Twitter Sentiment Analysis For Stock Market Prediction At Scale

Kirby Bloom, Ravi Ayyappan, Nishant Velagapudi, Brandon Cummings
University of California, Berkeley, W251 - Scaling Up, Really Big Data
{kbloom, ravi_ayyappan, nishray, brandon.cummings}@berkeley.edu

Abstract

Twitter provides very unique insights into the general mood and pulse for many different topics, such as politics, sports, or stock market analysis, in real time. In this paper we outline a data collection and analysis architecture for collecting real time twitter sentiment for hashtags related to the stock market topics. This architecture also highlights a daily per minute stock price collection for the Fortune 100. Using a combination of Elasticsearch, Spark, and MongoDB our collection set resulted in ~3,650,000 tweets collected, analyzed, and classified, as well as roughly ~358,000 changes collected for the S&P 100 stock prices (minute granularity). We outline a simple user interface using Kibana and toy example of a RNN to show the use of twitter sentiment data in our case study of S&P 100 companies on the New York Stock Exchange.

1 Introduction

Twitter is a microblogging social media site that allows a user to write up to 280 character microblogs called "tweets". A person signs up for an account and can follow other accounts to see the other account's tweets in their personal twitter newsfeed. This allows a person to follow accounts such as local news outlets to get real time tweets about what is going on in their community, or they can follow celebrities to see what is on the mind of big name artists. There are roughly a billion tweets sent every two days with an average of 6,000 per second **[1]**.

Twitter also has a feature called "hashtags", these are ways that people will tweet about a certain topic, and people can then view other's tweets about the same topic, such as "#business", "#BigData", or "#NYSE". All hashtags are preceded by the octothorpe (aka pound sign) and Twitter uses these hashtags to signal what is "trending" or what has been the most popular topic.

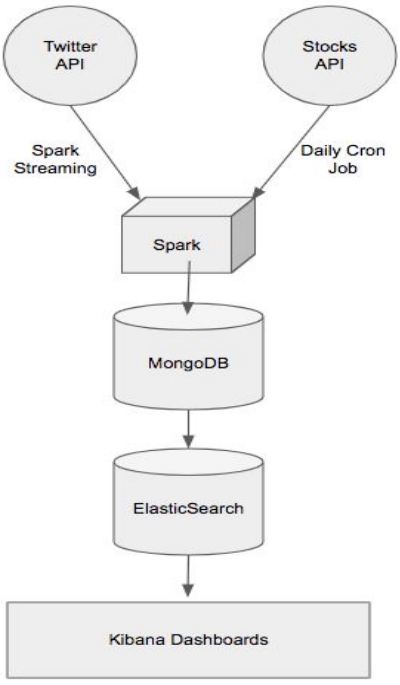
Many of these hashtags revolve around the stock market, where ownership positions in companies are bought and sold in milliseconds by very sophisticated trading algorithms. Most of these trading algorithms take into account tweets from larger companies or entities to help predict where certain stock prices are heading in real time.

The motivation for this paper is to outline a highly scalable data collection architecture to leverage continuous tweet sentiment for ongoing prediction. We will use stock market pricing as a case study, but anticipate that this architecture could be used in examples such as sports betting or election polling.

1.1 Impact of work

This data collection and analysis architecture is extremely practical for wealth management companies, hedge funds, or individual day traders wanting to access real time public sentiment of certain topics on Twitter and tie it back to the Fortune 100 stock minute stock data. We expect that this work will allow visual analysis of the link between sentiment and stock price.

2 Collection architecture



[Figure 1]

2.1 Spark

Spark is a general purpose distributed data processing engine that is suitable for a number of different tasks **[11,12]**. There are four main spark libraries: Spark SQL, Spark Streaming, MLlib, and GraphX. There are 4 common programming languages used with spark: Java, Python, Scala, and R. For our architecture we make use of both the Spark SQL and Spark Streaming libraries, both using Scala. Spark allows the use of massively distributed computing for large dataset problems, its speed comes from doing most of the work in memory on many different worker nodes, all choreographed by a master node. Our cluster contained one master node and three worker nodes.

2.2a Twitter API

There have been many strategies implemented to classify sentiment in tweets, such as parsing for emoticons **[2]** or gathering positive or negative keywords **[5]**, we decided to parse for certain hashtags using similar approaches to Nisar and Yeung **[3]**, and then classifying sentiment using the Stanford CoreNLP library **[4].**

In order to access tweets in real time from Twitter, a streaming API **[9]** is available to those who sign up for a developer account. This streaming API gives access to near real time tweets of around 6,000 tweets per second or a billion every couple of days.

