Reddit Sentiment Analysis

S. Lindskog[*] and J.A. Serur[†]

August 2020

Summary

It has never been easier for individual investors to get started trading stocks or options. Companies like Robinhood and Webull even offer zero commission trades, no account minimum size, and incentives like a free stock if a user creates an account. Recently, there has been a huge increase in the growth of users of these types of platforms. The combination of many people out of work because of the coronavirus and the US government stimulus package appears to have sparked this. This surge in new investors has sparked tons of activity on popular social media websites like Reddit. There users regularly post stock recommendations and trading strategies. It appears that many Reddit traders are grouping together and causing irrational stock market moves. Panic buying stocks for companies that just declared bankruptcy, betting against Warren Buffet, and ignoring the impact of the coronavirus on airlines and cruise ships are a few of the unusual market moves lately. The purpose of this project is to identify if there is a relationship between the Reddit sentiment on stocks and performance.

[*] New York University (NYU) - Courant Institute of Mathematical Sciences. Email: sebastian.lindskog@nyu.edu.

[†] New York University (NYU) - Courant Institute of Mathematical Sciences. Email: juan.serur@nyu.edu.

Contents

1 Introduction and Literature Review

For this project we plan to collect Reddit data and analyze it using big data and NLP techniques to check how much interest and what the Reddit sentiment is for different stocks. This data will then be used to verify the predictive power Reddit users have on the stock market. If Reddit users have significant foresight of the stock market a tool that shows the amount of Reddit interest in stocks over time and the performance of the stock would be useful for investors

Investors' race to capture alpha has led to major changes in the way investment decisions are made. Increasingly, funds are using systematic rule-based strategies programmed with mathematical algorithms to deliver investors' expectations. Also, because of the technological increase in recent years, the availability and possibility of analyzing large sources of databases, has sparked the interest of professionals to use alternative data, for example, from social media.

Unlike traditional information, such as time and sales data, order book data, and financial statements, information from alternative sources (from social media to satellite images) require more work regarding the extraction, cleaning, processing, storage, and finally, the transformation of the data to signals that can be used for financial purposes.

2 Data Collection

2.1 PRAW and Pushshift

The initial plan was to use the Python PRAW API in order to collect the comment data from Reddit. While PRAW it is very easy to use and has many features, it has very strict API request limits and it is not suitable for collecting large amounts of data. PRAW only allows about 100 items per API request. In order to avoid the request limits the Pushshift API was used instead. This allowed 20,000 comments to requested at a time and greatly reduced the time needed to collect data. One drawback of Pushshift is that the data is not as feature rich as PRAW. It lacks additional metrics like how controversial the comment is, the total number of up or down votes the comment receives, and it is not guaranteed to be real time accurate. This is because Pushshift is regularly collecting the Reddit data. For example, the total number of likes a comment receives may not always reflect the real time number of likes.

2.2 Stock Market Data

The Python package yfinance was used to collect the daily stock market data. This package made it incredibly simple to collect large amounts of Yahoo Finance historical OHLC data. The package is even capable of collecting intra-day data down to a 1 minute granularity, but only for the past week. In order to collect enough data in time we opted to use the daily OHLC data. One additional benefit of the yfinance package is that the stock prices are adjusted for stock splits, dividends, etc.

2.3 Multi-processing

Multi-processing allows the script to execute multiple sequences of instructions simultaneously. The biggest bottle neck for the project was downloading and processing the Reddit comments. Multi-processing was used to help speed up these steps. The data is first collected using the Pushshift API and then processed slightly. Then the data undergoes heavy processing when the CNN sentiment classifier attempts to label each comment as bullish, bearish, or neutral. The multi-processing greatly decreases the time for the data collecting and processing to run. In some cases it was about 2-3x decrease in run time.

2.4 Databases

Although a variety of databases were sampled for this project, by far the most efficient for our purposes was a local MongoDB, used for both the stock market data and the Reddit comment data. This gave us an opportunity to explore a NoSQL database. The MongoDB it is very fast and flexible. It allowed us to quickly create tables without the need to set up a schema. Lastly, we tried to set up another alternative on the cloud using an AWS DocumentDB. After more rough calculations this appeared to be reasonable. The DocumentDB is AWS's version of MongoDB and was very fast and easy to use.

2.5 Data Collecting Challenges

Texto de origen The data collection process posed several challenges. First, the PRAW package API limits were not suitable for collecting big data, the Pushshfit API had similar issues but not as strict, and lastly there were a number of database issues. In the end we were able to collect a large amount of data through Pushshift but were hit with the same API limits if you tried to use too many cores at once. To avoid this an experimental version of the data collection was tested on the IBM server using exponential waiting after each request. This worked but was still fairly slow. Another attempt was made using the Python packages asyncio and aiohttp. This seems to be the correct approach to the problem but an issue was that the list of urls to request is not static. As we request more urls, the list will grow to accommodate the 20,000 comment limit per request (unfortunately many of the wallstreetbet Posts had more than 20,000 comments).

