The information content of sentiment data from StockTwits

Joyce Cahoon

Abstract

We assessed the performance of a cross-sectional, long-short US equity strategy that relied soley on sentiment data from StockTwits. Factor analysis indicates a level of predictive alpha in our sentiment metric with full exposure five days after the signal is made available in our portfolio. Our performance over the November 2018 period resulted in a final rank of 105 out of 261 submissions and a total score of 0.338 on Quantopia. Its performance is mediocre at best and can be attributed the fact that the tuning mechanism for input parameters in the algorithm was based only on the backtest returns. Our results do not yet dismiss the value of leveraging unstructured data from online conversations to predict returns.

Contents

Introduction

As fund managers, we were tasked with constructing a cross-sectional, long-short US equity strategy on Quantopian. There were not many explicit constraints on the specific strategy we utilized, but it must pass the following criteria: (i) trade liquid stocks, (ii) have no more than 5% of capital invested in any one asset, (iii) have no more than 10% net dollar exposure, (iv) achieve mean daily turnover between 5% and 65% over a 63-trading-day rolling window, (v) attain gross leverage between 0.8x and 1.1x, (vi) have low correlation to the market, (vii) have less than 20% exposed to each of the 11 sectors as defined on Quantopian, and (viii) result in positive returns. While the last return criteria was not a constraint we included in our optimization, we did design our algorithm with the rest of the seven criteria in mind (Quantopian 2018).

The algorithm I eventually submitted was one based purely on the information content from StockTwits; more specifically, the 5-day moving average of the bull minus bear intensity associated with each tradable stock. We rely on this metric to assign each stock's weight in our portfolio, subject to certain constraints so that our algorithm passes Quantopian's criteria to remain in competition. Given the large literature around news and social media sentiment in driving volatility and liquidity in the market, we wanted to examine the power of this unstructured data in driving returns.

Our algorithm's performance over the November period resulted in a final rank of 105 out of 261 submissions and a total score of 0.338. Its performance is mediocre at best and can be attributed the fact that the tuning mechanism for my input parameters was based only on the backtest returns. Given our constraints, e.g. the length of moving window for our sentiment metric, were overfit to a backtest window between 2016 and 2018, our strategy may have experienced bad out-of-sample performance.

Nevertheless, this strategy did perform within the top 40% of all submissions, which corroborates many findings in literature that social media has an effect on driving prices. Since the majority of literature suggests extreme sentiment or extreme volume of online media content tend to drive returns, and the month of November can be characterized by extreme volatility, this may explain the high performance of our sentiment strategy. As expected, for December, the peaks in the VIX subsided given greater clarity around the Trump administration's trade policy and Fed's decision to stop raising rates, and our strategy fell a few ranks to 113 with a score of 0.316 (Gurdus 2018).

Related Work

The first paper that explored the relationship of text—unstructured data—in driving stock prices was Cutler et al. (1988)'s "What moves stock prices?" At the time, the field of computational linguistics was yet sophisticated enough to process large collections of text, so Cutler and his team examined 49 major events discussed in the NY Times Business desk and assessed the market changes associated with each event. Unfortunately, given the number of significant market moves associated with days where nothing significant was released, they were unable to identify any link between market moves and news, possibly stymeing explorations into this realm (Cutler et al. 1988). Fortunately, another empirical study emerged in 1998, which leveraged information on online message boards instead, identifying strong correlations between stocks with high message volume and high market valuation. In fact, Wysocki found message volume forecasted not

only abnormal stock returns the next day, but also trading volume the next day (Wysocki 1998).

Yet, given these results and similar findings from papers like Bagnoli et al. (1999), Antweiler and Frank (2004), corroborating the link between prices and social chatter, a number of studies find the opposite. Tumarkin and Whitelaw (2001) found no relationship for equities in the tech sector and the message activity on RagingBull.com. Das and Chen (2007) developed a classifier for investor sentiment from message board text, but was unable to to demonstrate its ability to forecast returns. Dewally (2003) found recommendations exchanged on sites like `misc.invest.stocks` and `alt.invest.penny-stocks` had no effect on the return characteristics of the stocks under discussion.

Despite these mixed claims, newer studies arose reporting the opposite. The seminar paper by Tetlock (2007) concluded high negative sentiment in the Wall Street Journal predicted temporary downward price pressure, echoing the thoughts expressed in Wysocki (1998) that extreme news affects market movement. Additional works that build on Tetlock's use of news as a fundamental factor is Mitra et al. (2008), Leinweber and Sisk (2011) and Fang and Peress (2009), each providing evidence for the predictive value in social chatter. In this class, we were also directed to the paper by Agrawal et al. (2019), which provides a great overview of literature that further promotes the use of Twitter and StockTwits to predict stock volatility, liquidity and return.

StockTwits Data

Key to our strategy is the output from the `bull_minus_bear` API call as it is this signal that is fed into the optimization function and this result that determines the order size of each asset. While there is no disclosure on the natural language processing model that calculates the bull and bear intensity of each social media message, it is helpful to look at examples and understand the nature of posts that ultimately determine the power of our sentiment measure.

As shown in Figure 1, there is a lot of noise in StockTwit data as users each utilize the platform for different reasons and do not have to disclose any of their personal stakes in the equities they support. Logically, users that are more influential should have a higher impact in driving overall investor sentiment, and thus driving prices, but there currently is no easily available API call that provides such a weighting. Moreover, these influencers change over time, increasing the difficulty around identifying posts that might hold more significance than another. As a result, relying on the aggregate bullish and bearish intensity averaged daily across all tagged StockTwits messages is the best surrogate we have to understanding community sentiment for a specific stock.

