An Analysis of WSB “meme” Stocks with Distributed Computing

Steven Taruc, Kenny Lau, Cal Schwefler
CSC 369
Team Diamond Hands

Overview:

Our goal with this project is to use the Twitter and/or Reddit API to scrape comments on these services for discussion on stocks (specifically volatile stocks ‘meme’ stocks such as GME and AMC) with a distributed system applied to handle the load of data.

Introduction:

The motivation for this project was driven by the extreme volatility and potential profits offered by meme stocks such as GME (Gamestop) or AMC (AMC Theatres). Particularly in late January, the stocks were widely discussed on social media platforms such as Reddit and Twitter. At this time both stocks experienced wild periods of growth and contraction bringing them both wide attention. It is widely suspected that these fluctuations are the result of retail traders (smaller traders who alone mean very little, but together have much influence and capital). Due to the influence from retail traders and their presence on social media, we believe that sentiment analysis can be used to analyze the opinions of retail traders on a given stock through social media and this data could potentially be used to predict the next movements of these stocks as well as other stocks with smaller market capitals that gain the attention of retail traders. Figures 1 and 2 display the initial periods of growth for GME and AMC that generate interest in the stock.

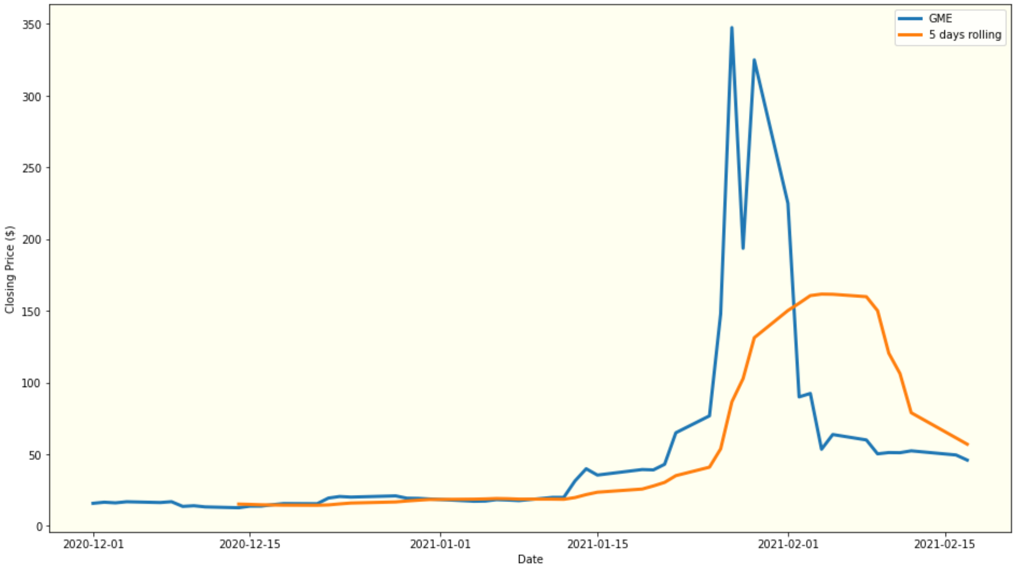


Figure 1: Stock price of GME over time in late January

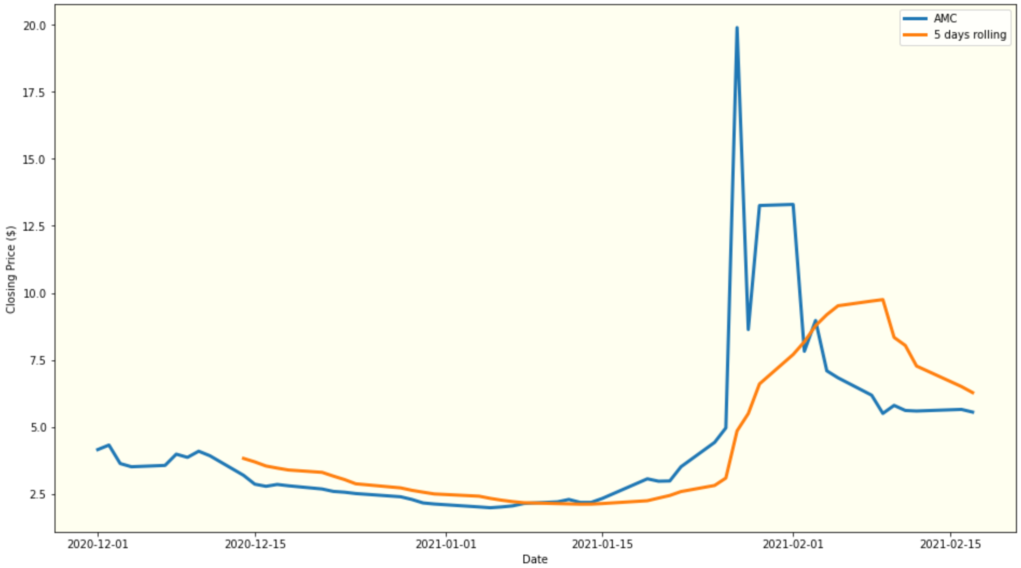


Figure 2: Stock price of AMC over time in late January

To collect data for our analysis, we chose to scrape data from both Reddit and Twitter (popular social media platforms for retail traders) using their respective APIs. Financial data was collected using the popular Yahoo Finance API. We chose to use distributed components for our system due to the large amount of data that must be collected and processed from these social media platforms. The volume of data is often more than a single system could reasonably collect in an efficient or effective manner. As such, distributed computing makes it possible for us to fully collect and analyze the data we gather.

We developed a schedule, as seen in Table 1, to map our progress on this project. We planned to first research and understand the APIs that were available to us for Reddit, Twitter, and Yahoo Finance. Once we understood the tools we had available, the next step was to begin data management and seeing how we could collect data and process it into useful results. Embedded in this step was the beginning of designing a distributed system to better parse and collect the data. With processed data, we would draw results using sentiment analysis.