3 Natural Language Processing

3.1 Introduction

In order to correctly classify the sentiment of the comments it was necessary to build a custom sentiment classifier. Generic pre-trained classifiers like VaderSentiment while excellent at classifying the general sentiment are not able to determine complex use cases. For instance, the Reddit comments needed to be analyzed to see if the comment was bullish, bearish, or neutral on whichever stocks the author was commenting on. The VaderSentiment package would just naively give a generic sentiment to these comments without considering if the author expected or wanted the stock to go up or down. The Python package spaCy was used to build and train a custom text sentiment classifier using Reddit data that we had manually classified.

3.2 spaCy

spaCy is an open-source library for NLP, mostly used in Python and Cython which provides pre-trained Neural Networks, specifically, Convolutional Neural Networks (CNNs) to perform NLP related taks in several languages (will provide images with examples).

spaCy utilizes a pipeline model to process raw text into Doc objects. The default pipeline will tokenize the text, tag parts of speech, parse the text, and classify named entities. This pipeline structure makes it easy to customize how the user want's the text parsed, quick to update models or disable steps in the pipeline. Below is a diagram showing the basic pipeline flow. Some of the powerful features of spaCy is NER or named entity recognition and POS or part of speech tagging. These features make it possible to create custom search patterns that would be nearly impossible to make using simple regular expressions.

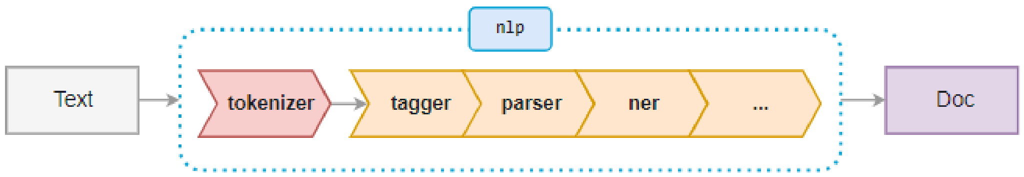

Figure 1: Pipeline structure of spaCy.

spaCy provides a variety of different ways to train models. It's possible to create statistical models for custom named entity recognition, classify text senitment, or even very advanced entity linking recognition models. Below is a diagram illustrating how spaCy trains a model to classify text.

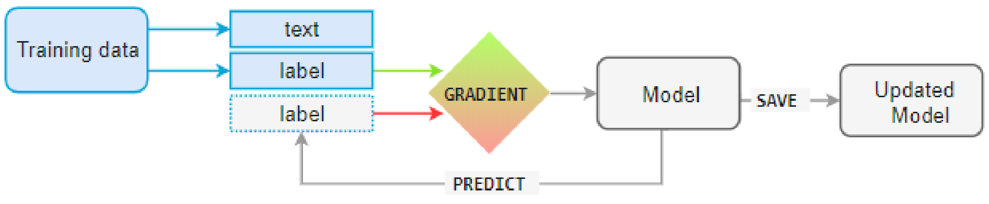

Figure 2: Training diagram of spaCy.

spaCy provides a step by step guide in order to train your own text classification model. We first had to classify about 600 Reddit comments and assign it a sentiment score for if it was a Bullish comment or if it was Not Bullish. 0 meant that the comment was not bullish and 1 meant it was bullish. Then we could train a classifier to detect bullish comments. The same process was repeated using training data with the labels Bearish and not bearish. This resulted in two classifiers to predict bullish or bearish comments. spaCy makes it easy to save the model, load it again in the future, and even retrain the model later. Both classifiers were used on each comment and the average score was used to determine if the comment was Bullish, Bearish, or Neutral.

3.3 Naive Bayes

The Naive Bayes classifier is a generative classifier. I.e., it uses the Bayes rule to expresses how to generate the features of a text if we already know the class. Let's start by postulating the Bayes' theorem as following

$$P(A \mid B)=\frac{P(B \mid A) P(A)}{P(B)} \tag{1}$$

with $P(A)$ and $P(B)$ being the probabilities of the events A and B occurring independently. Then, we can rewrite as the conditional probabilities of observing the class C given the observations $\{O_1 \dots O_N\}$

$$P\left(C \mid O_{1}, \ldots, O_{N}\right)=\frac{P\left(O_{1}, \ldots, O_{N} \mid C\right) P\left(C\right)}{P\left(O_{1}, \ldots, O_{N}\right)} \tag{2}$$

the estimation of the term $P\left(O_{1}, \ldots, O_{N} \mid C\right)$ can be tricky and that why we use the "Naive Bayes" approach, i.e., it makes the assumption of conditional independence such that $P\left(X_{i} \mid C, O_{1}, \ldots, O_{i-1}, X_{i+1}, \ldots, O_{N}\right)=P\left(O_{i} \mid C\right)$. This allows to rewrite the above equation as

$$P\left(C \mid O_{1}, \ldots, O_{N}\right)=\frac{P\left(C\right)}{P\left(O_{1}, \ldots, O_{N}\right)} \prod_{i=1}^{N} P\left(O_{i} \mid C\right) \tag{3}$$