From the free version of StockTwits API, we have access to the following daily metrics for 10,214 liquid US equities ("StockTwits" 2018):

- *Bull and bear scored messages*: Total number of daily messages that were labeled either "Bullish" or "Bearish" on the StockTwits platform by the user.
- *Bull or bear intensity*: Daily score between 0 and 4 aggregated across processed messages. 0 means no bull or bear sentiment was detected by StockTwits proprietary trading algorithm. 4 means very high bull or bear sentiment was detected. Exploratory data analyses indicate that very few stocks receive a

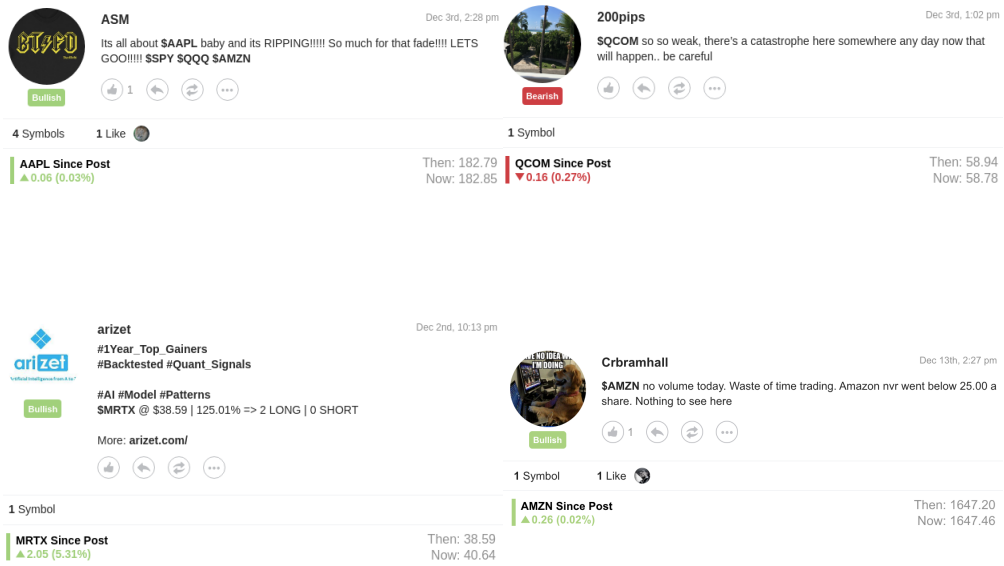

Figure 1: Four examples of messages posted on StockTwits. The bull and bear indicator is tagged by the user, and all stock tickers that follow the cashtag symbol is associated with this bull or bear flag. The bull and bear intensity is then computed by a proprietary StockTwits algorithm based on the content within each individual post.

score of 4.

- *Total scanned*: Total number of daily messages that appear on StockTwits platform that has at least one cashtag associated with it. The message is included in this count whether it is labeled or unlabeled and whether it can be processed by the StockTwist language algorithm or not.

Since *disagreement* among message content has been shown to be associated with greater liquidity and volatility, we focus on the metric of `bull_minus_bear` as it is one proxy for disagreement among investor opinion (Antweiler and Frank 2004). Given the overall trend in this metric is not stable as we will see in Figure 12 and 13, a moving average is taken. Eventually, we settle on 5-day moving average as the mean information coefficient (IC) is positive that this point. The IC is often chosen to evaluate whether a factor is predictive of returns as it is more robust to outliers and normality assumptions. It allowed us to better tease out whether two series move in concert or not.

IC was thus calculated from the rank of each data pair in each series—our intensity difference versus returns—as $IC = 1 - 6\sum_i d_i^2/n(n^2-1)$ where d_i represents the difference in ranks. As shown in the top left of Figure 2, prior to 5 trading days, our sentiment signal hurts our returns in that its forecast is negative, but after 5 trading days, we gain some predictive power. This is further corroborated in the top right plot in that returns are highest when the signal is delayed by four days. Given the IC results, we may want to build greater exposure to this factor as it still benefits us after 10 trading days. We can also leverage the positive effects of this factor by increasing our turnover constraints as it appears it does not hurt our portfolio to keep in there for a longer period of time.

Another asset is that our sentiment factor has relatively low exposures throughout, with most of the returns from volatility risk. However, there does appear to be persistent exposure to value and short term reversal, in that it is shifted to the left of zero; and persistent exposure to size and momentum, in that it is shifted to

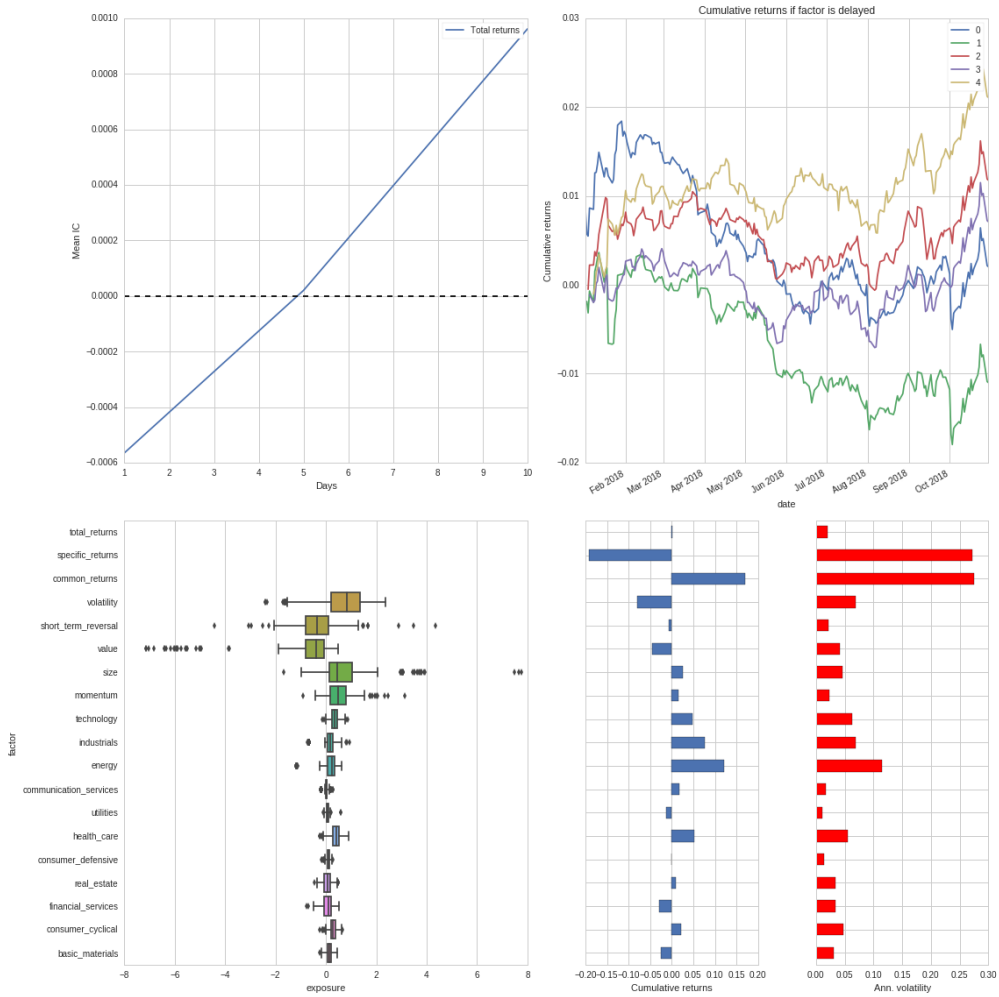

Figure 2: Examination of (top-right) information coefficient averaged over all possible asset returns, (top-left) cumulative returns when factor is delayed, (bottom-left) risk exposures, and (bottom-right) cumulative returns and volatility.

the right of zero. Ideally, we would choose a factor that does not display any skew, but our skews do not appear that large. We benefit from the fact that the width of each of our box and whisker plots are not that wide, so our exposures do not vary much over time.