We used a popular library, Twitter4j **[10]**, in combination with Spark Streaming in order to process tweets as fast as Twitter could send them, roughly 6,000 tweets per second **[1]**. In order to filter out tweets that were irrelevant to stock market topics, we chose to only store and analyze tweets that contained one of the following hashtags:

```
["NYSE","NASDAQ","DOW","SP500","USD","WarrenBuffett","IPO","Apple","AAPL","Tesla","Tsla","Facebook","nflx","fb","goog","amzn","qqq","economy","personalfinance","401k","retirement","millennial","debt","Fed","GDP","investing","daytrading","business","finance","equity","wallstreet","trader","financialnews","financialadvice","largecap","smallcap","microcap","OPEC","gold","china","commodities","oil","advisors","wealthmanagement","stockmarket","money","forex","stocks","forextrader","binaryoptions","bitcoin","entrepreneur","trading","investor","forextrading","profit","daytrader","forexsignals","invest", "millionaire" ,"cryptocurrency","investment", "wealth", "stock"]
```

[Figure 2]

Any tweet containing any one of these hashtags went through additional processing to determine the sentiment 1, 2, 3, 4, or 5, 1 being more negative and 5 being more positive. After determining sentiment, this score, the tweet text, the sentiment score, along with most of the tweets metadata (username, time of day, etc...) would be stored in MongoDB (see section 2.4). Here is an example of the final result from parsing for sentiment and storing:

Text:	$AMZN https://t.co/G9v8Q902yP

	given $2,100.00 PT by Wells Fargo & Co. buy rating. https://t.co/WMqBJk9yxB $AMZN #AMZN via @RatingsNetwork
UserScreenName:	AutonomousCNS
Hashtags:	#AMZN
Sentiment:	1
Date:	December 4th 2018, 07:39:28.640
Additional metadata:	...

[Table 1]

2.2b Sentiment analysis

Sentiment was calculated using the StanfordNLP library [18]. The Stanford CoreNLP pipeline allows for simple text processing and annotation. In particular, the sentiment analysis tool attaches a binarized tree of the sentence to the basic data structure of the sentence. The tree nodes are then annotated using a pre-trained recurrent neural network (RNN): subtrees are then scored and those scores are re-aggregated into a document level sentiment. To accommodate minimum 3gb memory requirement of the CoreNLP pipeline, we upgraded each node of the spark cluster to 8GB of memory. We also monitored a tmux session to ensure that there were no timeouts due to the additional processing time required to score sentiment on a per-tweet basis.

2.3 Stock market API and analysis

In order to tie twitter sentiment to stock prices, we chose to collect stock prices on a per minute change for the S&P 100, which are the 102 (because of two of its component companies have two classes of stock) leading US stocks **[13]**. Using an IEXTrading open API **[8]**, we were able to get the per minute changes for each of the following stock symbols:

["AAPL","ABBV","ABT","ACN","AGN","AIG","ALL","AMGN","AMZN","AXP","BA","BAC","BIIB","BK","BKNG","BLK","BMY","BRK.B","C","CAT","CELG","CHTR","CL","CMCSA","COF","COP","COST","CSCO","CVS","CVX","DHR","DIS","DUK","DWDP","EMR","EXC","F","FB","FDX","FOX","FOXA","GD","GE","GILD","GM","GOOG","GOOGL","GS","HAL","HD","HON","IBM","INTC","JNJ","JPM","KHC","KMI","KO","LLY","LMT","LOW","MA","MCD","MDLZ","MDT","MET","MMM","MO","MRK","MS","MSFT","NEE","NFLX","NKE","NVDA","ORCL","OXY","PEP","PFE","PG","PM","PYPL","QCOM","RTN","SBUX","SLB","SO","SPG","T","TGT","TXN","UNH","UNP","UPS","USB","UTX","V","VZ","WBA","WFC","WMT","XOM"]

[Figure 3]

A simple Spark script was setup and submitted to our cluster for work using a cron job worker that would run every day at 5:01pm CT. This job would loop through every ticker symbol from Figure 3 and fetch the per minute changes and then store the results in MongoDB (section 2.4). Here is an example of the stock data stored:

Symbol:	AMZN
Date	12082018
Hour	10
Minute	30
MarketChange	-0.31

[Table 2]

2.4 High volume document store MongoDB

MongoDB is a popular open source document storage NoSql database **[15]**. We decided to use MongoDB because of its flexibility to accept high volumes of records and ease of indexing alongside ElasticSearch, as well as domain expertise on our team.