Then, the classification procedure is quiet simple. We have a learning vocabulary V with M words v_i. So

$$P\left(O_i \mid C_a\right) = \prod_{j=1}^{M} U_{i,j,a} \tag{4}$$

$$Q_{i,j,a} = P\left(v_j \mid C_a\right) \ \forall \quad O_{i,j} = 1 \tag{5}$$

$$Q_{i,j,a} = 1 - P\left(v_j \mid C_a\right) \ \forall \quad O_{i,j} = 0 \tag{6}$$

In the above step, we use the frequency of the words to the estimate the conditional probabilities. Then, the predicted class C_p can be is obtained as

$$C_p = \text{argmax}_{C_a} P\left(C_a\right) \prod_{i=1}^{N} \prod_{j=1}^{M} \left[P\left(v_j \mid C_a\right)\right]^{O_{i,a}} \left[1 - P\left(v_j \mid C_a\right)\right]^{1-O_{i,a}} \tag{7}$$

3.4 Logistic Regression

Opposite to the Naive Bayes approach, this is a discriminative algorithm. I.e., it simply categorizes a given signal regardless how the data was generated. We use the Logistic regression with the softmax classifier, so we have

$$z = \sum_{i}^{N} w_i O_i + b \tag{8}$$

where w_i is the weighted of observation O_i and b is the bias term. Then, the softmax function is

$$\text{softmax}(z) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \ 1 \leq i \leq k \tag{9}$$

where K are the classes. We need to express this as probabilities. So, we have that $P(y = c) = \text{softmax}(z)$.

To learn the model parameters Θ, i.e., the weights w_i and the bias b, the algorithm uses the cross-entropy loss function

$$\mathcal{L}(\hat{y}, y) = -\sum_{k=1}^{K} 1\{y = k\} \log p(y = k \mid O) \tag{10}$$

$$= -\sum_{k=1}^{K} 1\{y = k\} \log \frac{e^{w_k \cdot O + b_k}}{\sum_{j=1}^{K} e^{w_j \cdot x + b_j}} \tag{11}$$

Then, the gradient of the function can be written as

$$\frac{\partial \mathcal{L}}{\partial w_k} = -(1\{y = k\} - p(y = k \mid O))x_k \tag{12}$$

$$= -\left(1\{y = k\} - \frac{e^{w_k \cdot O + b_k}}{\sum_{j=1}^{K} e^{w_j \cdot O + b_j}}\right) O_k \tag{13}$$

3.5 Convolutional Neural Network

CNNs have been a core tool used in Image Classification and Computer Vision systems used in a wide range of applications. NLP is not an exemption, CNNs become a cornerstone in this field.

Let's start with convolution definition as

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{14}$$

$$= \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau \tag{15}$$

the convolution operation is performed by the so-called convolution layer or Kernel (aka, Feature Detector). Additional layers works within the whole process. The normalization layer (note that this is a different scheme than the commonly used batch normalization) and the maxout layer, i.e., the layer with a *max* activation function. Further, it adds a softmax layer onto the convolution layer:

$$\text{softmax}(z) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad 1 \leq i \leq k \tag{16}$$

In the figure below, the upper plot is the cumulative YTD returns for Zoom. The lower plot is comparing the aggregate sentiment score using the generic Vader sentiment classifier vs the aggregate bullish-bearish sentiment score from the custom CNN for Zoom. It shows that certain days the Vader and CNN classified the sentiment very differently.