Unfortunately, when we examine our exposures through cumulative returns and volatility, we find that our sentiment factor may not be as strong as we desired. As shown by the bottom left bar graph in Figure 2, most of our returns are obtained from exposure to common risk factors, and, as shown in the right bar graph, are in fact driven by these common risk factors. Ideally, we would like to strip away the contribution from these common risk factors as portfolio performance from this exposure can easily be replicated from ETFs or other easy, cost-effective methods. Alternatively, we can work to identify asset classes to include in our portfolio that would offset the exposure risks identified. Nevertheless, this factor analysis was conducted only over the sentiment signal in 2018, so there exist other extended periods prior to 2018 in which our factor may over or underperform.

Trading Strategy

Given the ease of implementation and some good alpha characteristics as delineated above, we chose to use the sentiment factor in our trading strategy. Continuous backtesting was run to derive the constraint parameters as outlined:

1. We filter a universe of liquid assets from `QTradeableStocksUS` that pass the following criteria:
   - It is not trading within 2 days of earnings as assets are generally more volatile within these dates.
   - It has not been announced as an acquisition target to further reduce any possible volatility.
   - We are able to calculate a 5-day moving average of the bull-minus-bear signal.
2. We build an alpha vector for the universe of liquid assets filtered from our step above. The alpha model we use is quite simple: we rank the assets by its bull-to-bear intensity, averaged over the past 5 days as evaluated from StockTwits, and find a set of new portfolio weights that maximizes the sum of each asset's weight times this alpha value. Our objective defined in `MaximizeAlpha` is thus simply a function of this sentiment datastream as we believe this ranking is similar to expected returns of each asset. Our routine effectively goes long on assets with high bullish signal and short on those with a high bearish signal.
3. Once a week, we calculate the portfolio that maximizes the alpha-weighted sum of our position sizes, subject to the following constraints:
   - Our portfolio maintains a gross leverage of, or less than, 1.0x.
   - Our portfolio has no more than 5% in any single asset.
   - Our portfolio does not pass mean daily turnover of 80%.

Our simple strategy can be summarized as follows:

$$\max_{\boldsymbol{w}\in\mathbb{R}^n} \boldsymbol{\alpha}^T\boldsymbol{w} \quad \text{subject to} \quad |w_i| \leq 0.05,\ \sum_i |w_i| \leq 1.00,\ \sum_i |w_i - w_{i-1}| \leq 0.80,\ \sum_i w_i = 1$$

where $\boldsymbol{w}$ represents the weights attached to each asset in our optimal portfolio.

Backtest Analysis

We test our trading strategy with $10 million in capital from September 30, 2016 to December 3, 2018. Table 1 outlines our performance during this period:

Table 1: Key metrics from our final backtest between 2016 and 2018.

Annual Return	1.7%	Skew	0.18
Cumulative Return	3.6%	Gross Leverage	0.99
Annual Volatility	4.0%	Daily Turnover	18.6%
Sharpe Ratio	0.43	Alpha	0.01
Max Drawdown	-3.1%	Beta	0.02

While the annualized and cumulative returns are low, at 1.7% and 3.6% respectively, we settle on the

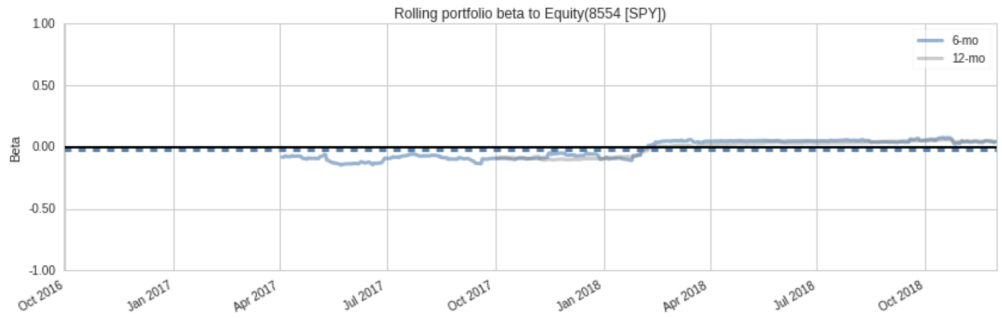

Figure 3: Rolling beta calculated within the backtest period suggests the strategy is market neutral.

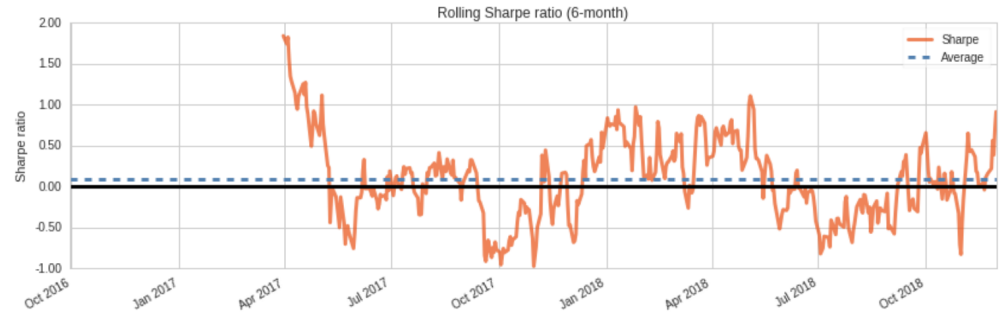

Figure 4: Rolling Sharpe calculated within the backtest period suggests it is highly inconsistent.

parameters in our trading strategy as it has a number of good properties. For a start, its overall beta of 0.02 is ideal, suggesting low correlation with the market and less volatility. As shown in Figure 3, this overall average is representative as the rolling beta does not deviate far from this value.

