Analyzing the Real-Time Tweets and their impact on Stock Prices Using NiFi, Solr and Spark MLlib

Report Prepared by:

Addit

Contents

Background

Introduction

Market sentiment or 'crowd psychology' determines price movement in financial securities. Apart from financial metrics, such sentiment can be gauged by leveraging social media e.g. customers' tweets. With advent of technology, such tweets are increasing exponentially and to truly extract the power of such data, we require a tool that is capable of analyzing live streaming data. One such tool, Spark Streaming, leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in minibatches and performs RDD transformations on those mini-batches of data. We will utilize Spark Streaming to stream live Twitter data, perform sentiment analysis and uncover statistically significant relationships with stock price fluctuation. We will also leverage Apache NiFi and Solr to create a visualization tool that enable us to analyze and understand the tweets and the theme associated.

Goal of analysis

There are millions of users expressing their opinion on social media. In the age of Internet of Things and Big Data, it is imperative for companies to keep track of the public sentiment on their products, campaigns and offerings, to gain competitive advantage. The goal of this project is to showcase how data from social media can be streamed, structured and analyzed to gain understanding about the tweets being addressed to them in terms of location, users and themes/topics. In our second step, we are exploring avenues to showcase statistical significance between public sentiment on twitter and stock price.

To explore the second section of our project, we chose Target Corporation as our test company and leveraged their new payment app launch for analysis. We wanted to see if there is any statistical correlation between the sentiment of the public addressing Target Corporation and their stock price that is traded on New York Stock Exchange.

Type of data

We utilized data from two data sources:

1. Twitter Live Stream: We obtained data from twitter in two ways:
    a) Using Twitter API through the means of Apache NiFi
    b) Using Python's Tweepy Streaming API

    This data being a live stream, we utilized Spark Streaming to ensure that there is no loss of data. We extracted fields like message text, location, language, username and time of the tweet as and when the data arrived.

2. Stock Market Data: We used Alpha Advantage's Rest API to get minute by minute stock price data. This wasn't a live stream and we utilized python to get updated information every minute

Methodology

Tools Used:

1. Apache Solr:
   a) Overview: Solr is reliable and scalable method of providing distributed indexing and for load-balanced querying. It also has the feature of automated failover and recovery
   b) Use: Solr is a standalone designed for business search server with a REST-like API. To work with it, you must give the documents to it in JSON, XML, CSV or binary format over HTTP. You can then query it via HTTP GET and receive JSON, XML, CSV or binary results
   c) Use in this Project: We provided Solr with Twitter json files which it indexed for use for our other application

2. [1]Apache NiFi:
   a) Overview: It is an easy to use, powerful, and reliable system to process and distribute data
   b) Use: NiFi is mainly designed to automate the flow of data between systems.
   c) Use in this Project: We have utilized NiFi to get twitter data and transform it to further be used by Apache Solr

3. Lucidwork's Banana Dashboard:
   a) Overview: Banana project is a visualization tool that has been designed for optimally dealing with the time series data of Solr. It has been made by forking the Kibana project
   b) Use: Used to create powerful dashboards which can also leverage D3.js for providing more effective visualizations
   c) Use in this Project: We have utilized Banana dashboard in conjunction with Solr to provide a dashboard that people can use for exploration of twitter data

4. Apache Spark:
   a) Overview: Apache Spark is a fast engine that has been designed to for large big-data processing
   b) Use: To interact with the Spark engine, we can write queries in Java, python or Scala and carry out data analysis. Spark's module, Spark MLlib, is designed to run machine learning algorithms. Spark also has spark streaming, that can be used to process streaming data in a fault tolerant way.
   c) Use in this Project: We have utilized both Spark Streaming and Spark MLlib in the project. Spark Streaming has been used to process Twitter data and Spark MLlib has been used for Linear Regression

Descriptive analytics:

We have used two methods of streaming live twitter feed and analyzing it.