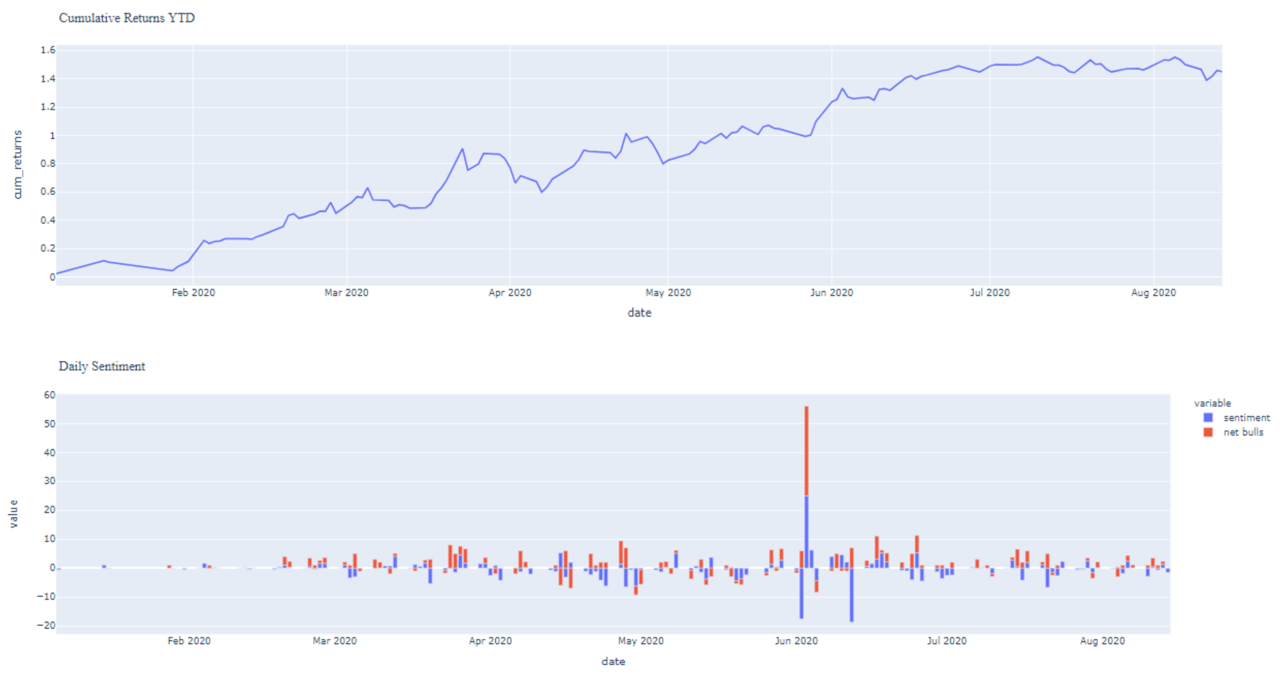

Figure 3: Zoom Vader sentiment vs CNN sentiment.

3.6 NLP Challenges

There were many issues with training the spaCy sentiment text classifier. It was pretty time consuming to classify so many comments. This task requires a great domain knowledge about trading to be able to classify properly, and yet some comments are still extremely difficult to classify. For example, some comments were either too vague or had buy and sell recommendations or were written so strangely it was not clear what the author meant. A good improvement on this project would be to use spaCy's named entity recognition feature to try to parse the comments were the author is talking about multiple stocks. spaCy seems capable of detecting which verbs are being applied to which nouns. Another issue faced was that the terminology and slang words in comments is constantly changing. This means that the model will have to be updated regularly to keep up with these new words. The company that created spaCy, ExplosionAI, offers an incredible NLP tool called Prodigy. Prodigy lets the user quickly set up a UI to manually classify training data. This tool accepts a list of words, comments, or posts and then creates a UI so the user can quickly classify them as Positive or Negative (or any custom defined categories). Then the user can use the built in command line tools to traing the model with this new training data. However, due to our budget constraints, this tool was not used in this project.

4 Quantitative Trading Strategies

This section is devoted to the experimental results. To assess whether alternative information can add “alpha” to the investment decision process, a passive strategy -buying and holding the main index- will be compared with an active strategy based on the “Sentiment Factor”.

In addition, a second active strategy will be carried out combining classic technical indicators, such as the moving average with sentiment analysis. Empirical evidence has shown that “augmenting the signal” with alternative data can improve the performance of classical trading strategies.

4.1 Momentum Strategies

Momentum is a well-know phenomenon in financial markets. We believe that the strategy can be improved by not only looking at the stock trend but also augmenting the signal with the text sentiment extracted.

Here we analyze a momentum-based trading strategy. A myriad of momentum approaches can be applied, but to keep it simple, we measured the momentum metric as the mean returns (μ) over the last month normalized by the standard deviation (σ). So, we have

$$M = \frac{\mu(t)_i}{\sigma(t)_i} \tag{17}$$

$$\mu(t)_i = \frac{\sum_{t=1}^{T} r(t)_i}{T} \tag{18}$$

$$\sigma(t)_i = \sqrt{\frac{\sum (r(t)_i - \mu(t))^2}{T}} \tag{19}$$

Above, T is the estimation period and $r(t)_i$ is the return of stock i at time t. Then, the strategy is built by sorting the stock from higher momentum stocks to lower momentum and the stocks belonging to the top decile are the winners and bottom decile the losers. To avoid look-ahead bias, signals with closing price at time t are used at time $t+1$.[1]

Plot 4 shows a "pure momentum play". It shows the difference between the top decile and the bottom decile based on the 1-month momentum metric.[2]

