Feeding data for plotting and naming x and y labels
    pl.plot(dfrolling['Time'], dfrolling['movavg'], linewidth=3.0, alpha=0.5, color="re
d")
    pl.xlabel('Time', size=14)
    pl.ylabel('5 min moving avg sentiment value', size=14)
    pl.gcf().set_size_inches(10,5)
    plt.grid(linestyle='--', linewidth=0.5, alpha=0.2)
    plt.title('LIVE Twitter sentiment score for UBS', fontsize=16)
    plt.tick_params(axis='both', bottom=True, top=True, left=True, right=True, directio
n='in', which='major', grid_color='green', length=5, width=2, color="black")
```

```
    # clearing the old output so that I see only newest data and only 1 chart.
    display.clear_output(wait=True)
    display.display(pl.gcf())
    pl.gcf().clear()

    # Insert here value in seconds (300), how often I want the algorithm pull new data.
Useful to give some time to allow
    # for new tweets to be generated
    time.sleep(300)

    # End of the loop

# Once the loop has run its predefined number of times, I finally save the full databas
e 'df' on a disk
df.to_csv('movavg.csv', encoding='utf-8', index=False)
```




```
<Figure size 720x360 with 0 Axes>
```

6) Model validation

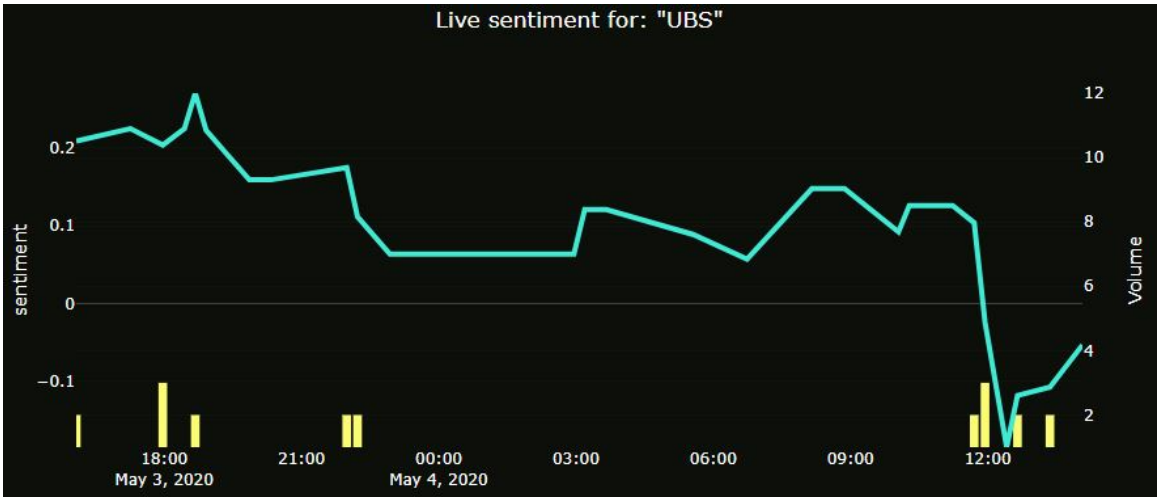
This isn't the first time someone comes up with an idea to source data from Twitter. Analogous models have been built in the past and one similar example can be found at socialsentiment.net. In order to validate my model, I will compare my results to the below results from socialsentiment.net.

From app.brand24.com I can also take the volume of tweets containing UBS.

In [8]:

```
from IPython.display import Image
Image(filename='UBS sentiment.jpg')
```

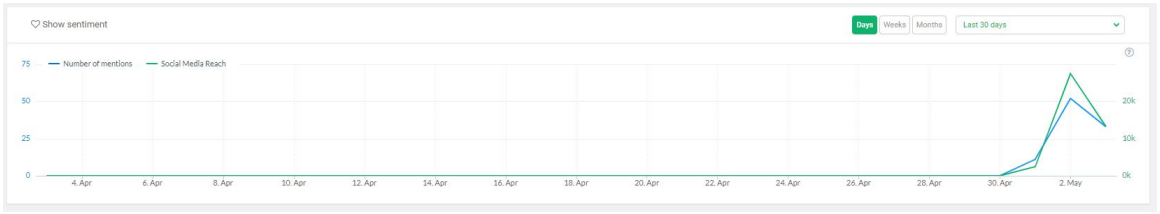
Out[8]:



In [9]:

```
from IPython.display import Image
Image(filename='UBS twitter volume.jpg')
```

Out[9]:



I see some similarity in the trend of my chart and socialnetwork.net, but as expected, they're not exactly the same. Sentiment value is hovering around 0 for both tools and it is in this case likely caused by the lack of tweets containing UBS in combination with tweets with neutral sentiment score. This is not surprising for a Monday noontime in London. This doens't mean, however, that one of the tools is necessarily wrong/invalid though. I would just need to make sure I use my tool with care for specific purposes.

From volume perspective, the more tweets there are the more likely it is that the sentiment can be used as a proxy for general public attitudes. In this case I can notice a peak of tweets following UBS quarterly results announcement on 28th April. Nevertheless, the number of tweets overall is extremely low compared to keywords like "bitcoin" or "Boris Johnson" for example.

What are the possible reasons these scores on 2 tools are different:
1) My chart calculates sentiment value for 15min period, socialnetwork.net seems to be doing it much more frequently
2) Different algorithm. I'm using a simple textblob library to calculate average sentiment value of tweets. socialnetwork.net may be using a more or less advanced sentiment score calculation method
3) Based on sample tweets shown on the bottom of socialnetwork.net chart and comparing these to the tweets used in my tool (movavg.csv), I can tell that both tools include occasional erroneous tweets like website links which include consecutive letters "ubs" and as such should only be used as rather high level sentiment indicators
4) My tool pulls tweets only in English language, socialnetwork.net includes and analyses also other languages

5) Conclusion

Summary findings: Based on the chart above, I can tell that tweets containing keyword: 'UBS' have been rather neutral in the past 1 hour. There are no big swings in the mean sentiment analysis, however, it would be interesting to put it into perspective over a longer (or shorter) time period at the time of big announcements. Generally speaking, the number of tweets is also lower in the absence of significant events.

Due to Twitter restrictions, unfortunately I could only live plot tweets with 1h delay, however, should you have access real live Twitter feed, then this tool could be used for various situations for indicating changes in sentiment - for example helping to spot areas of concern for UBS management and prepare the management for a response quickly.

Based on this 1h tweet data analysis, there doesn't seem to have been any significant announcements (or unexpected problems) that would show up on my chart.

Real life implications: Despite no significant discoveries in the past 1h of my analysis, to get started, I would recommend UBS Communication department to do the following:
1) Test the tool when in the case of upcoming significant announcements/events. The results of the tool could give additional insight into how the announcement it percieved by the public and also crafting messages for future announcements (if time sensitive, may require upgrade of twitter developer account)
2) Use the tool to track the sentiment of our competitors and compare how they are doing against UBS (could be done with existing tool, no need for paid full version Twitter developer account)
3) Have the tool running on the background indefinitely and send an email alert (incl. sample tweets) when breaching certain sentiment threshold
4) If the tool proves to be valuable, invest in further development of the tool (see below)

Further research/improvements:
1) Accuracy: These results are based on TextBlob pre-defined dictionary, which doesn't take into account more complicated sentences/meanings and as such should be taken with some suspicion. To develop the algorithm further it may be interesting to go into deep learning field as outlined here: https://blog.insightdatascience.com/how-to-solve-90-of-nlp-problems-a-step-by-step-guide-fda605278e4e (https://blog.insightdatascience.com/how-to-solve-90-of-nlp-problems-a-step-by-step-guide-fda605278e4e).

2) Specifications: Based on the number of tweets and the need of timely information, I would recommend to play around with refreshing frequency (sleep value) and with moving average specifications. For identifying macro trends a larger moving average values (and slower refresh rate) could be useful, for micro trends smaller moving average values and faster refresh values could be handy.

3) Further development:
- It would be interesting for example if the application sent an email alert if the sentiment value breached some designated value
- An additional idea for further development would be to make user experience simpler, by having the application ask user input for keyword and then run the chart
- I could have several moving averages (5sec, 5min, 1h for example) plotted on 1 live chart, which would help puting sentiment values into perspective
- The tool could be used for comparing sentiment values for different keywords at the same time, for example: furlough vs sabbatical vs pay cut