Sentiment Analysis based stock market recommendations

Rajeev Aravindakshan Nair

Capstone Project (WQU, MSFE)
(rajeevaravindnair@hotmail.com)

Abstract

Sentiment analysis on a large corpus of market/analysts' sentiments, on a day-to-day basis, and use the result to recommend buy/sell/hold actions on various stock market tickrs. Though sentiment analysis can be deployed on a wide variety of text-based data that's available on the internet, such as – Google Search results, Tweets, etc…, the MVP for this project concentrates primarily on scraping news and commentaries on the open web and leverage on the wisdom of the crowd to make investment decisions.

1. Introduction

The attempt of this project is to leverage the wisdom of the 'informed' crowd and build trading recommendations based on that. There is a lot of first party research that happens and a lot of them are available for free via market commentaries and analysts' blogs. An NLP based model that can consolidate these sentiments and commentaries and thereby classify the 'mood' of market to various degrees that ranges from negative to neutral to positive, can then be executed using automated trading algorithms.

3. Main Title

Sentiment Analysis based stock market recommendations

4. Author Name(s) and Affiliation(s)

Rajeev Aravindakshan Nair

4.1. Affiliations

Student MScFE at WorldQuant University

7. Steps involved in building a MVP

Step 1: Identify top stock market analyses website in the geography of interest. (Example Finviz.com)

Step 2: Write a program that scrapes the website comprehensively.

Tools to be used: beautifulsoup & python.

Step 3: In the scraped corpus identify key tickrs/stocks mentioned and bucket them along with the related bag of words.

Method: Fuzzy match with a tickr corpus/list.

Step 4: Refine the bag of words (by removing stop words etc..) and prepare it for sentiment analysis.

Step 5: Use a suitable NLP framework to perform sentiment analysis in the refined bag of words.