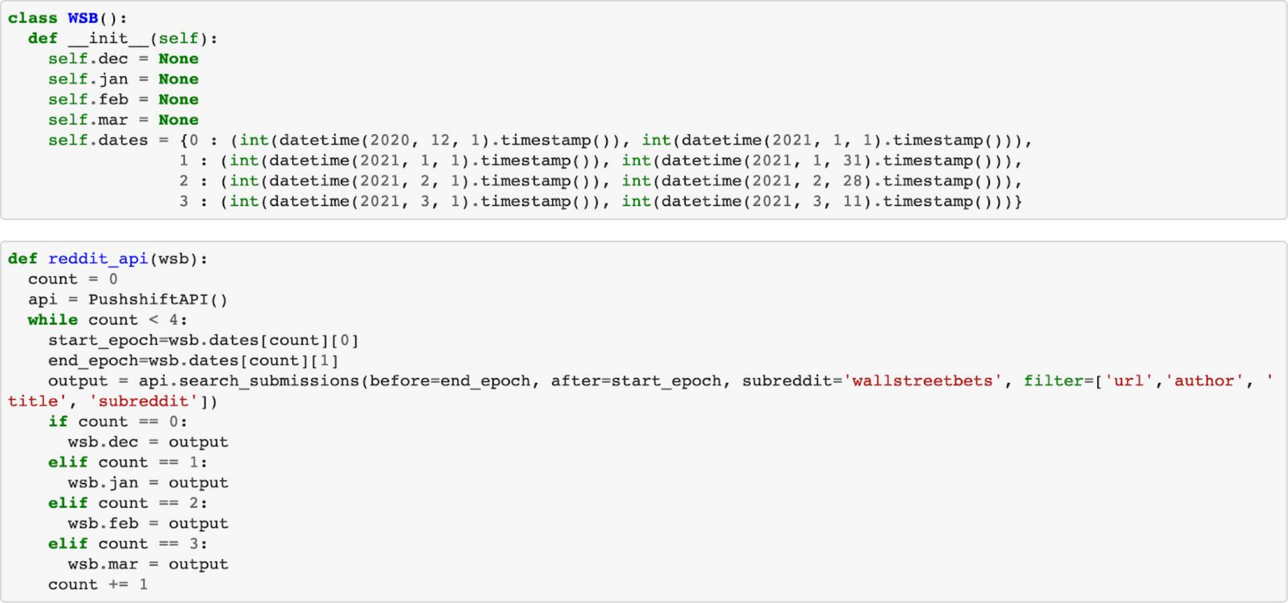
Table 1: Project Schedule

Task Name	W5	W6	W7	W8	W9	W10	Deadline
Initial Research	■	■					2/11
Twitter/Reddit/ Finance API Investigation	■	■					2/11
System Design		■					2/11
Continuous Research		■	■				2/18
Data preprocessing			■				2/18
Product development		■	■	■	■		3/4
Product refinement					■	■	3/11
User Testing						■	3/11
Documentation						■	3/11

Methods:

Reddit Data

The volatility of GME and AMC all started on one platform, Reddit. To get a much better understanding of what was going on, we decided to scrape Reddit posts from the subreddit r/WallStreetBets. Data was scraped using the PushShift API created by Reddit users for developing and gathering data from Reddit. Using this API, we were able to gather data from December 2020 to March 11 2021. There was over 150MB of data that took around 2 hours to get. The posts contained information about the username, title, URL, and date for each post and was received in JSON format. The post data was converted to a Spark dataframe so that it could be cleaned up and processed in a distributed manner which saved us a ton of time. The code can be found in the iPython notebook named “Reddit_WSB.ipynb”. Figure 3 below shows the code for scraping Reddit posts off of r/WallStreetBets.



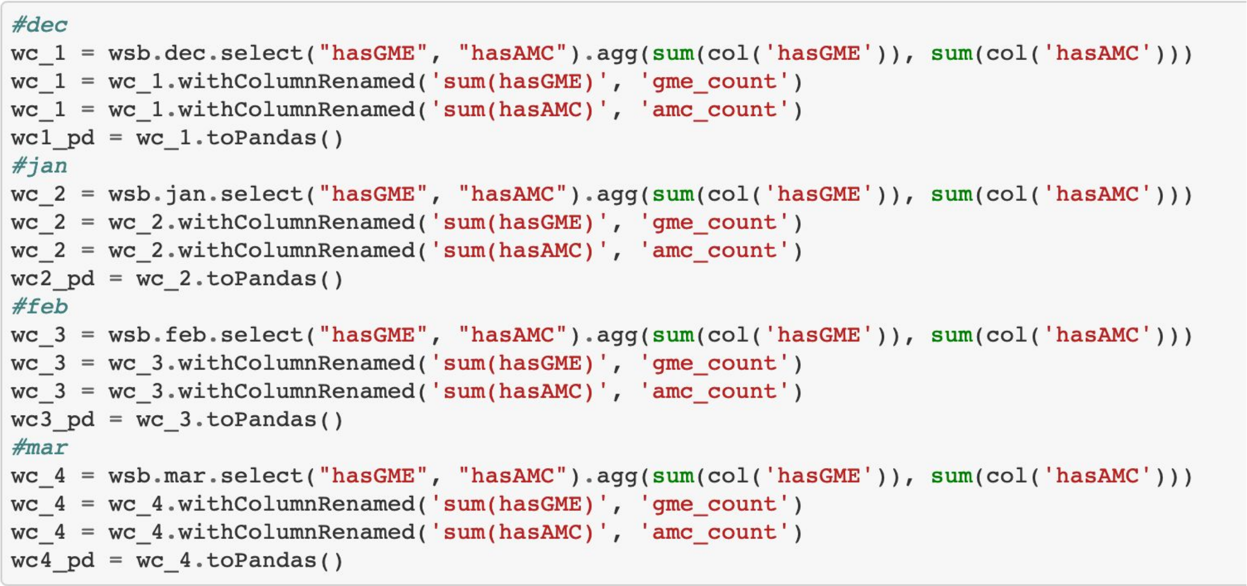
```
class WSB():
  def __init__(self):
    self.dec = None
    self.jan = None
    self.feb = None
    self.mar = None
    self.dates = {0 : (int(datetime(2020, 12, 1).timestamp()), int(datetime(2021, 1, 1).timestamp())),
                  1 : (int(datetime(2021, 1, 1).timestamp()), int(datetime(2021, 1, 31).timestamp())),
                  2 : (int(datetime(2021, 2, 1).timestamp()), int(datetime(2021, 2, 28).timestamp())),
                  3 : (int(datetime(2021, 3, 1).timestamp()), int(datetime(2021, 3, 11).timestamp()))}
```

```
def reddit_api(wsb):
  count = 0
  api = PushshiftAPI()
  while count < 4:
    start_epoch=wsb.dates[count][0]
    end_epoch=wsb.dates[count][1]
    output = api.search_submissions(before=end_epoch, after=start_epoch, subreddit='wallstreetbets', filter=['url','author', '
title', 'subreddit'])
    if count == 0:
      wsb.dec = output
    elif count == 1:
      wsb.jan = output
    elif count == 2:
      wsb.feb = output
    elif count == 3:
      wsb.mar = output
    count += 1
```