Our MongoDB setup consisted of two primary document databases, one for storing tweet

sentiment from section 2.2, and the second for storing per minute stock changes from section 2.3.

In order to store the data efficiently, our MongoDB instance was set up as a shard cluster to enable horizontal scaling from the onset. When deciding which type of shard key to use, we chose to use a hash on the unique ids of each document provided from the system of origin. The main driver behind our decision was to avoid creating any hot spots. Understanding that Twitter information can be bursty at times as well as topics, we felt using the hashed id would ensure a flush usage pattern across machines.

2.5 ElasticSearch

Instead of creating our own indexes for performance and frontend interfaces, we decided to use ElasticSearch **[16]** which is an open source search engine based on the Lucene library. ElasticSearch allows for massive text parsing in near real time due to its indexing and searching implementation.

In order to hydrate our ElasticSearch indexes, we chose an open source python tool, the Mongo-Connector **[19],** responsible for porting all documents from our MongoDB instance over to our ElasticSearch instance. The connector tool was built leveraging a plugin architecture that enables developers to build document managers that define a MongoDB source and a destination document structure and technology interface. For our particular setup, we were able to leverage the Elastic2_DocManager out of the box.

The Mongo-Connector was run as a background service that polled our MongoDB metadata to detect additions or deletes made against any of the specified collections. Once these changes were detected, the connector would either send the create an entry in the corresponding ElasticSearch index or remove it.

3 Performance and results

Over roughly a 9 day period, our overall performance resulted in 3,655,277 tweets classified in real time and 358,020 per minute stock changes for the S&P 100 collected on a daily basis around 5:01pm CT.

Our Spark cluster ran into some memory issues at the beginning when running both the twitter parsing and the stocks daily job. This led us to upgrade individual nodes which resulted in a final processing cluster of 3 worker nodes with 2 vCPUs with 8GB of memory and 100GB persistent storage.

Our MongoDB databases will eventually need to increase from their current size of 500GB, seeing as we would reach over a few billion records easily by tweaking some of the filtering, at the current filtering rate they have plenty of room for a few months.

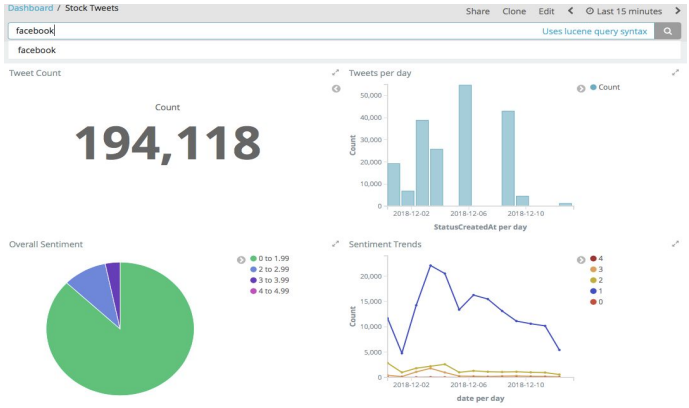
The resulting data has led us to a number of different use cases for this data. Next, we outline an example visualization using Kibana for interactive user interfacing with the data, as well as a continuous training model for stock market change predictions using a Recurrent Neural Network.

4 Analysis & Predictions

4.1 Kibana visualization

ElasticSearch was built alongside the front-end visualization framework Kibana **[17]** and has native integrations for near real time search visualizations. We setup a very simple connector to interface our data collection and allowed for quick parsing for insights.

For example, the below figure depicts the ‘stock tweet’ dashboard that shows the number of tweets received for ‘facebook’ along with the overall sentiment and the sentiment trend for the past couple of days.

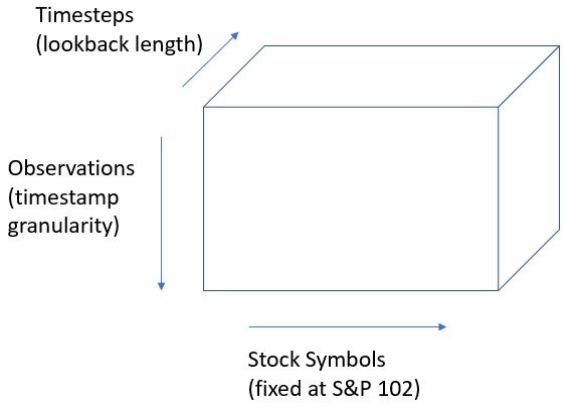


[Figure 4]

Figure 4 shows the overall tweets collected with our stock filter keywords and with the word 'facebook', which resulted in 194,118 tweets over roughly 9 days. The pie chart in the bottom left of Figure 4 shows overall sentiment, green is 1, which is overwhelmingly negative over the period of time collected.