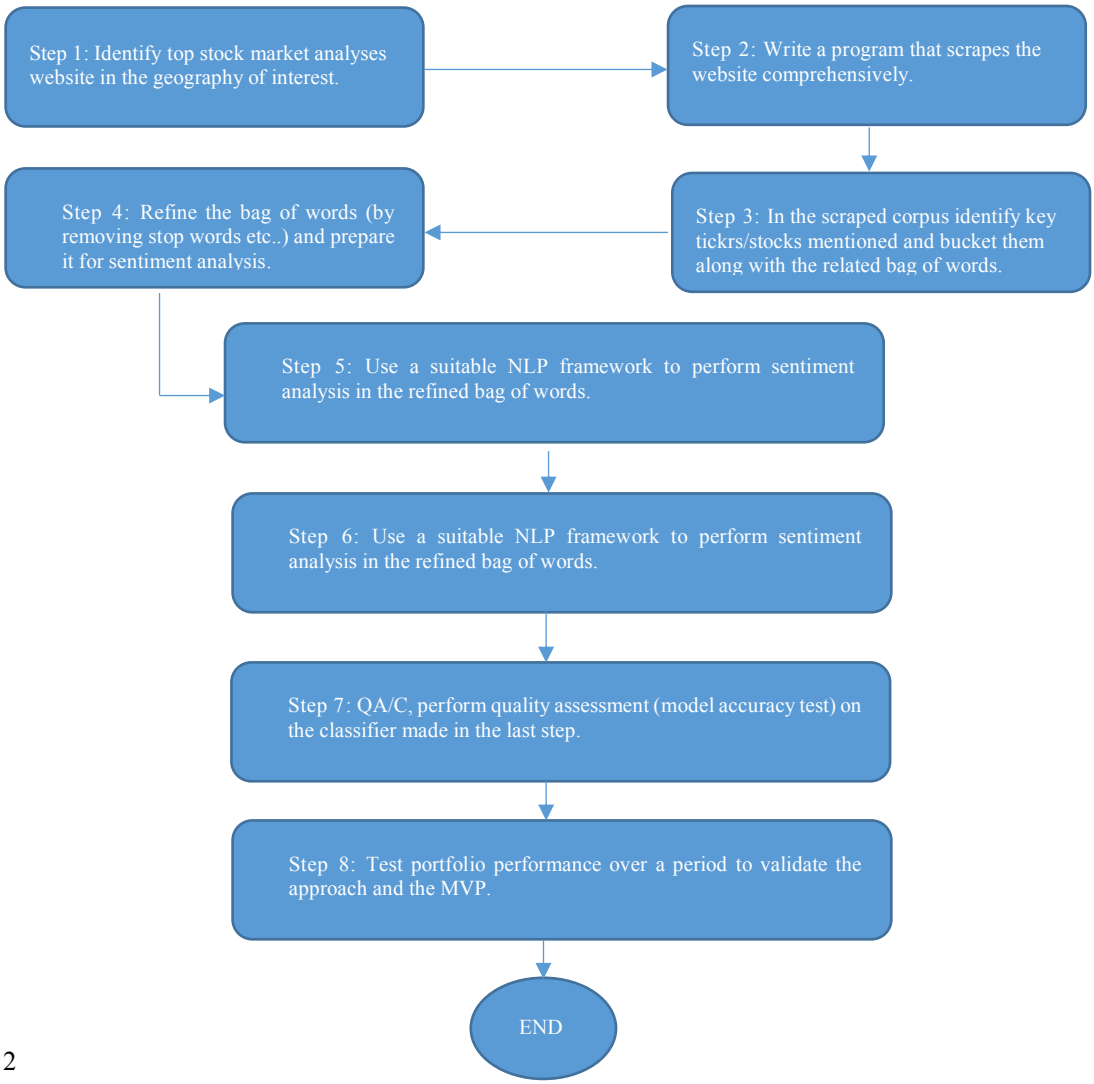
Tools to be used: scipy , nltk

Step 6: Recommend Buy/Sell (weighed by the confidence of the calculation) on various stock tickrs.

Step 7: QA/C, perform quality assessment (model accuracy test) on the classifier made in the last step.

Step 8: Test portfolio performance over a period to validate the approach and the MVP.

FLOWCHART



8. Minimum Viable Product MVP

Refer the python notebook attached

Assumptions:

1) A minimum viable product is made to analyze only the commentaries around FAANG stocks (Facebook, Apple, Amazon, Netflix and Google)
2) Commentaries from news headlines as reported in Finviz.com is scraped to form the insights.

Steps:

1) Import the required packages

Import all the relevant packages

In [121]:
```
#Basic modules
import time
import requests
from bs4 import BeautifulSoup
import os
import pandas as pd
import progressbar

#Yahoo finance modules
from pandas_datareader import data as pdr
import yfinance as yf
yf.pdr_override()

#NLP Modules
import nltk
import twython
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.sentiment.vader import SentimentIntensityAnalyzer

nltk.download('stopwords')
nltk.download('punkt')
nltk.download('vader_lexicon')

sia = SentimentIntensityAnalyzer()
```

2) Schedule a daily scraping exercise for the stocks of interest. For the MVP, only FAANG stocks are considered (FAANG: Facebook , Apple, Amazon, Netflix & Google)

For MVP purpose we limit our analyses to the FAANG stocks.

We use tickr news section from finviz.com to get the relevant commentaries about the stock and market.

In [122]:
```
tickr = ['fb','aapl','amzn','nflx','googl']

p = progressbar.ProgressBar()

p.start()

fb = requests.get("https://finviz.com/quote.ashx?t=FB")
aapl = requests.get("https://finviz.com/quote.ashx?t=AAPL")
amzn = requests.get("https://finviz.com/quote.ashx?t=AMZN")
nflx = requests.get("https://finviz.com/quote.ashx?t=NFLX")
googl = requests.get("https://finviz.com/quote.ashx?t=GOOGL")

fb_soup = BeautifulSoup(fb.content, 'html.parser')
aapl_soup = BeautifulSoup(aapl.content, 'html.parser')
amzn_soup = BeautifulSoup(amzn.content, 'html.parser')
nflx_soup = BeautifulSoup(nflx.content, 'html.parser')
googl_soup = BeautifulSoup(googl.content, 'html.parser')

p.finish()
```

```
100% |########################################################################|
```

Scrape the commentaries seciton from finviz.com

In [123]:
```
#FB

table = fb_soup.find_all(lambda tag: tag.name=='table')
rows = table[31].findAll(lambda tag: tag.name=='tr')
FB=[]
for i in range(len(rows)):
    td=rows[i].find_all('td')
    FB=FB+[x.text for x in td]

#AAPL

table = aapl_soup.find_all(lambda tag: tag.name=='table')
rows = table[31].findAll(lambda tag: tag.name=='tr')
AAPL=[]
for i in range(len(rows)):
    td=rows[i].find_all('td')
    AAPL=AAPL+[x.text for x in td]

#AMZN

table = amzn_soup.find_all(lambda tag: tag.name=='table')
rows = table[31].findAll(lambda tag: tag.name=='tr')
AMZN=[]
for i in range(len(rows)):
    td=rows[i].find_all('td')
    AMZN=AMZN+[x.text for x in td]

#NFLX

table = nflx_soup.find_all(lambda tag: tag.name=='table')
rows = table[31].findAll(lambda tag: tag.name=='tr')
NFLX=[]
for i in range(len(rows)):
    td=rows[i].find_all('td')
    NFLX=NFLX+[x.text for x in td]

#GOOGL

table = googl_soup.find_all(lambda tag: tag.name=='table')
rows = table[31].findAll(lambda tag: tag.name=='tr')
GOOGL=[]
for i in range(len(rows)):
    td=rows[i].find_all('td')
    GOOGL=GOOGL+[x.text for x in td]
```

3) To quantify sentiments, I have defined a metric called Positivity_Index

Sentiment Analysis

Positivity index is defined as the ratio of positive comments & negative comments multiplied by 100.