Figure 3: Reddit Data Scraping Process

Reddit Analysis and Results

Now that the Reddit data has been collected and formatted nicely into a Spark dataframe, it is time to use the power of Spark aggregation features to extract useful information from the dataframe. First, let us see how popular GME and AMC were by looking at the number of mentions of these stocks over the 4 months.



```
#dec
wc_1 = wsb.dec.select("hasGME", "hasAMC").agg(sum(col('hasGME')), sum(col('hasAMC')))
wc_1 = wc_1.withColumnRenamed('sum(hasGME)', 'gme_count')
wc_1 = wc_1.withColumnRenamed('sum(hasAMC)', 'amc_count')
wc1_pd = wc_1.toPandas()
#jan
wc_2 = wsb.jan.select("hasGME", "hasAMC").agg(sum(col('hasGME')), sum(col('hasAMC')))
wc_2 = wc_2.withColumnRenamed('sum(hasGME)', 'gme_count')
wc_2 = wc_2.withColumnRenamed('sum(hasAMC)', 'amc_count')
wc2_pd = wc_2.toPandas()
#feb
wc_3 = wsb.feb.select("hasGME", "hasAMC").agg(sum(col('hasGME')), sum(col('hasAMC')))
wc_3 = wc_3.withColumnRenamed('sum(hasGME)', 'gme_count')
wc_3 = wc_3.withColumnRenamed('sum(hasAMC)', 'amc_count')
wc3_pd = wc_3.toPandas()
#mar
wc_4 = wsb.mar.select("hasGME", "hasAMC").agg(sum(col('hasGME')), sum(col('hasAMC')))
wc_4 = wc_4.withColumnRenamed('sum(hasGME)', 'gme_count')
wc_4 = wc_4.withColumnRenamed('sum(hasAMC)', 'amc_count')
wc4_pd = wc_4.toPandas()
```

Figure 4: Getting GME and AMC Counts

See in figure 4 above, we were able to use Spark aggregation features and user defined functions called “has_gme” and “has_amc” to create new Spark dataframes that contained the mentions of GME or AMC.

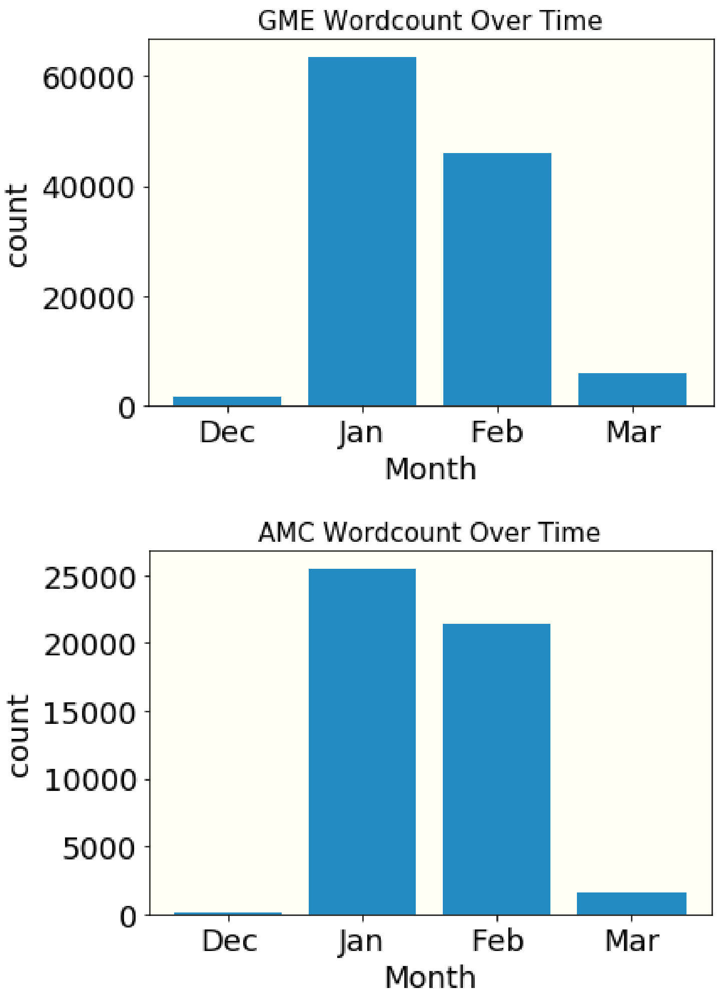
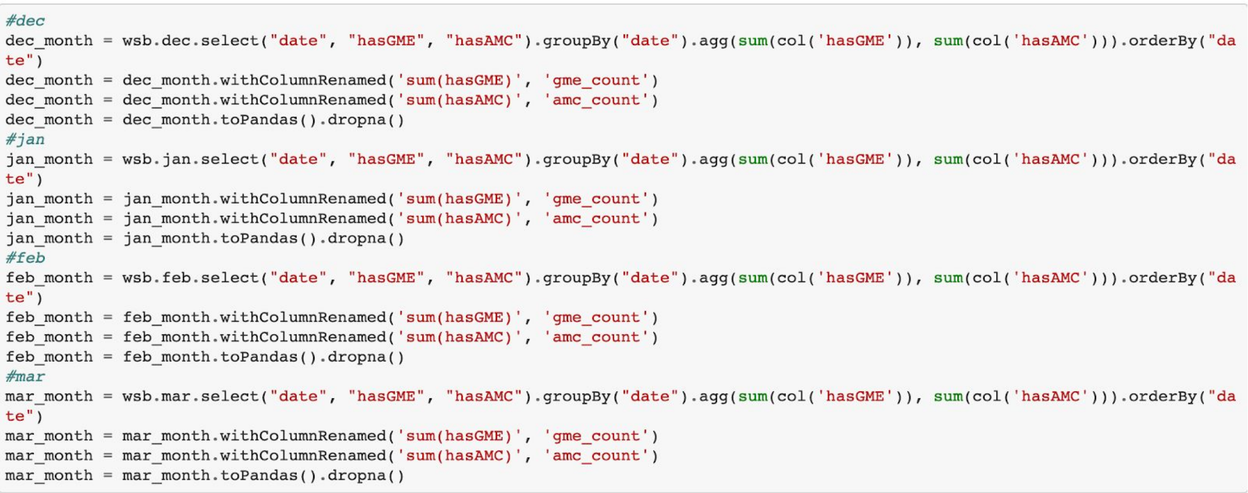


Figure 5: GME and AMC Counts Visualized

Figure 5 above shows the counts of GME and AMC visualized over time. We can see that both GME and AMC stocks follow very similar trends. There was very little discussion in December as things were fairly normal with prices hovering around $20 and $2 respectively. Things start to take off in January and we see that there were over 60,000 posts for GME and over 25,000 posts for AMC. Going into February we see that things start to die down a little but there are still tens of thousands of posts about these stocks at the time.