Our Sharpe ratio of 0.43 suggests low risk-adjusted return. Generally, leverage can be applied to such low risk strategies to increase the absolute return; however, when we examine the Sharpe ratio over our backtest period, we find high variability. This lack of consistency in Figure 4 makes our trading strategy much more difficult to assess.

From our cumulative returns in Figure 5, we see our underperforming Sharpe ratios, within the June 2017 and January 2018 window, as well as the May 2018 and October 2018 window, occur during periods where the cumulative market returns are increasing at a rate much higher than that of our portfolio. This makes sense given our initial exploratory result; our sentiment metric has the highest exposure to volatility risk.

When we align our rolling Sharpe ratios in Figure 4 against our underwater plot in Figure 6, we also find when our strategy performs worse. It narrows our original windows to months surrounding the July 2017 and June 2018. Coincidentally, both these periods are marked by heavy gains in the broader market due to good macroeconomic news like job gains and high earnings. Given our low beta, our strategy might result in better performance in times of market decline.

Overall, our drawdowns are minimal, none surpassing the 4% mark. We thus move on to examine whether there is any seasonality to our strategy in Figure 7. While it is not explicit, it does appear our strategy has performed well within the month of November for 2016, 2017 and 2018. The annual returns, however, are quite low in 2017 and 2018 compared to 2016, which could be explained by the traffic on StockTwits platform.

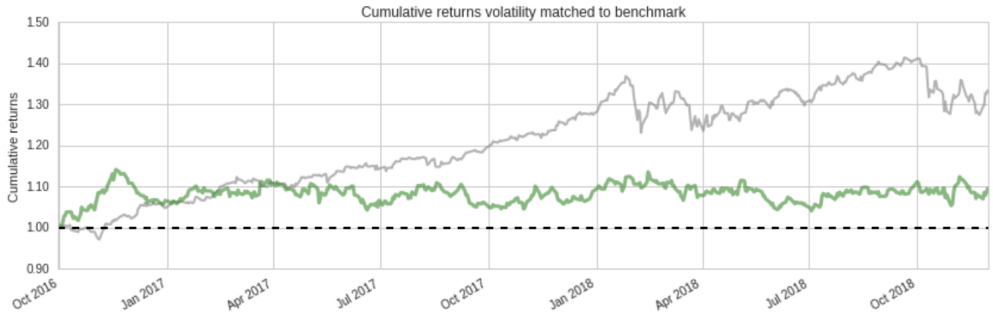

Figure 5: Cumulative returns of our portfolio against the market portfolio.

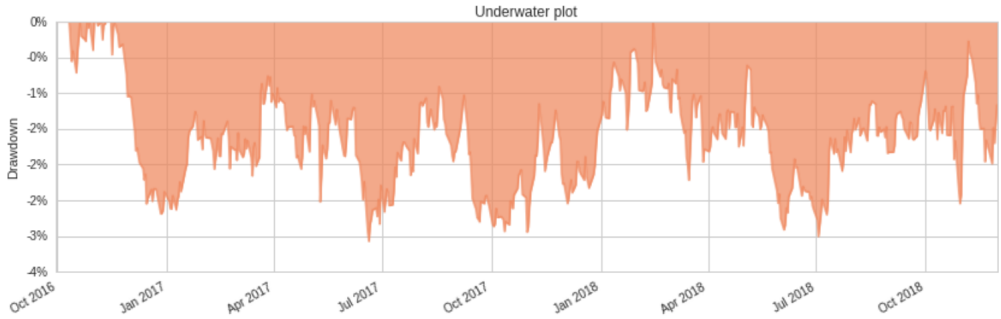

Figure 6: Underwater plot highlighting periods of max drawdown.

As we will examine further on an individual stock basis, it appears our bull minus bear intensity is evaluated on very little data.

As we saw in our exploratory analysis in Figure 2, we also see that we have relatively low exposures to each industry in Figure 8, protecting us from idiosyncratic shocks. We also see in Figure 9 and 10 an expected pattern over our backtest period as we fixed these metrics in our original strategy.

Our original constraint of $|w_i| < 0.05$ is also satisfied as indicated by the top 10 long and short positions. No individual security can have a significant impact on portfolio returns as shown in Figure 11. To further understand our use of our sentiment metric, we examine the top 5 long and short stocks individually, extracting

Figure 7: (Left) Monthly returns of our overall portfolio broken down by year. (Center) Annual returns. (Right) Histogram of monthly returns.

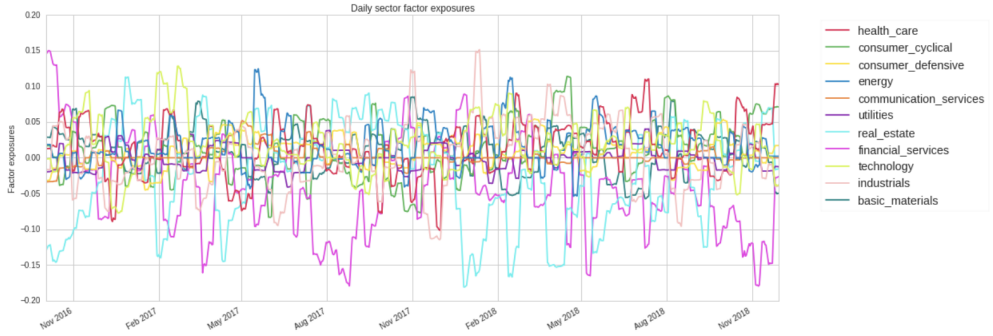

Figure 8: Daily factor exposures to each of the 11 industries as defined on Quantopian.