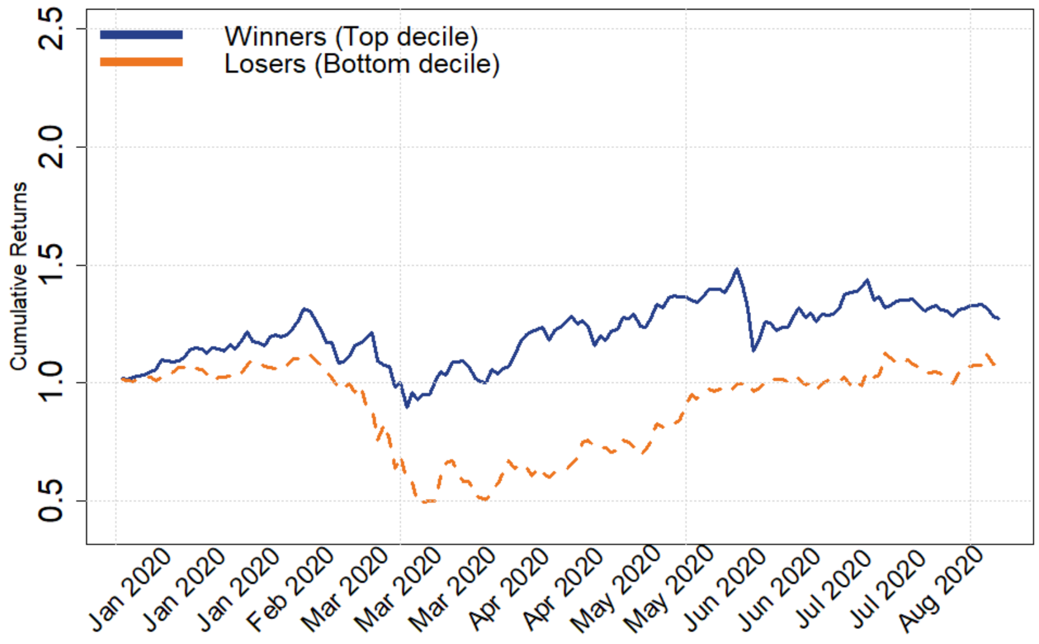

Figure 4: Pure momentum strategy based on stock returns normalized by the standard deviation. Portfolios are equally-weighted (1/N).

4.2 Augmented Momentum Strategies

The next step is to test if the signal can be augmented (and improved) by using sentiment analysis. Generally, signals (alphas) are ephemeral and therefore, investors do not use one or a few signals but instead tens, hundreds or even thousand of alphas to improve signal stability and remove noise.

[1] This is a conservative measure. The strategy could be made in such a way that the signal is used in the same day t without issues.

[2] Transaction cost are set to be flat $\sim 0.01\%$, according to the fees and commissions of Interactive Brokers. Additional costs like market impact and slippage are not considered.

Plot 5 depicts the momentum strategy augmented with the most naive sentiment analysis (VADER). Comparing with the pure momentum play (see Plot 4), the performance worsen. This can be due to two aspects. On the one hand, the predictive power of the signal decrease, and on the other hand, the signal is less stable and therefore, it increases transaction costs.

In addition, it is interesting to notice that the Loser stocks, which are ranked based on negative sentiment, perform even better than the Winner stocks, i.e., those with positive momentum and positive sentiment. The first conclusion is that VADER could act as a contrarian indicator, although further analysis would need to be conducted to conclude why the strategy behaved in such a way.

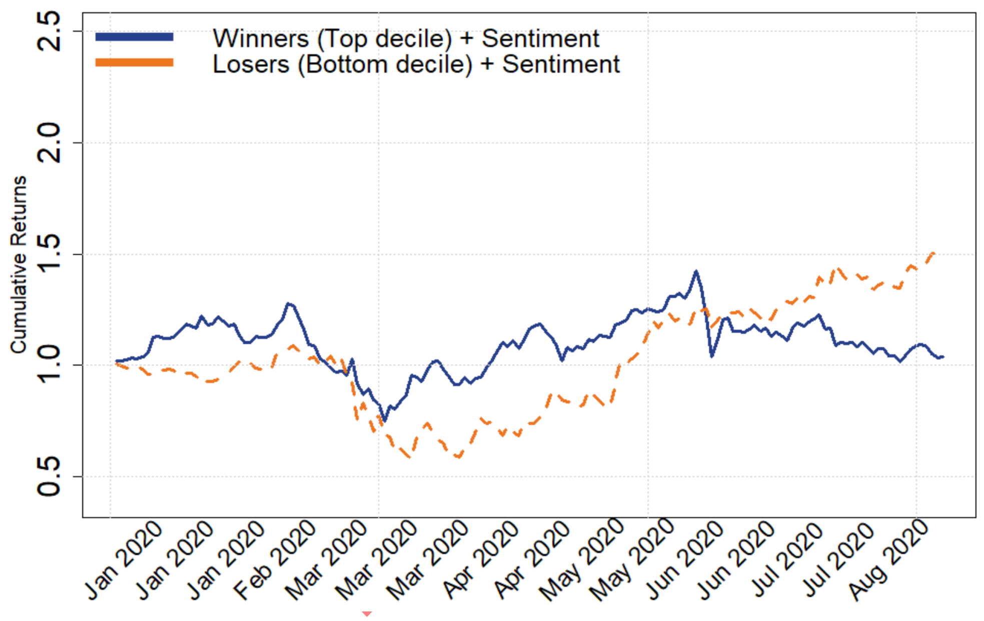

Figure 5: Momentum strategy with signal augmented by VADER sentiment.