Analysis of tweets and theme associated using Apache NiFi, Solr and Banana Dashboard

Before analyzing the sentiment and its impact on stock prices, a company would want to understand the tweets being addressed to them in terms of location, users and themes/topics. Getting such an understanding is also important if a company observes a sudden surge or drop in the sentiment. In such scenarios, the company would like to understand the underlying themes/topics and take appropriate corrective measures. Apache NiFi and Solr allow us to extract tweets and analyze them in an extremely convenient and user-friendly way. Following steps have to be performed in order to implement it:

- Install Hortonworks Data Platform (HDP®) 2.6.3 on Hortonworks Sandbox (link). Once installed on the virtual machine, change the network setting as follows before starting the virtual machine:
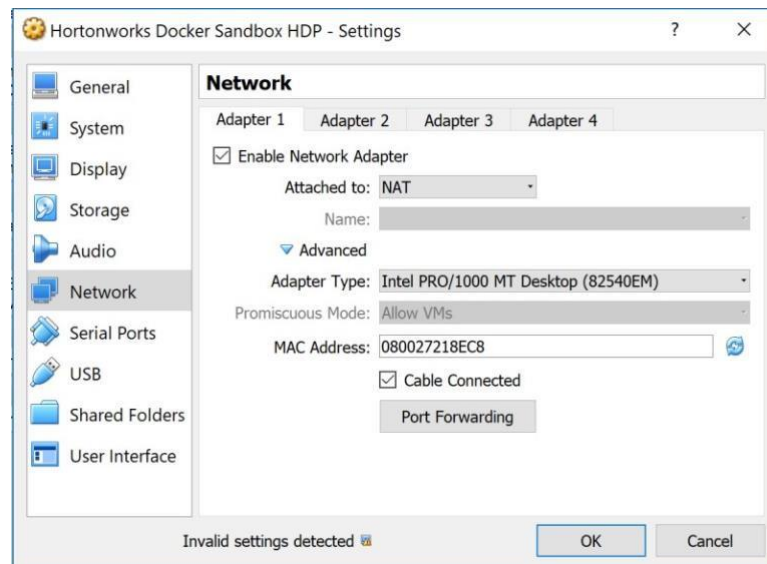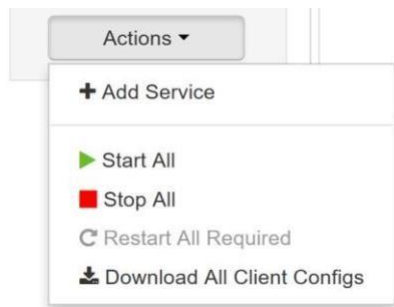

- Once started, open the SSH of the virtual machine using 127.0.0.1:4200/ to reset the Ambari password. Use "ambari-admin-password-reset" and "ambari-agent restart" to reset and restart Ambari.
  Note: Initial username and password are "root" and "hadoop"

```
sandbox login: root
root@sandbox.hortonworks.com's password:
You are required to change your password immediately (root enforced)
Changing password for root.
(current) UNIX password:
New password:
Retype new password:
[root@sandbox-hdp ~]# ambari-admin-password-reset
Please set the password for admin:
Please retype the password for admin:
```

```
[root@sandbox-hdp ~]# ambari-agent restart
Restarting ambari-agent
Verifying Python version compatibility...
Using python  /usr/bin/python
Found ambari-agent PID: 647
Stopping ambari-agent
Removing PID file at /var/run/ambari-agent/ambari-agent.pid
ambari-agent successfully stopped
Verifying Python version compatibility...
Using python  /usr/bin/python
Checking for previously running Ambari Agent...
Checking ambari-common dir...
Starting ambari-agent
Verifying ambari-agent process status...
Ambari Agent successfully started
Agent PID at: /var/run/ambari-agent/ambari-agent.pid
Agent out at: /var/log/ambari-agent/ambari-agent.out
Agent log at: /var/log/ambari-agent/ambari-agent.log
```

- Open Ambari dashboard using 127.0.0.1:8080/ and username: admin and password (set in the previous step). Once opened, "Add Service" to install NiFi.

```
Actions ▼
  ＋ Add Service
  ▶ Start All
  ■ Stop All
  C Restart All Required
  ⬇ Download All Client Configs
```

Select NiFi and install using default configurations. Ignore warnings in the process. Post installation, you would observe NiFi as one the components on the dashboard. You need to restart all the components to ensure they work properly post installation.

- Once all the services/components have been restarted, we need to install Solr. It will be installed using a similar method and with its default configurations.

Configure Solr

o Open your terminal shell and SSH back into the sandbox. Run the following commands as the solr user
(su solr) o Change settings in solr to ensure that a tweet's timestamp format is recognized.