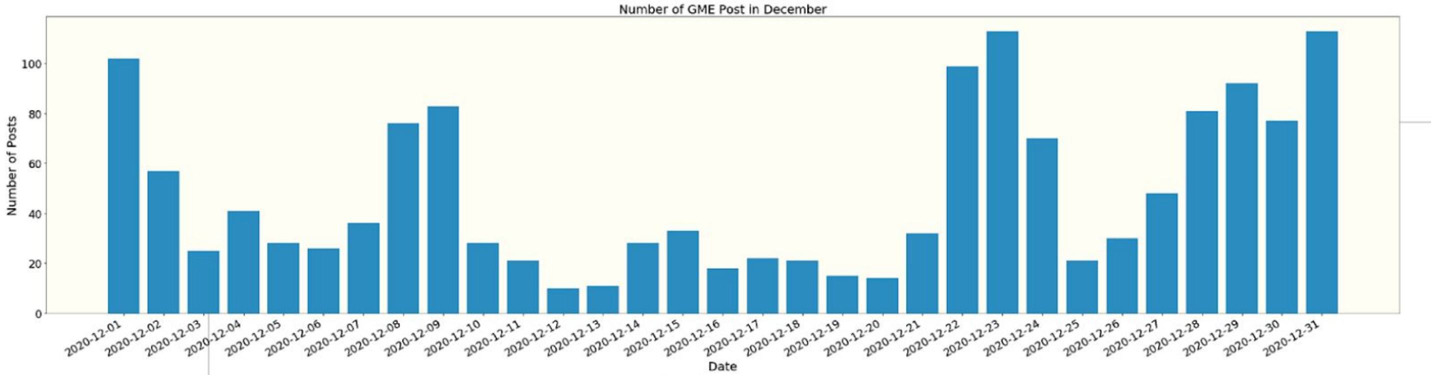
After getting this preliminary data about the stocks, we really wanted to see how active the posts were talking about GME and AMC on a month to month basis.

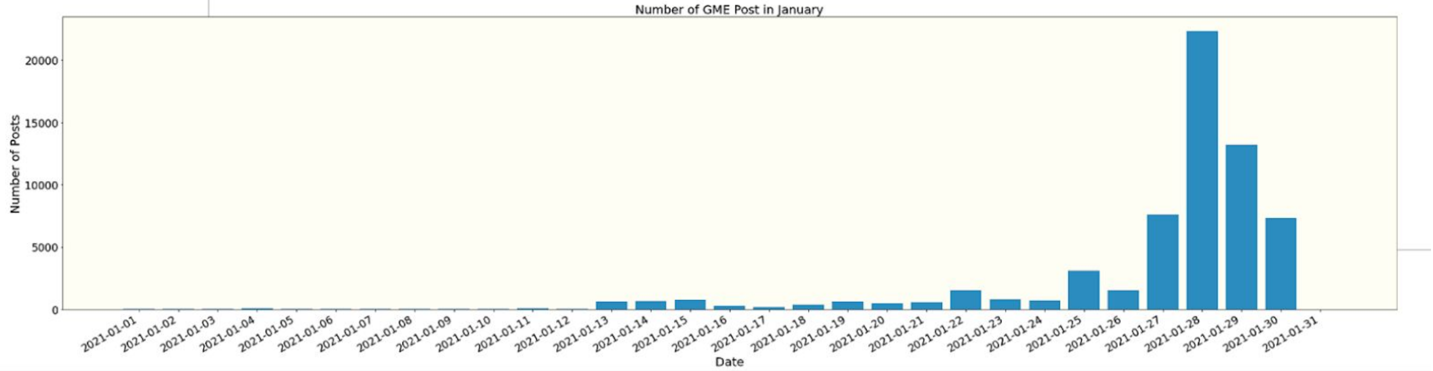


```
#dec
dec_month = wsb.dec.select("date", "hasGME", "hasAMC").groupBy("date").agg(sum(col('hasGME')), sum(col('hasAMC'))).orderBy("da
te")
dec_month = dec_month.withColumnRenamed('sum(hasGME)', 'gme_count')
dec_month = dec_month.withColumnRenamed('sum(hasAMC)', 'amc_count')
dec_month = dec_month.toPandas().dropna()
#jan
jan_month = wsb.jan.select("date", "hasGME", "hasAMC").groupBy("date").agg(sum(col('hasGME')), sum(col('hasAMC'))).orderBy("da
te")
jan_month = jan_month.withColumnRenamed('sum(hasGME)', 'gme_count')
jan_month = jan_month.withColumnRenamed('sum(hasAMC)', 'amc_count')
jan_month = jan_month.toPandas().dropna()
#feb
feb_month = wsb.feb.select("date", "hasGME", "hasAMC").groupBy("date").agg(sum(col('hasGME')), sum(col('hasAMC'))).orderBy("da
te")
feb_month = feb_month.withColumnRenamed('sum(hasGME)', 'gme_count')
feb_month = feb_month.withColumnRenamed('sum(hasAMC)', 'amc_count')
feb_month = feb_month.toPandas().dropna()
#mar
mar_month = wsb.mar.select("date", "hasGME", "hasAMC").groupBy("date").agg(sum(col('hasGME')), sum(col('hasAMC'))).orderBy("da
te")
mar_month = mar_month.withColumnRenamed('sum(hasGME)', 'gme_count')
mar_month = mar_month.withColumnRenamed('sum(hasAMC)', 'amc_count')
mar_month = mar_month.toPandas().dropna()
```

Figure 6: Getting Number of Post Each Day of Month

In figure 6 above, you can see that we aggregated the data using Spark to group by the date and get the count of GME and AMC for each date. This will allow us to see the trends in each month as well as across the 4 months.