4.2 RNN for deep learning forecasting

As a sample application of the data collected, we restructured the stock ticker prices to a time series format with variable lookback length and rounded times.



[Figure 5]

We then trained a RNN (32 dimension LSTM cells) to predict the next stock price change given past stock prices. Changing the chunk size in the time series yielded different behavior (e.g. stock price delta over 30 minutes is different than stock price delta over 5 minutes). The lookback length (number of timesteps to give the RNN for each prediction) in combination with the chunk size, determined how large the training dataset was. We observed poor predictive performance, with the RNN only able to achieve above baseline RMSE for Amazon stock prices using a 1 and 3 minute window.

	AMZN		FB		AAPL	
Minutes	Avg_shift	RMSE	Avg_shift	RMSE	Avg_shift	RMSE
1	0.0166	0.012	0.0105	0.61	0.0023	0.031
3	0.0494	0.0287	0.0317	0.104	0.00684	0.124
5	0.0827	0.178	0.0525	0.157	0.0122	0.0224
10	0.166	0.491	0.106	0.109	0.022	0.258

The next step involved adding rounded sentiment data before making predictions. We average tweet sentiment for tweets containing either the entire company name or the Fortune 100 symbol (e.g. "Facebook" or "FB"). We impute sentiment for windows of time with no relevant tweets and observe boosted predictive power:

	FB	AMZN	AAPL
Average	0.325	0.352	0.386
RMSE	0.171	0.309	0.783

We are able to beat baseline fluctuation prediction for both Facebook and Amazon stock price predictions at a 30 minute interval. More data would permit the exploration of more complex architectures and longer time series. In the current paradigm, we have to use a large time window to reduce the periods where we have no relevant tweets.

5 Future work

We have built a scalable infrastructure for collecting relevant tweets and calculating sentiment scores. Future would orient around growing the data: this would lead to more valuable insights and better price predictions from the RNN models. Finding a broader list of hashtags related to a company (using product names/common abbreviations) could bolster the sentiment signal. Furthermore, we would also suggest a longer period of data collection.

6 References

1) "Twitter Usage Statistics". Internet Live Stats. http://www.internetlivestats.com/twitter-statistics/
2) Alec Go, Richa Bhayani, Lei Huang. Twitter Sentiment Classification Using Distant Supervision
3) Tahir M. Nisar, Man Yeung. 2017. Twitter as a tool for forecasting stock market movements: A short-window event study.
4) "CoreNLP Tutorials", Stanford CoreNLP. https://stanfordnlp.github.io/CoreNLP/tutorials.html
5) Anshul Mittal, Arpit Goel. 2016. Stock Prediction Using Twitter Sentiment Analysis.
6) "TwitterHashTagJoinSentiments.scala", Github https://github.com/apache/bahir/blob/master/streaming-twitter/examples/src/main/scala/org/apache/spark/examples/streaming/twitter/TwitterHashTagJoinSentiments.scala
7) "SocialSent: Domain-Specific Sentiment Lexicons for Computational Social Science", SocialSent. William Hamilton, Kevin Clark, Jure Leskovec, Dan Jurafsky. 2017. https://nlp.stanford.edu/projects/socialsent/
8) "Charts", IEXTrading. 2018. https://iextrading.com/developer/docs/
9) "Consuming Streaming Data", Twitter. 2018. https://developer.twitter.com/en/docs/tutorials/consuming-streaming-data.html
10) "Twitter4j", twitter4j. http://twitter4j.org/en/
11) "Spark Overview", Apache Spark. 2018. https://spark.apache.org/docs/2.4.0/
12) Carol McDonald. 2018. Spark 101: What is it, What is does, and Why it matters.
13) "S&P 100", spindices. 2018. https://us.spindices.com/indices/equity/sp-100
14) "CronHowTo", Ubuntu. 2018. https://help.ubuntu.com/community/CronHowto
15) "MongoDB", MongoDB. 2018. https://www.mongodb.com/
16) "ElasticSearch", Elastic Co. 2018. https://www.elastic.co/
17) "Kibana", Elastic Co. 2018. https://www.elastic.co/products/kibana
18) Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55-60.
19) "Mongo-Connector", Github. 2018. https://github.com/yougov/mongo-connector