  - *cp -r /opt/lucidworks-hdpsearch/solr/server/solr/configsets/data_driven_schema_configs /opt/lucidworks-hdpsearch/solr/server/solr/configsets/tweet_configs*

  - *vi /opt/lucidworks-hdpsearch/solr/server/solr/configsets/tweet_configs/conf/solrconfig.xml*

  - Once the file is opened in vi type, search for */solr.ParseDateFieldUpdateProcessorFactory* Just below the above, add the following:*<str>EEE MMM d HH:mm:ss Z yyyy</str>*

7

- Download the banana dashboard (which runs on top of Solr) through the following commands:
  - *cd /opt/lucidworks-hdpsearch/solr/server/solr-webapp/webapp/banana/app/dashboards/*
  - *mv default.json default.json.orig*
  - *wget https://raw.githubusercontent.com/abajwa-hw/ambariNiFiservice/master/demofiles/default.json*

- Create a collection in Solr to save the tweets from NiFi using the following command:
  - */opt/lucidworks-hdpsearch/solr/bin/solr create -c tweets -d tweet_configs -s 1 -rf 1 -p 8983*

Import Twitter data flow in NiFi

- NiFi template to extract Twitter data can be downloaded from the following <u>link</u> o
  Open NiFi using 127.0.0.1:9090/NiFi and upload the downloaded .xml template
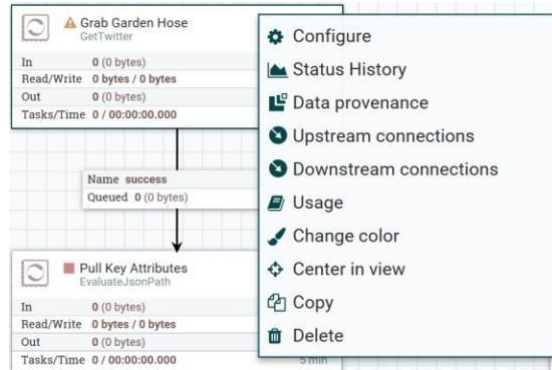

Instantiate the Template
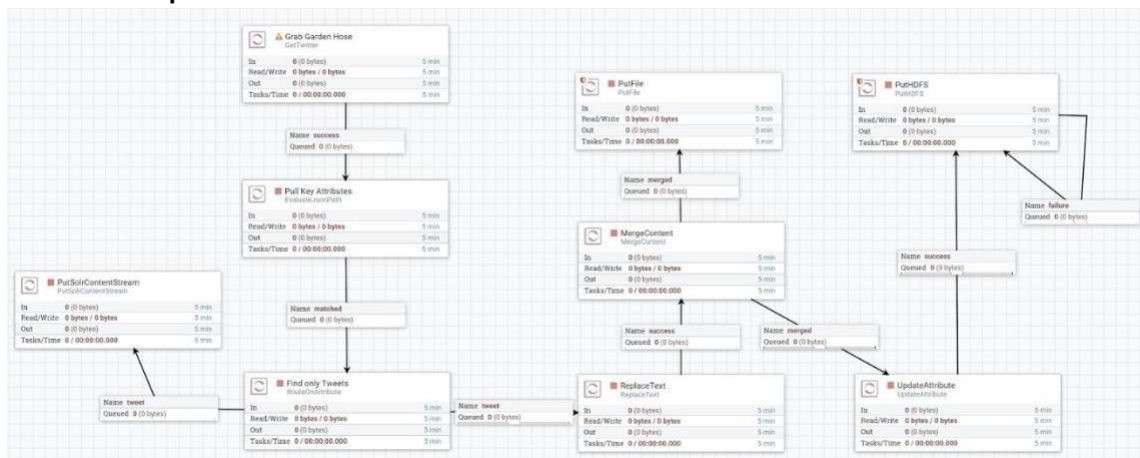
Extract Tweets o Configure the Grab Garden House

module:

- Enter the following information: Consumer Key, Consumer Secret, Access Token and Access Token Secret along with the Terms to Filter on.

o Start the entire process.

- **Open the Banana Dashboard on Solr:** The dashboard can be accessed on
  http://127.0.0.1:8983/solr/banana/index.html#/dashboard

FINAL DASHBOARD

Analyzing the impact of sentiment on stock price using Spark Streaming and Spark MLlib:

For the second part of the project, we wanted to understand the impact of social media sentiment over stock price, comparisons were made for minute-by-minute stock prices and the sentiments of live tweets for the corresponding minutes.

For gathering the stock prices, Alpha Vantage REST API can be utilized to gather one new stock price each minute. To obtain the stock price data, follow these steps:
1. Request a link to connect to the stock API through the alpha vantage URL
2. Then write a python code to set this process for getting stock data at a consistent time window, by having a loop with a sleep time of 60 seconds for each cycle. Set the loop to run for the number of minutes desired.
3. Load in the data from the URL as a JSON file and transform it into a pandas dataframe.