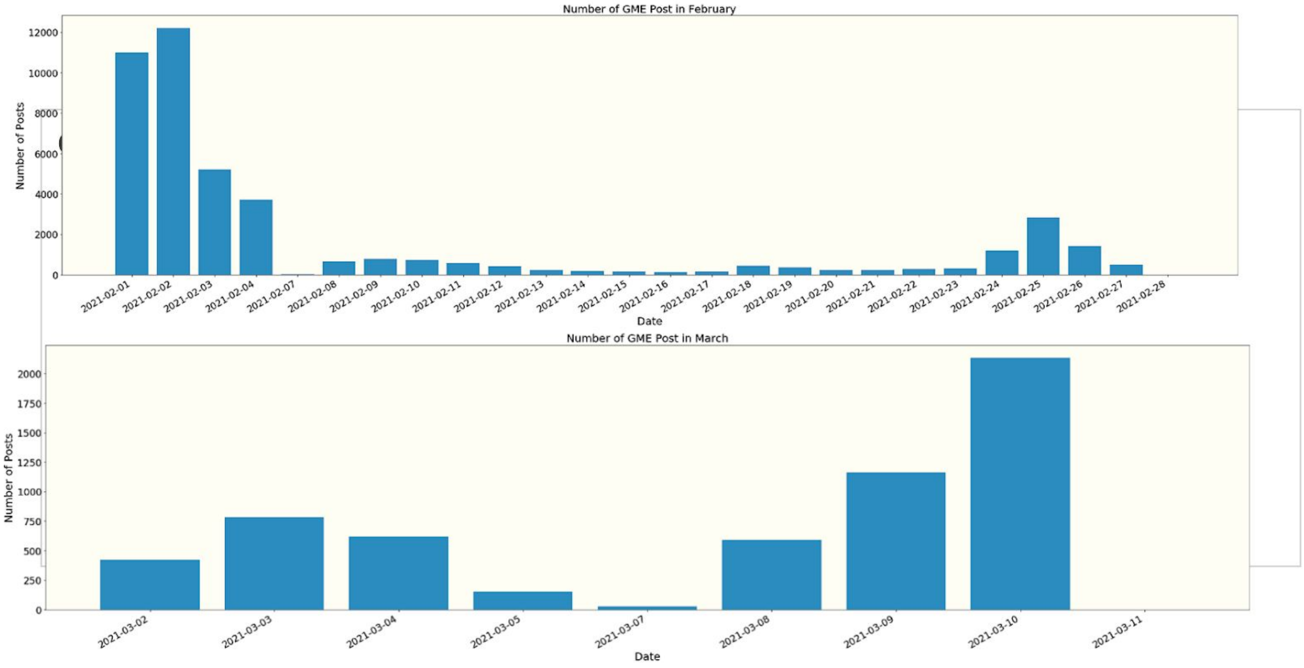
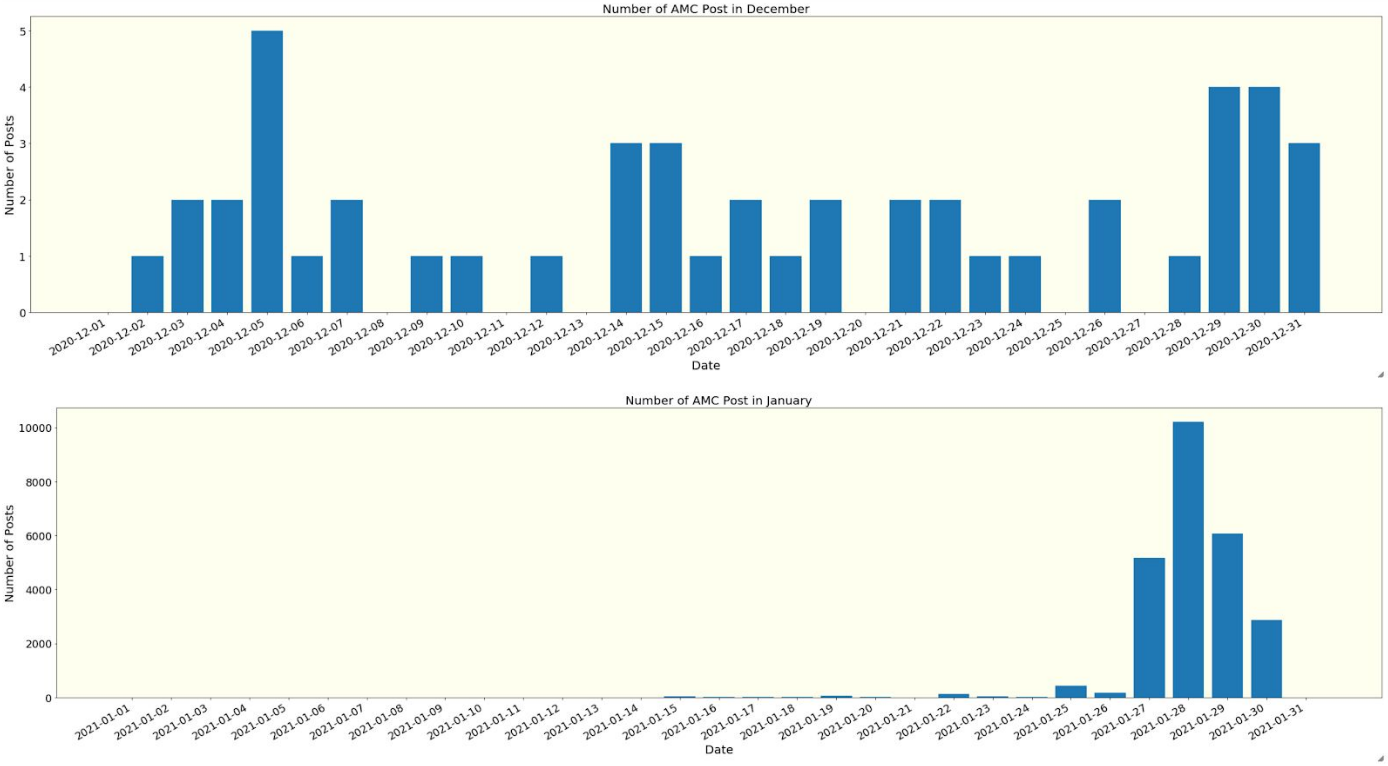
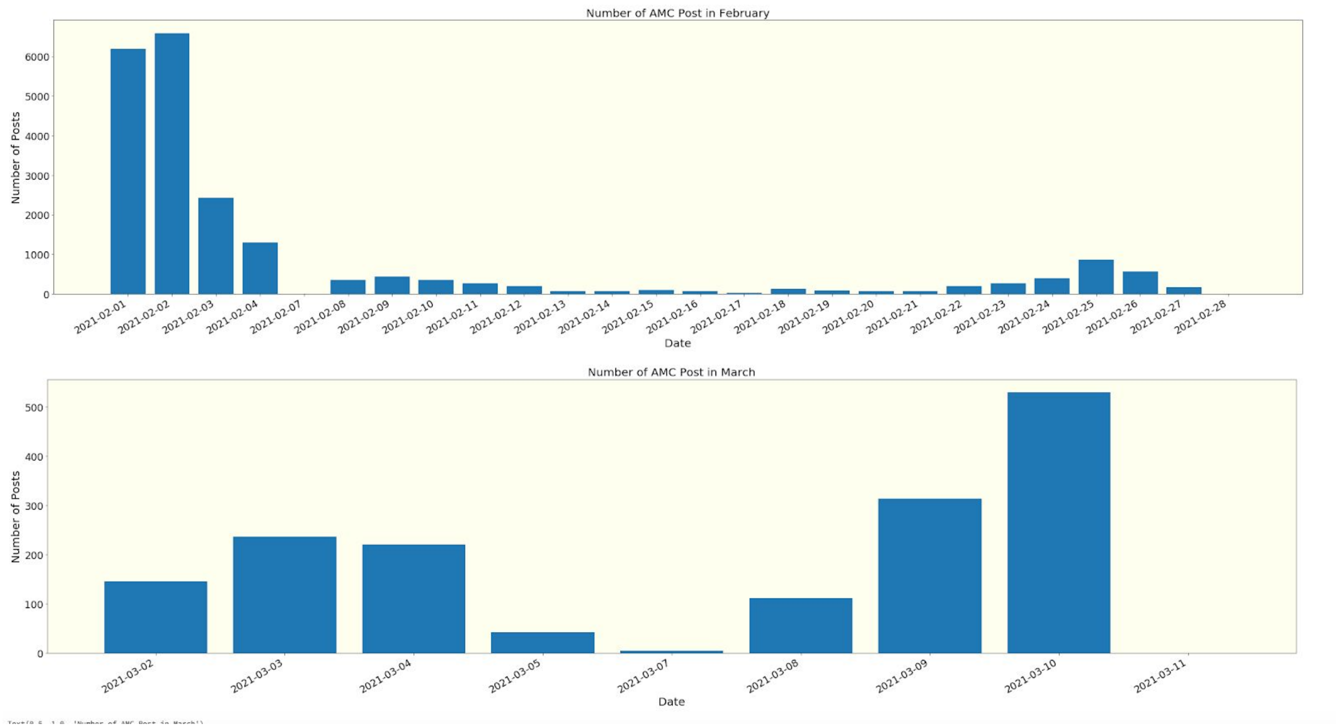