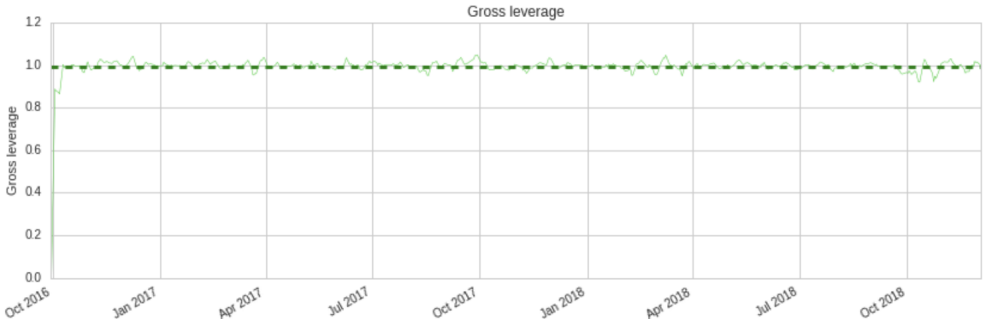

Figure 9: Gross leverage fluctuates around 1.0x as originally set in our trading algorithm.

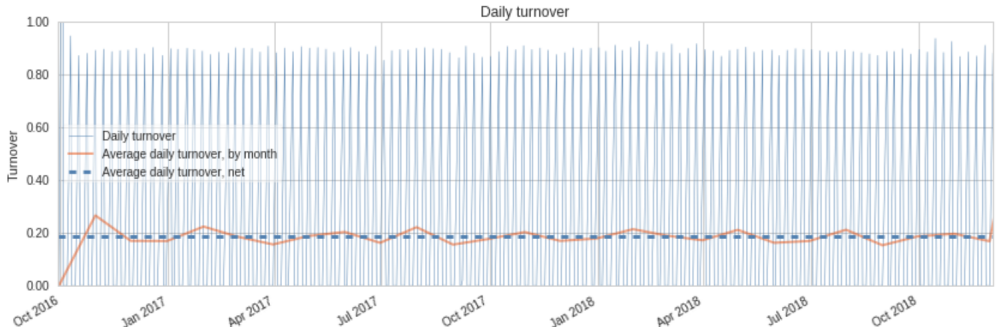

Figure 10: Daily turnover of 0.80x shown weekly as originally set in our trading algorithm.

Top 10 long positions of all time	max	Top 10 short positions of all time	max
SAGE-47332	2.53%	CMPR-27674	-2.48%
SGEN-22563	2.44%	SYNH-48027	-2.46%
ESPR-44989	2.41%	MCY-5017	-2.45%
RGEN-6449	2.39%	MRTX-45080	-2.41%
DLTH-49615	2.36%	SSD-11386	-2.40%
TREX-20028	2.35%	FSM-41915	-2.39%
LOGI-16649	2.34%	BZH-10728	-2.37%
FFIN-10148	2.33%	FORM-25182	-2.34%
NGHC-44929	2.30%	AVNS-47929	-2.31%
EIGI-45735	2.30%	PARR-43375	-2.29%

Figure 11: Top 10 long and short positions within the backtest period.

each respective count of number of bull and bear messages assessed on StockTwits, as well as the 5-day rolling average of its bull minus bear intensity within our backtest window in Figure 12 and 13.

Performance

As shown below and alluded to earlier, our results were above average as the overall market in November was highly volatile. Given our strategy gets most of its returns from volatility, we were able to achieve a relatively high score of 0.338. Moreover, we almost experienced no drawdown despite the drawdowns in the broader US equities market. Our final leaderboard results at the end of our one-month trading period from November 1^{st} to November 30^{th} are shown in Table 2.

Table 2: Leaderboard results after the one-month trading window of November 2018

Metric	Our Result	Overall
rank	105	-
score	0.338	0.35
max_beta_to_spy_126day	0.076	0.14
max_cumulative_common_returns	0.009	0.04
max_leverage	1.047	1.05
max_max_drawdown	0.000	-0.00
max_net_dollar_exposure	0.032	0.04
max_total_returns	0.025	0.14
min_total_returns	-0.007	-0.02
max_turnover	0.905	1.07
max_volatility_126day	0.044	0.06

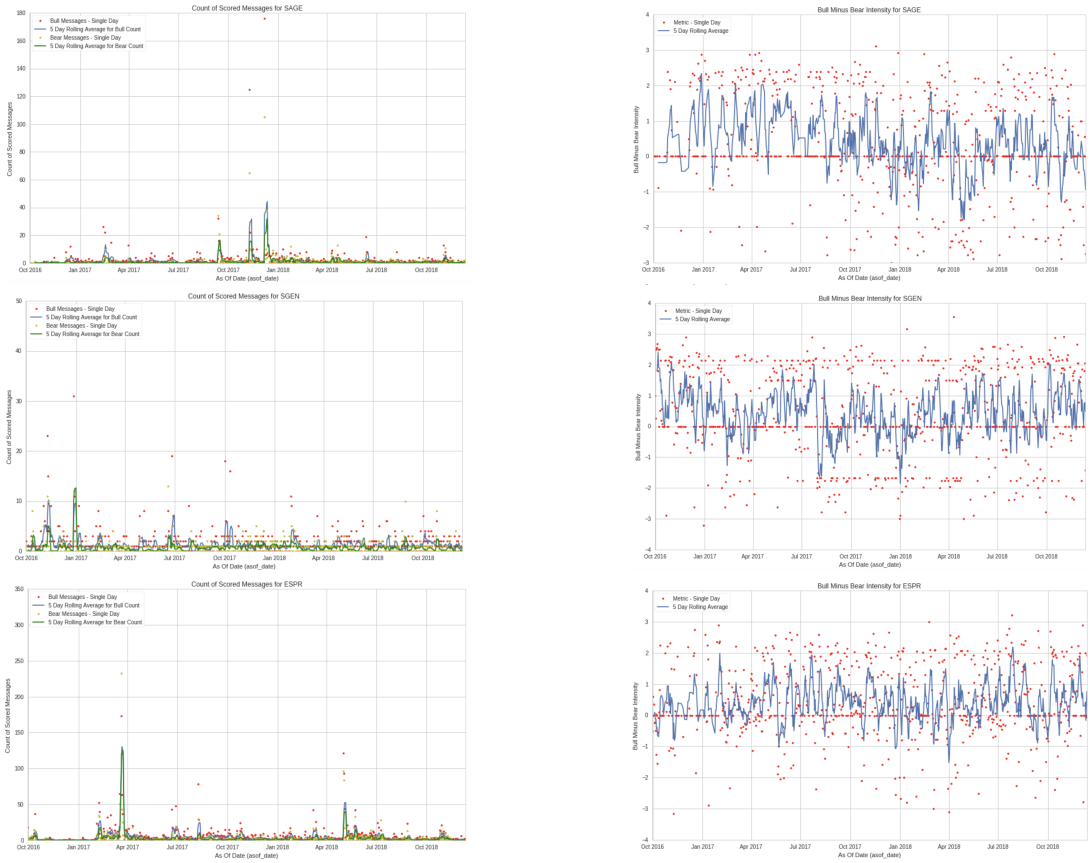

Figure 12: Raw bull and bear message counts and bull minus bear intensities for the top 3 long positions. From top to bottom, we have SAGE, SGEN, and ESPR shown.