Plot 6 shows the momentum strategy with the Logistic classifier.

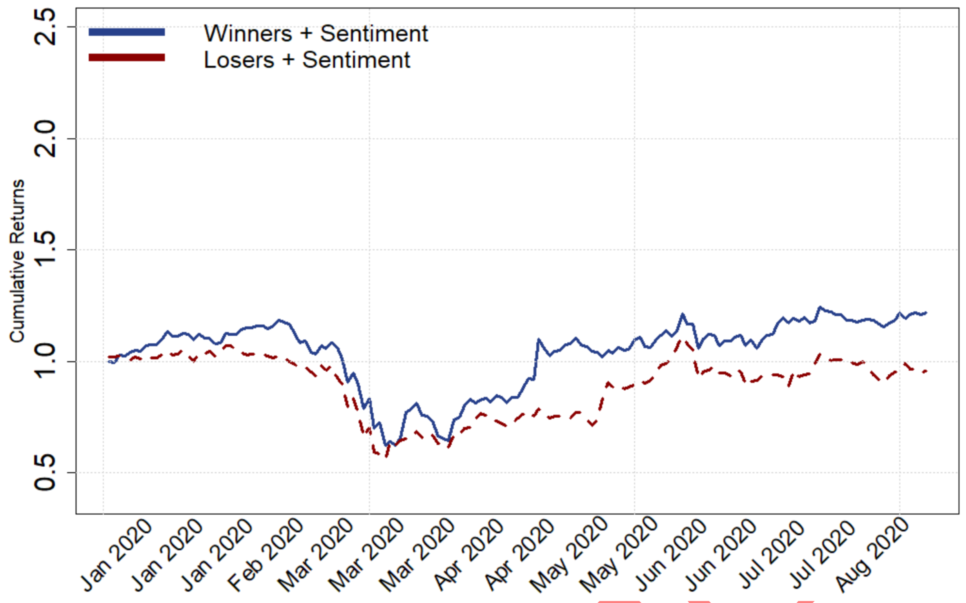

Figure 6: Momentum strategy with signal augmented by Logistics sentiment classifier.

Although it outperformed the VADER classifier, it still does not perform better than the pure momentum play. We also tested the strategy using the Naive Bayes classifier, obtaining results similar to those of Logistic regression.

Plot 7 depicts the momentum strategy augmented with the CNNs based sentiment analysis. Opposite to the VADER approach, this is a wiser approach to extract sentiment from investors posts (see Subsection 3.5). When we compare with the pure momentum play (see Plot 4), the performance improves significantly. Opposite to the VADER signal, the CNNs based signal improved the stability and predictive power of the blended signal, improving the overall strategy.

Notice that during the COVID-19 crash, the winner decile (augmented by the CNNs sentiment) performed remarkably well, reducing significantly the drawdowns relative to the prior approaches.