Figure 7: Getting Number of Post Each Day of Month GME Visualized





```
Text(0.5, 1.0, 'Number of AMC Post in March')
```

Figure 8: Getting Number of Post Each Day of Month AMC Visualized

Figure 7 and 8 above shows the number of posts each day of each month visualized for GME and AMC. Once again, in December there was low activity about these stocks when their prices were around $20 and $2 respectively. Things started to take off in late January with over 20,000 and 10,000 mentions for GME and AMC on January 28th, the day after both stocks reached their all time high. Going into February and March posts start to fall off pretty drastically for both stocks as prices no longer have massive fluctuations.

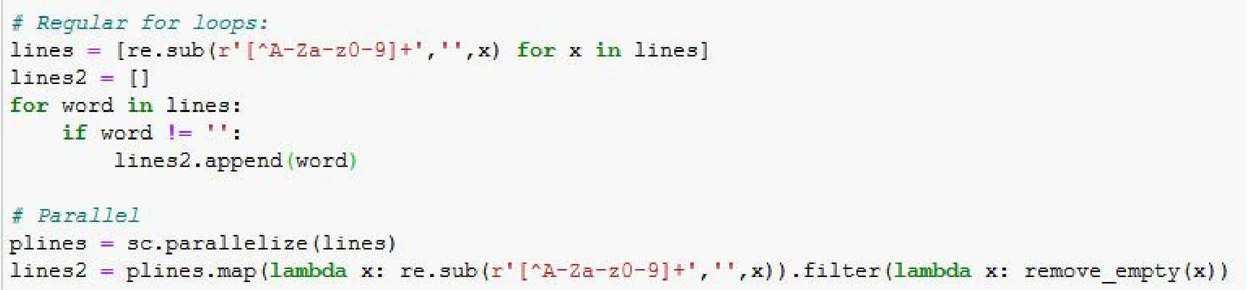
Twitter Data

To understand the engagement of these ‘meme' stocks on one of the most popular social media platforms, Twitter, we use the Twint API to mass gather tweets along with specific features for each tweet. In this case, we used an iPython notebook we named ‘twint.ipynb' to scrape all tweets containing the word ‘GME' for each month from December 1, 2020 to March 11, 2021. We decided to not scrape tweets containing ‘AMC' since we expect the trends for this word to be similar to ‘GME''s trends and the scraping process takes a large amount of space and time. In total, we were able to scrape 13,670 tweets from December, 716,686 from January, 398,966 from February, and 113,704 in March. This adds up to a total of 1,243,026 tweets containing information about username, like count, retweet count, location, language, and other variables in JSON format. These JSON files are then converted to PySpark Relational Data Frames for the distributed ETL process.

Distributed Computing With Twitter Data

To dive deeper into the large number of tweets, we use PySpark SQL functions to select specific queries of these RDD’s. In this case, we focused on extracting the tweet text as strings in parallelized collections. Since these tweets may contain unwanted characters such as “@”’s, punctuation marks, or emojis, we use PySpark’s map and filter functions to apply string manipulation in parallel. Specific manipulation tools we used were RegEx and Natural Language Toolkit functions that helped take out unwanted characters or removed stop words from tweets. This way, we are able to filter words out that do not bring importance to our sentiment analysis. Figure 9 displays the difference between filtering the words through parallel than regular for loops. We then converted these filtered words as a pandas DataFrame so we can easily plot our analysis. Pyspark was also helpful in plotting the trends of tweet count overtime, as we were able to efficiently create aggregate tables organizing tweet count by date.