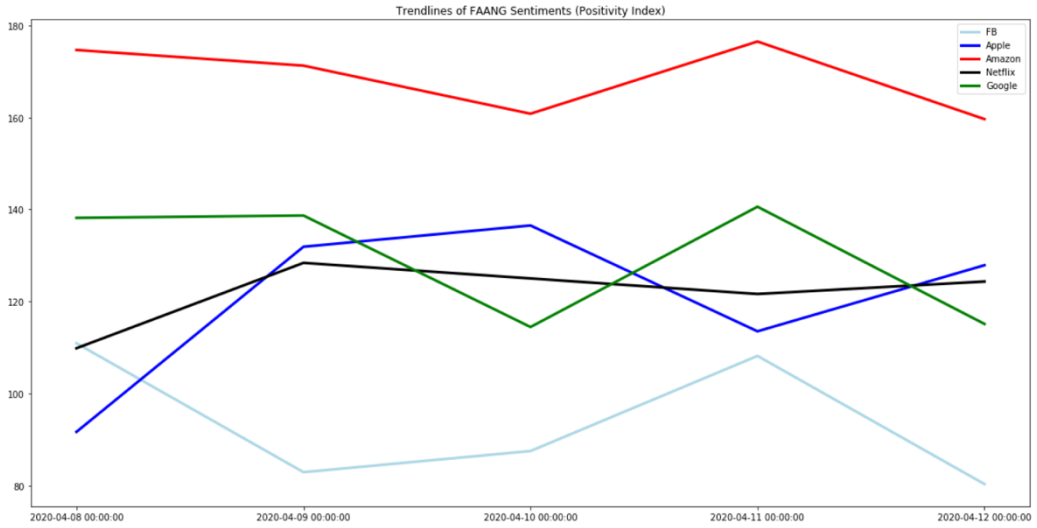
An index of 100 refers to neutral sentiments. An index below 100 represents negative sentiments and an index above 100 refers to positive sentiments

In [125]:
```
df['positivity_index']= ''
df['buy/sell'] =''

for i in range(0,len(df)):
    text = str(df.comments[i])
    text_tokens = word_tokenize(text)
    tokens_without_sw = [word for word in text_tokens if not word in stopwords.words()]
    negative = sia.polarity_scores(text)['neg']
    positive = sia.polarity_scores(text)['pos']

    Positivity_Index = positive*100/negative

    print("Positivity Index of " + tickr[i] + " is : " ,Positivity_Index)
    if Positivity_Index > 100:
        Signal = 'BUY'
    else:
        Signal = 'SELL'

    df['positivity_index'].loc[i] = Positivity_Index
    df['buy/sell'].loc[i] = Signal
```

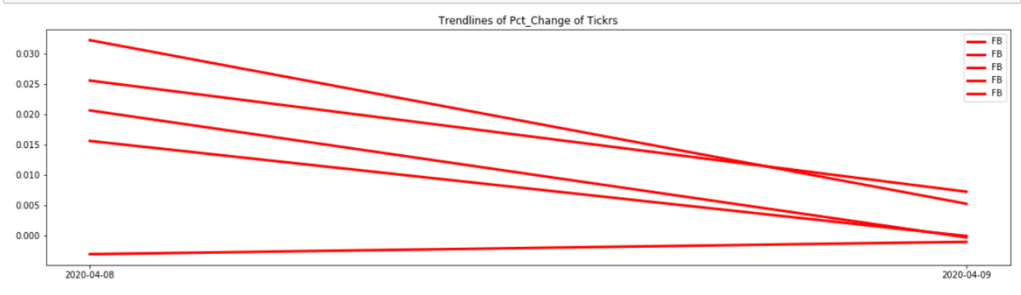
```
Positivity Index of fb is :  82.90598290598291
Positivity Index of aapl is :  133.33333333333331
Positivity Index of amzn is :  174.64788732394368
Positivity Index of nflx is :  130.13698630136986
Positivity Index of googl is :  138.66666666666669
```

Save sentiment data for the day

4) Plot of sentiments observed over the days analyzed:



5) Plot of Closing price changes over the two live days for which data was analyzed



9. Conclusion

```
In [231]: Corr_Matrix
Out[231]:
```

	tickr	correlation coefficient
0	FB	1.0
1	AAPL	-1.0
2	AMZN	1.0
3	NFLX	1.0
4	GOOGL	-1.0

9. Result

Disclaimer: *Due to limitations in doing backdated scraping. Analysis was limited to 4 day's data scraped from FINVIZ.com*

Next steps in the roadmap

Step 1 : Identify top blogs and opinion columns in the geography of interest (beside the news commentaries) and scrape the data that's relevant to top tickrs.

Step 2: Currently positivity_index is a relative score. This has to be quantified objectively. As in our example above, the range of positivity_index is from 88.6 to 134.9 . Besides calling out buy/sell actions, we shold be able to assign weights to the buy/sell trades. To arrive at these weights needs extensive back testing to fine tune the weights.