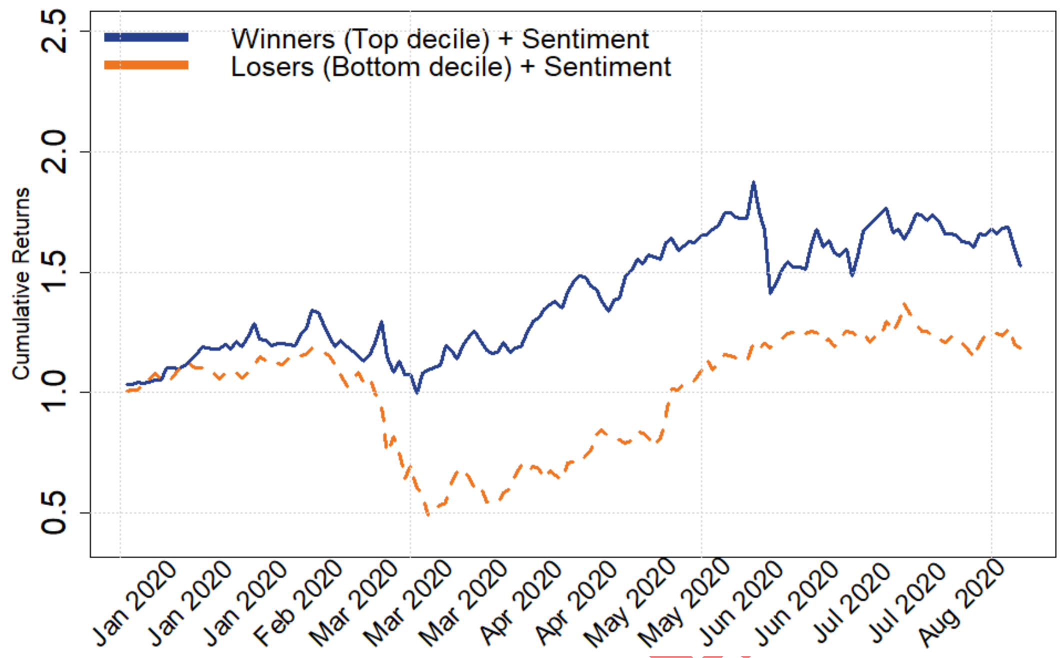

Figure 7: Momentum strategy with signal augmented by sentiment based on CNNs.

Plot 8 shows the Winner strategy (top momentum stocks) to compare the performance between the pure momentum play and the momentum plus CNNs based sentiment. It can be seen that the spread is economically significant, even considering any additional transaction cost that the sentiment re-balancing could add.

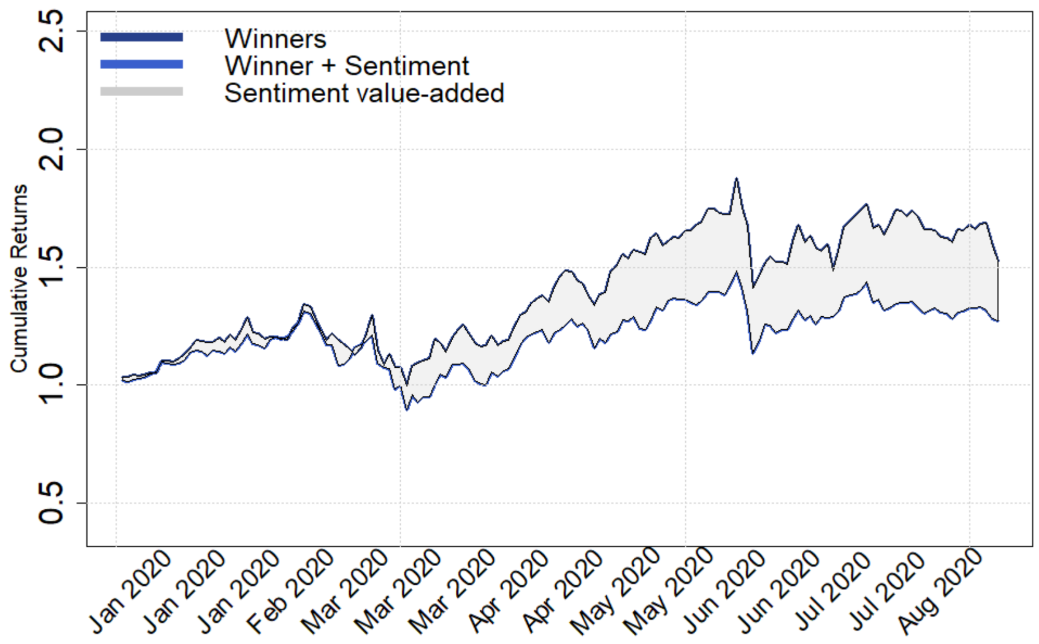

Figure 8: Gray shaded area shows the value added by the sentiment.

4.3 Augmented Mean-Reversion Strategies

Mean Reversion opposite to trend-following/momentum bets, mean reversion strategies can be exploited by identifying weak trends. Again, augmenting the signal with the text sentiment extracted, the strategy can be improved. E.g., a stock with high momentum which starts to show a negative sentiment could be a good candidate for a mean reversion bet.

Again, for mean-reversion, several approaches can be used to measure the signal. These mean-reversion strategies are more suitable for the intraday frequency, however we attempted to build a simple mean-reversion indicator using daily data as following

$$\text{IBS} = \frac{P_C - P_L}{P_H - P_L} \tag{20}$$

IBS means Internal Bar Strength. P_C is the last closed price, P_H is the last high price and P_L is the last low price. The idea here is simple; stocks with IBS close to 1 are "expensive" and those stocks with IBS close to 0 are “cheap”. So, an investor would buy the expensive stocks and short the cheaper ones.

We tested this trading strategy by augmenting the signal with the sentiment indicators. Plot 9 shows that this strategy works as a momentum/trend-following approach and not as a mean-reverting one. The explanation is simple. Mean-reversion strategies work better for intraday data, however, for daily (and lower frequencies), the momentum effect is stronger than the mean-reversion. This is why, the “expensive stocks” (high momentum) outperforms “cheap stock” (low momentum). To show how the IBM indicator can be used for momentum purposes, we also show a long/short strategy that buys the high momentum (expensive) stocks and shorts the low momentum (cheap) stocks. The signal is augmented with the sentiment based on CNNs.