Figure 9: For loops vs. Pyspark



```
# Regular for loops:
lines = [re.sub(r'[^A-Za-z0-9]+','',x) for x in lines]
lines2 = []
for word in lines:
    if word != '':
        lines2.append(word)

# Parallel
plines = sc.parallelize(lines)
lines2 = plines.map(lambda x: re.sub(r'[^A-Za-z0-9]+','',x)).filter(lambda x: remove_empty(x))
```

The code blocks above achieve the same result but written in two ways: for loops and PySpark. The regular expression and ‘remove empty words’ functions are applied to a set of string lines. Pyspark looks more efficient and cleaner than the large chunk of regular for loops.

Twitter Analysis and Results

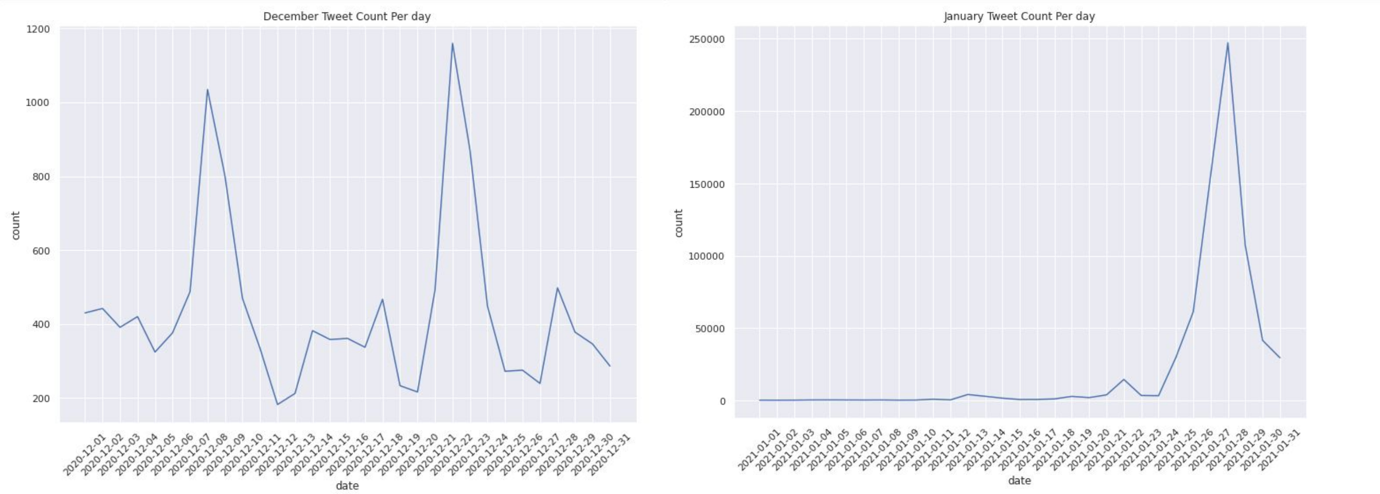


Figure 9: Tweet Count per Day by Month (December 2020 - January 2021)

Seen in figure 9 above, we can see little engagement in the popularity of ‘GME’ during the month of December with around 200-1000 tweets per day. In January, a massive increase of 250,000 tweets arose during the initial GME spike along with the time many platforms enforced temporary buying restrictions, (1/26 - 1/29).