Figure 13: Raw bull and bear message counts and bull minus bear intensities for the top 3 short positions. From top to bottom, we have CMPR, SYNH, and MCY shown.

Discussion

Closer examination of the data in Figure 12 and 13 suggests that our sentiment indicator should not have been the *primary* component of our trading strategy. As shown by the number of scored bull and bear messages in red and yellow in Figure 13, the highest number of messages that were evaluated to calculate the bull or bear intensity was less than 16. These 16 messages appeared around February 2018 for Syneos Health (SYNH) right around when SYNH released their positive earnings revision, resulting in a spike of their stock price from $35.80 to $41.90 per share (Edward 2018). However, our strategy does not trade on information within 2 days of any earnings period, so despite StockTwits providing more text data than usual, we do not leverage this additional information.

Unlike the top equities that we have shorted, in Figure 12, the bull and bear intensies are based off of a more substantial set of StockTwits. In fact the overall bull and bear count of messages processed for our top long equities are much higher with a max count around 180, 30, and 250 for SAGE, SGEN, and ESPR. In general, the number of bull messages are greater than the number of bear messages, suggesting most users on StockTwits generally lean towards promoting *buys* rather than *sells*. Moreover, we see that the peaks in message volume are oftentimes not associated with earnings or acquisition announcements, but other major corporate milestones. For example, the peak in message volume for Sage Therapeutics (SAGE) occurs around December 2017. These ~180 messages can be attributed to an announcement made by the CEO in which SAGE received positive results after its Phase 2 trial for a drug that treats depression (Aiello 2017). We can also see a peak in message volume in the prior month of November, which can be attributed to the successful clinicial trial results for another drug for postpartum depression (Lovelace 2017). Both of these events are marked by high peaks in our bull minus bear intensity, providing some impetus for the value in our sentiment metric.

In general, while our approach is market neutral and has benefitted us during a period of high volatility in asset prices, it is too simple. We should have adhered to ideas from literature that have shown time and again that sentiment can be utilized to *augment* a traditional long-short equity strategy (Agrawal et al. 2019). Instead, our optimization routine depended *entirely* on our sentiment metric, which explains its poor performance overall. To build on our simplistic approach, a number of avenues we can consider include:

1. *Build a classifier from scratch*: As we have examined, the volume of data provided by StockTwit is not enough to fully capture general investor sentiment on a stock. We should attempt to build our own classifier from possibly richer sources of messages (e.g., Weibo, Twitter) that can provide a more robust measure of the bullish and bearish signals. Moreover, constructing our own language processing algorithm provides a better means to filter out noise or manipulative StockTwits as users each have their own agenda for making excessive claims about a particular asset.

2. *Include fundamental factors*: Given the extensive literature on factor modeling (Fan and Yao 2015), we should consider leveraging these time-tested metrics and ratios, like market cap and book-to-price, that have captured the financial characteristics of an asset. We can start with the typical (i) excess return of small market cap minus big market cap companies, (ii) excess return of companies with high book-to-price against low book-to-price ratios, and (iii) excess return of companies that overperformed to those that underperformed, and study how the inclusion of these factors along with our sentiment index might affect returns. We might even find that some factors are more efficient than others, so

swapping them in for our sentiment index might be necessary. Once we understand exactly how much we are exposed to specific factors, given the lack of predictability in the market in general, we should consider beta hedging to avoid any big dependencies on performance on a certain factor. This essentially entails taking the exposure β we have to a factor and shorting it by the proportional value of our total portfolio βM.

3. *Comprehensive exploratory data analysis*: There are basic steps we should have carried out prior to executing our strategy on Quantopian. Namely, we should have checked for normality, homoskedasticity, and autocorrelation in our model, as violations results in parameters that often biased, inconsistent and inefficient. If these assumptions hold, then we can actually normalize our factor values, allowing for them to be easily comparable across different asset classes. If the assumptions hold weakly, we might consider other transformations, like Winsorization, that will still allow us to build a robust statistical model for trading.

Given some good characteristics and performance in our approach overall, we have yet ruled out the value of using information from online conversations like StockTwits, as a means to build a compelling long-short US equity strategy.

Appendix

Quantopian Submission

```
# Import Algorithm API functions
from quantopian.algorithm import (
    attach_pipeline,
    pipeline_output,
    order_optimal_portfolio,
)
# Import Optimize API module
import quantopian.optimize as opt
# Pipeline imports
from quantopian.pipeline import Pipeline
from quantopian.pipeline.data.psychsignal import stocktwits
from quantopian.pipeline.factors import SimpleMovingAverage
# Import built-in universe and Risk API method
from quantopian.pipeline.filters import QTradableStocksUS
from quantopian.pipeline.experimental import risk_loading_pipeline
# Get event data
from quantopian.pipeline.factors.eventvestor import (
    BusinessDaysUntilNextEarnings,
    BusinessDaysSincePreviousEarnings,
)
from quantopian.pipeline.filters.eventvestor import IsAnnouncedAcqTarget
from quantopian.pipeline.factors import BusinessDaysSincePreviousEvent
def initialize(context):
    # Constraint parameters
    context.max_leverage = 1.0
    context.max_pos_size = 0.05
    context.max_turnover = 0.8
    # Attach data pipelines
    attach_pipeline(
        make_pipeline(),
        'data_pipe'
    )
    attach_pipeline(
        risk_loading_pipeline(),
        'risk_pipe'
    )
    # Schedule rebalance function
    schedule_function(
```

```
        rebalance,
        date_rules.week_start(),
        time_rules.market_open(),
    )
def before_trading_start(context, data):
    # Get pipeline outputs and
    # store them in context
    context.output = pipeline_output('data_pipe')
    context.risk_factor_betas = pipeline_output('risk_pipe')
# Pipeline definition
def make_pipeline():

    not_near_earnings = ~((BusinessDaysUntilNextEarnings() <= 2) |
      (BusinessDaysSincePreviousEarnings() <= 2))

    not_acq_tar = ~IsAnnouncedAcqTarget()

    universe = (
        QTradableStocksUS()
        & not_near_earnings
        & not_acq_tar
    )

    sentiment_score = SimpleMovingAverage(
        inputs=[stocktwits.bull_minus_bear],
        window_length=5,
        mask=universe
    )
    return Pipeline(
        columns={
            'sentiment_score': sentiment_score,
        },
        screen=sentiment_score.notnull()
    )
def rebalance(context, data):
    # Create MaximizeAlpha objective using
    # sentiment_score data from pipeline output
    objective = opt.MaximizeAlpha(
      context.output.sentiment_score
    )
    # Create position size constraint
    constrain_pos_size = opt.PositionConcentration.with_equal_bounds(
        -context.max_pos_size,
```

```
        context.max_pos_size
    )
    # Constrain target portfolio's leverage
    max_leverage = opt.MaxGrossExposure(context.max_leverage)
    # Constrain portfolio turnover
    max_turnover = opt.MaxTurnover(context.max_turnover)
    # Constrain target portfolio's risk exposure
    factor_risk_constraints = opt.experimental.RiskModelExposure(
        context.risk_factor_betas,
        version=opt.Newest
    )
    # Rebalance portfolio using objective
    # and list of constraints
    order_optimal_portfolio(
        objective=objective,
        constraints=[
            max_leverage,
            constrain_pos_size,
            max_turnover,
            factor_risk_constraints,
        ]
    )
```