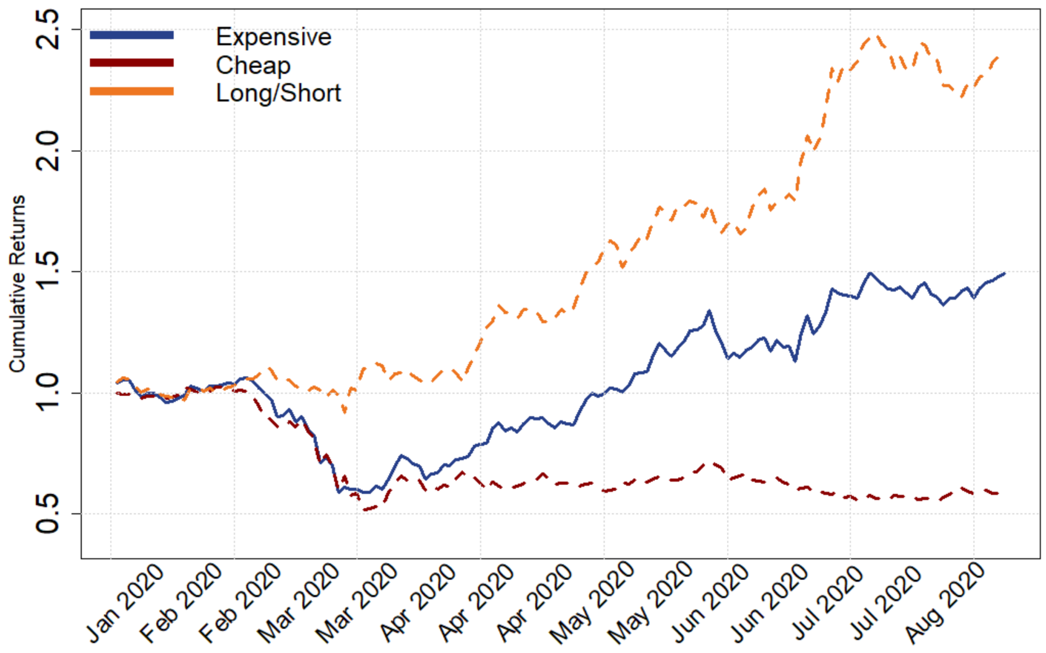

Figure 9: Mean-reversion strategy with signal augmented by sentiment based on CNNs. Portfolios are equally-weighted (1/N).

We can conclude that this kind of mean-reversion indicators works better as a momentum indicators for the case of low-frequency data. However, it is interesting to see that even in that case, the (momentum) strategy performed remarkably well. We also tested this strategy with the other sentiment approaches described in the sections above but none of them outperformed the CNNs sentiment classifier.

5 Conclusions and Further Research

A key aspect in continuing our research is to expand the data source and not just be limited to Reddit. For example, using Twitter, Facebook and news articles could help to further improve the classification and hence the trading strategies. A good way to improve the project would be set up a sentiment collecting pipeline that can use a variety of data sources. This way if one data source becomes unpopular another could replace it quickly. We have included the start of a way to improve the data collecting by using Spark streaming with Scala for Twitter. The script filters all live tweets and aggregates the number of mentions of a ticker each second using a 5 minute sliding window. This could be an effective way to get a quick overview of the most active stocks.

Another way to continue our research would be to test more trading strategies, like "volatility-based options bets", which are a common way to get exposure to the volatility of a determined asset is to buy calls and puts in-the-money or out-of-the-money (Straddles/Strangles). We believe that the amount of posts and the sentiment of a stock can help to identify stocks with higher probability of showing extreme movements, profiting from a long volatility position.

References

Avellaneda, M. and Stoikov, S. (2019) High-frequency trading in a limit order book. *Quantitative Finance* 8(3): 217-224.

Brownlees, C.T. and Gallo, G.M. (2006) Financial econometric analysis at ultra-high frequency: Data handling concerns. *Computational Statistics & Data Analysis* 51(2006): 2232-2245.

Meucci, A. (2005) Risk and Asset Allocation. 1st edition, Springer-Verlag.

Potters, M. and Bouchaud, J.-P. (2003) More statistical properties of order books and price impact. *Physica A* 324: 133–140.

Tsay, R. S. (2013) Multivariate Time Series Analysis: With R and Financial Applications. 1st edition, Wiley.

Tsay, R. S. (2010) Analysis of Financial Time Series. 3rd edition, Wiley.

Wursthorn, Frankl-Duval, and Zuckerman. (2020) Everyone's a Day Trader Now,
`https://www.wsj.com/articles/everyones-a-day-trader-now-11595649609`