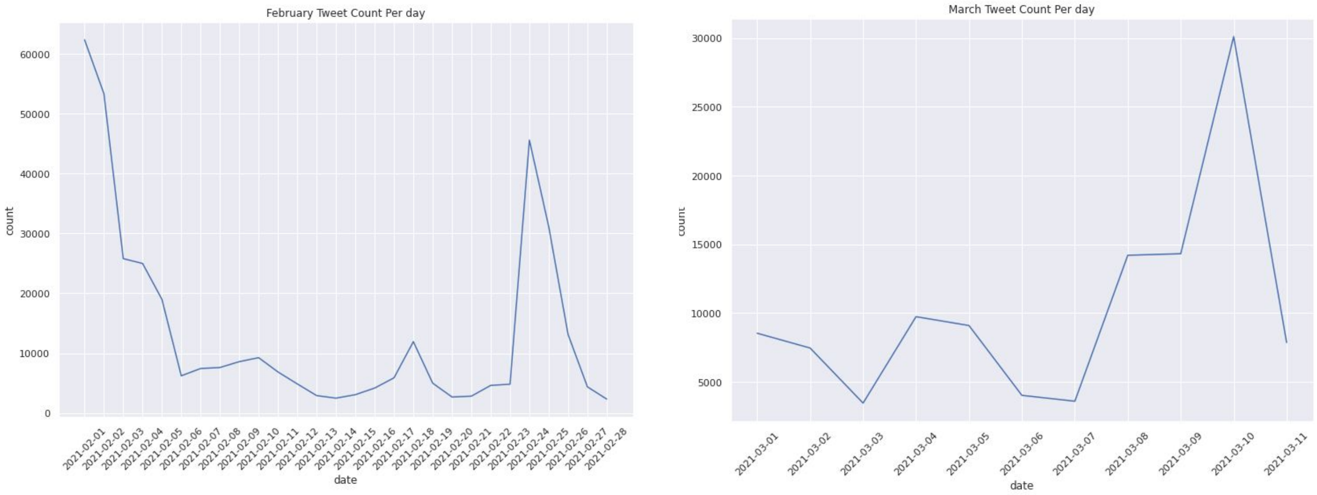
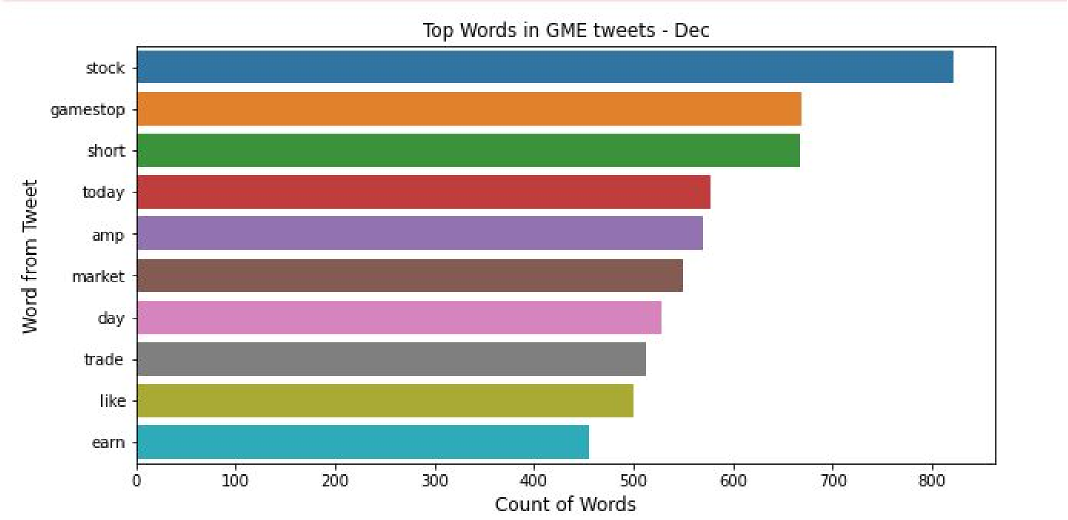
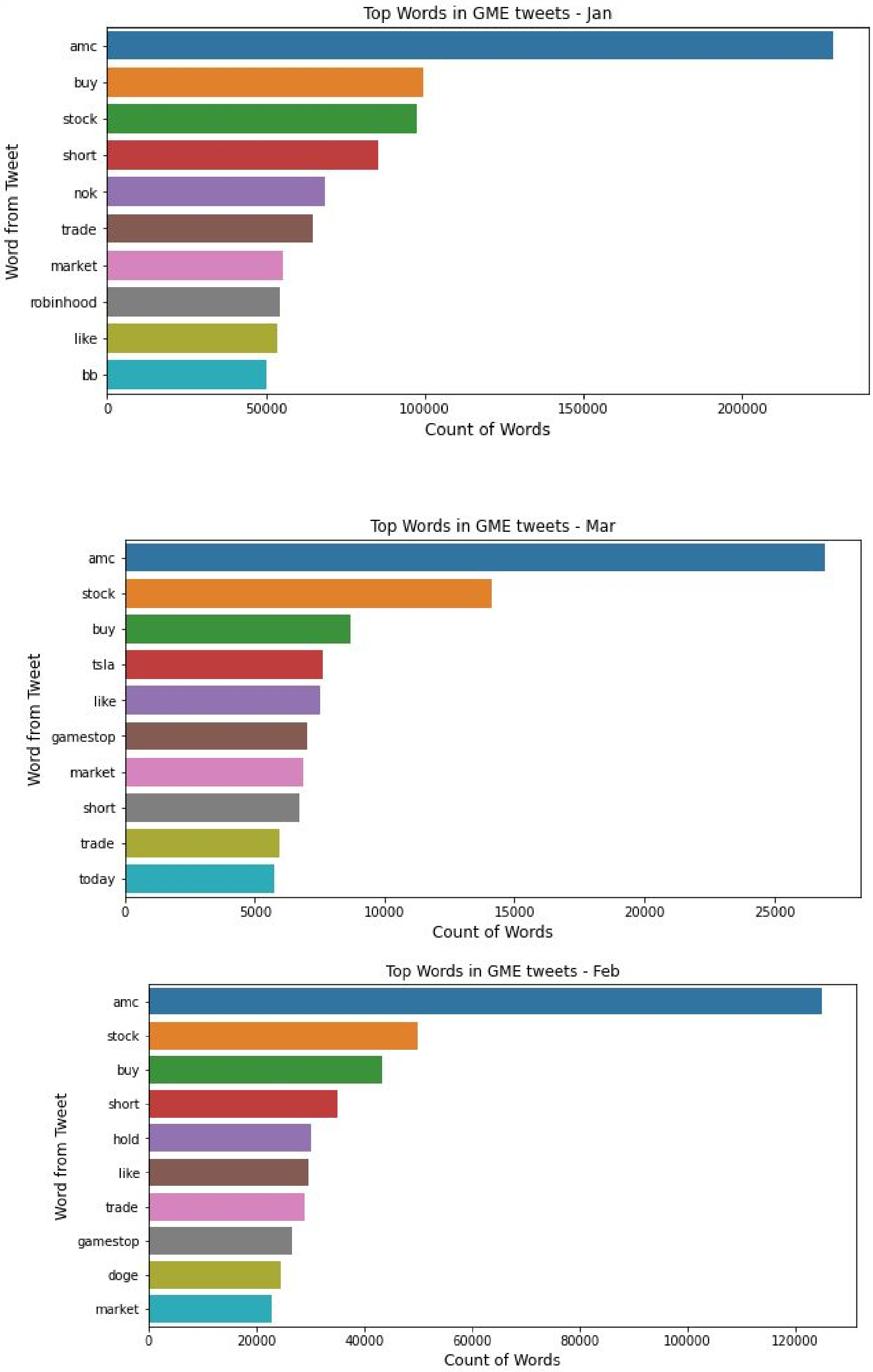


Figure 10: Tweet Count per Day by Month (February - March 2021)

Figure 10 shows that popularity of ‘GME’ in tweets have decreased drastically in the beginning of February but spiked during the congressional hearing for GME on February 18th and the second spike of GME on February 23rd. In March, it appears that the engagement of this stock is steadily rising as of now.





Top Words in GME tweets - Jan

Top Words in GME tweets - Mar

Top Words in GME tweets - Feb

Figure 11: Top Words in GME Tweets by Month

In Figure 11, we discover key words in tweets containing ‘GME’ that may be influenced from or influencing the trends of the stock market. We notice that the words ‘buy’ and ‘hold’ are one of the most words included in these tweets, more than ‘trade’, starting January. In addition, other stocks and cryptocurrency including ‘amc’, ‘tsla’, ‘nok’, ‘bb’, and ‘doge’ are being heavily talked about. These sentiments can attest to the large influence of WallStreetBets on the market decisions of stock traders around the world.

Conclusions/Future Plans:

The biggest lesson learned from this project is that parallel computing plays a key role in processing large amounts of data. Being able to efficiently load and aggregate gigabytes of data proves that tools such as PySpark, Ray, Dask, MongoDB, and others are important in large scale development. In addition, we were astonished by how social media reacted to the recent surge of ‘meme’ stocks. We were aware of the possible trends across Reddit and Twitter, but actually surprised to see the exact numbers to put into perspective. Especially during January and February, talks of ‘GME’ and ‘AMC’ jumped to insane popularity as we can somewhat see in March as well. Also, noticing specific keywords such as ‘buy’, ‘hold’, or mentions of other stock indexes have shown the high impact of this phenomenon.

In the future, we hope to expand this project to its full capabilities if possible. One approach we want to take is streaming the data we collect real-time through PySpark Streaming services. With this, we will be able to do real time analysis and make predictions for stocks on the rise or those based on historical data. We also want to investigate other social media platforms such as news outlets or Instagram, to gather varieties of sentiment for predictions. Finally, we would like to implement machine learning or NLP models to generate complex and accurate analysis.