Data Exploration

```
# Import the free sample of the dataset
from quantopian.interactive.data.psychsignal import stocktwits_free  as dataset
# Import data operations
from odo import odo
# Import other libraries
import pandas as pd
import matplotlib.pyplot as plt
import datetime as dt
# Filtering for an equity (here SSD)
ssd = dataset[dataset.sid == 11386] # aapl is 24
ssd_df = odo(ssd.sort('asof_date'), pd.DataFrame)
ssd_df2 = ssd_df[ssd_df.asof_date >= dt.date(2016, 9, 30)]
plt.plot(ssd_df2.asof_date, ssd_df2.bull_scored_messages, marker='.', linestyle='None', color='r')
plt.plot(ssd_df2.asof_date, ssd_df2.bull_scored_messages.rolling(window = 5, center = False).mean())
plt.plot(ssd_df2.asof_date, ssd_df2.bear_scored_messages, marker='.', linestyle='None', color='y')
plt.plot(ssd_df2.asof_date, ssd_df2.bear_scored_messages.rolling(window = 5, center = False).mean(), co
plt.xlabel("As Of Date (asof_date)")
```

```
plt.ylabel("Count of Scored Messages")
plt.title("Count of Scored Messages for SSD")
plt.legend(["Bull Messages - Single Day", "5 Day Rolling Average for Bull Count", "Bear Messages - Sing
plt.plot(ssd_df2.asof_date, ssd_df2.bull_minus_bear, marker='.', linestyle='None', color='r')
plt.plot(ssd_df2.asof_date, ssd_df2.bull_minus_bear.rolling(window = 5, center = False).mean())
plt.xlabel("As Of Date (asof_date)")
plt.ylabel("Bull Minus Bear Intensity")
plt.title("Bull Minus Bear Intensity for SSD")
plt.legend(["Metric - Single Day", "5 Day Rolling Average"], loc=2)
```

Factor Analysis

```
from quantopian.pipeline import Pipeline
from quantopian.research import run_pipeline
from quantopian.pipeline.factors import SimpleMovingAverage
from quantopian.pipeline.factors import CustomFactor
from quantopian.pipeline.filters import Q1500US
from quantopian.pipeline.filters import QTradableStocksUS
from quantopian.pipeline.classifiers.morningstar import Sector
from quantopian.pipeline.data.builtin import USEquityPricing
from quantopian.pipeline.data.psychsignal import stocktwits
from quantopian.research.experimental import get_factor_returns, get_factor_loadings
import alphalens as al
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import empyrical as ep
import alphalens as al
import pyfolio as pf
universe = QTradableStocksUS()
pipeline_factor = SimpleMovingAverage(
    inputs=[stocktwits.bull_minus_bear],
    window_length=5)
pipe = Pipeline(screen=universe, columns={'alpha': pipeline_factor})
def compute_specific_returns(total_returns, factor_returns=None, factor_loadings=None, assets=None):
    if assets is not None:
        factor_loadings = get_factor_loadings(assets, start, end + pd.Timedelta(days=30))
        factor_returns = get_factor_returns(start, end + pd.Timedelta(days=30))
    elif factor_loadings is None or factor_returns is None:
        raise ValueError('Supply either assets or factor_returns and factor_loadings')
```

```
    factor_returns.index = factor_returns.index.set_names(['dt'])
    factor_loadings.index = factor_loadings.index.set_names(['dt', 'ticker'])
    common_returns = factor_loadings.mul(factor_returns).sum(axis='columns').unstack()
    specific_returns = total_returns - common_returns
    return specific_returns
def factor_portfolio_returns(factor, pricing, equal_weight=True, delay=0):
    if equal_weight:
        factor = np.sign(results['alpha'])
        bins = (-1, 0, 1)
        quantiles = None
        zero_aware = False
    else:
        bins = None
        quantiles = 5
        zero_aware = True

    pos = factor.unstack().fillna(0)
    pos = (pos / pos.abs().sum()).reindex(pricing.index).ffill().shift(delay)
    # Fully invested, shorts show up as cash
    pos['cash'] = pos[pos < 0].sum(axis='columns')

    factor_and_returns = al.utils.get_clean_factor_and_forward_returns(
        pos.stack().loc[lambda x: x != 0],
        pricing, periods=(1,), quantiles=quantiles, bins=bins,
        zero_aware=zero_aware)

    return al.performance.factor_returns(factor_and_returns)['1D'], pos
def plot_ic_over_time(factor_data, label='', ax=None):
    mic = al.performance.mean_information_coefficient(factor_data)
    mic.index = mic.index.map(lambda x: int(x[:-1]))
    ax = mic.plot(label=label, ax=ax)
    ax.set(xlabel='Days', ylabel='Mean IC')
    ax.legend()
    ax.axhline(0, ls='--', color='k')

def plot_cum_returns_delay(factor, pricing, delay=range(5), ax=None):
    if ax is None:
        fig, ax = plt.subplots()
    for d in delay:
        portfolio_returns, _ = factor_portfolio_returns(results['alpha'], pricing, delay=d)
        ep.cum_returns(portfolio_returns).plot(ax=ax, label=d)
    ax.legend()
    ax.set(ylabel='Cumulative returns', title='Cumulative returns if factor is delayed')
```

```
def plot_exposures(risk_exposures, ax=None):
    rep = risk_exposures.stack().reset_index()
    rep.columns = ['dt', 'factor', 'exposure']
    sns.boxplot(x='exposure', y='factor', data=rep, orient='h', ax=ax, order=risk_exposures.columns[::-1

def plot_overview_tear_sheet(factor, prices, factor_returns, factor_loadings, periods=range(1, 15)):
    stock_rets = pricing.pct_change()
    stock_rets_specific = compute_specific_returns(stock_rets, factor_returns, factor_loadings)
    cr_specific = ep.cum_returns(stock_rets_specific, starting_value=1)

    factor_data_total = al.utils.get_clean_factor_and_forward_returns(
        factor,
        pricing)

    factor_data_specific = al.utils.get_clean_factor_and_forward_returns(
        factor,
        cr_specific)

    portfolio_returns, portfolio_pos = factor_portfolio_returns(factor, pricing)
    factor_loadings.index = factor_loadings.index.set_names(['dt', 'ticker'])
    portfolio_pos.index = portfolio_pos.index.set_names(['dt'])
    risk_exposures_portfolio, perf_attribution = pf.perf_attrib.perf_attrib(
        portfolio_returns,
        portfolio_pos,
        factor_returns,
        factor_loadings,
        pos_in_dollars=False)
    fig = plt.figure(figsize=(16, 16))
    gs = plt.GridSpec(4, 4)
    ax1 = plt.subplot(gs[0:2, 0:2])
    plot_ic_over_time(factor_data_total, label='Total returns', ax=ax1)

    ax2 = plt.subplot(gs[0:2, 2:4])
    plot_cum_returns_delay(factor, pricing, ax=ax2)
    ax3 = plt.subplot(gs[2:4, 0:2])
    plot_exposures(risk_exposures_portfolio.reindex(columns=perf_attribution.columns),
                   ax=ax3)
    ax4 = plt.subplot(gs[2:4, 2])
    ep.cum_returns_final(perf_attribution).plot.barh(ax=ax4)
    ax4.set(xlabel='Cumulative returns')
    ax5 = plt.subplot(gs[2:4, 3], sharey=ax4)
    perf_attribution.apply(ep.annual_volatility).plot.barh(ax=ax5, color='r')
```

```
    ax5.set(xlabel='Ann. volatility')
    gs.tight_layout(fig)

start = pd.Timestamp("2018-01-01")
end = pd.Timestamp("2018-10-01")
results = run_pipeline(pipe, start_date=start, end_date=end).dropna()
assets = results.index.levels[1]
pricing = get_pricing(assets, start, end + pd.Timedelta(days=30), fields="close_price")
# Load risk factor loadings and returns
factor_loadings = get_factor_loadings(assets, start, end + pd.Timedelta(days=30))
factor_returns = get_factor_returns(start, end + pd.Timedelta(days=30))
plot_overview_tear_sheet(results['alpha'], pricing, factor_returns, factor_loadings)
```

References

Agrawal, S., Azar, P., Lo, A., and Singh, T. (2019), "Practical applications of momentum, mean-reversion, and social media: Evidence from stocktwits and twitter," *Practical Applications*, Institutional Investor Journals Umbrella, 6, 1–4.

Aiello, C. (2017), "Sage ceo hopes experimental depression drug will be as big as prozac," https://www.cnbc.com/2017/12/08/sage-ceo-hopes-experimental-depression-drug-as-big-as-prozac-104888278.html.

Antweiler, W., and Frank, M. (2004), "Is all that talk just noise? The information content of internet stock message boards," *The Journal of finance*, Wiley Online Library, 59, 1259–1294.

Bagnoli, M., Beneish, M., and Watts, S. (1999), "Whisper forecasts of quarterly earnings per share," *Journal of Accounting and Economics*, Elsevier, 28, 27–50.

Cutler, D., Poterba, J., and Summers, L. (1988), "What moves stock prices?" National Bureau of Economic Research.

Das, S., and Chen, M. (2007), "Yahoo! For amazon: Sentiment extraction from small talk on the web," *Management science*, INFORMS, 53, 1375–1388.

Dewally, M. (2003), "Internet investment advice: Investing with a rock of salt," *Financial Analysts Journal*, CFA Institute, 59, 65–77.

Edward, A. (2018), "Syneos health reaches a new high," https://finreviewer.com/2018/12/11/syneos-health-inc-synh-reaches-47-64-formed-h-albireo-pharma-albo-shorts-lowered-by-9-88/.

Fan, J., and Yao, Q. (2015), *The elements of financial econometrics*, Science Press.

Fang, L., and Peress, J. (2009), "Media coverage and the cross-section of stock returns," *The Journal of Finance*, Wiley Online Library, 64, 2023–2052.

Gurdus, E. (2018), "Cramer: Volatility charts suggest now is the time to buy into stocks," *CNBC*, https:

//www.cnbc.com/2018/11/27/cramer-volatility-charts-suggest-now-is-the-time-to-buy-into-stocks.html.

Leinweber, D., and Sisk, J. (2011), "Event driven trading and the'new news'."

Lovelace, B. (2017), "Sage therapeutics shares soar after postpartum depression drug meets main goal," https://www.cnbc.com/2017/11/09/sage-therapeutics-shares-soar-after-postpartum-drug-meets-main-goal.html.

Mitra, L., Mitra, G., and Bartolomeo, D. (2008), "Equity portfolio risk (volatility) estimation using market information and sentiment," *Quantitative Finance*.

Quantopian (2018), https://www.quantopian.com/contest.

"StockTwits" (2018), https://stocktwits.com/.

Tetlock, P. (2007), "Giving content to investor sentiment: The role of media in the stock market," *The Journal of finance*, Wiley Online Library, 62, 1139–1168.

Tumarkin, R., and Whitelaw, R. (2001), "News or noise? Internet postings and stock prices," *Financial Analysts Journal*, CFA Institute, 57, 41–51.

Wysocki, P. (1998), "Cheap talk on the web: The determinants of postings on stock message boards."