4. Format the dataframe to be easily merged with the tweet data.

To gather the live tweets, a python API called Tweepy creates a client listening to live tweets which are ingested and processed by Spark Streaming and then saved into a persistent Hive table. This process requires two separate files: one to stream and read the tweets from twitter, and the other to call the read file and process the tweets.

The following steps accomplish building the read file:
1. First, it is required to set up a twitter account to acquire a consumer key, consumer secret, access token, and access secret which will be used to initialize the twitter connection.
2. Create a file called TweetRead that will connect with twitter and stream tweets once called. This tweetRead file will listen to the port set up on the local machine and wait for direction.
3. Specify a filter for the tweets
4. The tweets will be encoded into messages to both print and send to the host client.

The following steps accomplish setting up the file to process the tweets:
1. First, it is necessary to ensure both the spark and SQL contexts are initiated before streaming can begin.
2. Next, A global SQL context should be instantiated to ensure the reading of the streaming data and the writing of the tweets into a table are through the same SQL context. If not, the data will not be written to a table.
3. The next step sets up the socket stream to connect with the TweetRead file. Ensure that the machine and port used here, and the port listed in the TweetRead file are identical.
4. Next, set the streaming window to 20 seconds. This window measures the speed to which the port is collecting and processing the tweets, creating the size of the batch to be processed.
5. Then create a function which converts the time of the tweet into the eastern time zone to be consistent with the stocks, since the stock prices are gathered from the NYSE.
6. Thus far, the above functions have not been called. The following code will call these functions for each batch of tweets.

```
( lines.map( lambda word: ( word.lower(), 1 ) )
    .map( lambda r: Row(tweet=r[0], timeTweet = get_time_zone() ))
    .foreachRDD(lambda rdd: rdd.toDF().write.mode("append").saveAsTable("tweets"))
)

ssc.start()
```

7. When the start is called, each tweet will then be passed through the initialized port in an rdd which will be broken into Dstreams and passed to the above functions. Once the start is initialized, the above lines will continue to run in the background and cannot be modified.
8. We choose the second authentication method because service accounts and their corresponding access to different Google Cloud services (including Google Translate) can be

easily centrally managed and the keys are more secured and convenient to use in programming.

9. After the tweets are transformed, they are appended to a hive table.

10. Tweets about multi-international companies or other global wide topics are always coming in all kinds of languages. While English tweets are normally dominant in number, other languages such as Spanish and French still make substantial proportions. Therefore, it would be wasteful to simply discard non-English tweets, let alone the sentiment result would be biased as well.

11. To continuously translate large number of words coming at a high speed, web based services are perfect for the job and Google Translate API as an established service becomes our choice.

12. Now let us simply walk you through the implementation.

13. Translation of Non- English Tweets: After inserting the data in hive table, the non-English tweets were converted into English by the following procedure:

    a) Google Cloud account creation and Translation API activation are straightforward. After that, we need to choose the authentication method to access the service. Google provides two authentication methods:

        1. API Key: encode the key in the request URL;

        2. b) Service Account: when service accounts are created by project manager, keys are automatically generated as json files and can be saved onto local disk to be used by Google Cloud client library as default.

    b) We choose the second authentication method because service accounts and their corresponding access to different Google Cloud services (including Google Translate) can be easily centrally managed and the keys are more secured and convenient to use in programming.

    c) By defining the target language as English and feeding the text to be translated, Google API will return the result in a nicely formatted json object. Just pay attention to make sure that text to be translated are decoded into Unicode before sent to the Google Translate.

    d) In our case, the function was embedded into another tweet sentiment calculation function which was then applied over the tweets data frame. To avoid requests exceeding limit (100,000 characters per 100 second per user), we can put our function to a very short sleep after each execution.

14. Sentiment Calculation: The sentiments can be calculated using a bag of words method. A list of positive and negative words will be used to compare the words contained in the tweets. a) The sentiment function will be called, which inherently calls the translation in every tweet and outputs a pandas dataframe.

    b) Next the dataframe must be transformed such that each minute of tweets is contained in one row in order to visualize the minute by minute tweet sentiment.

    c) Using matplotlib animation, create a live animation to see the sentiment changing over time. The following is a frame of the animation created. The blue represents the positive sentiment and the orange represent the negative sentiment. Each frame shows the proportion of positive and negative words for each tweet in the corresponding minute.

Sentiment by Minute

Blue = Positive
Orange = Negative
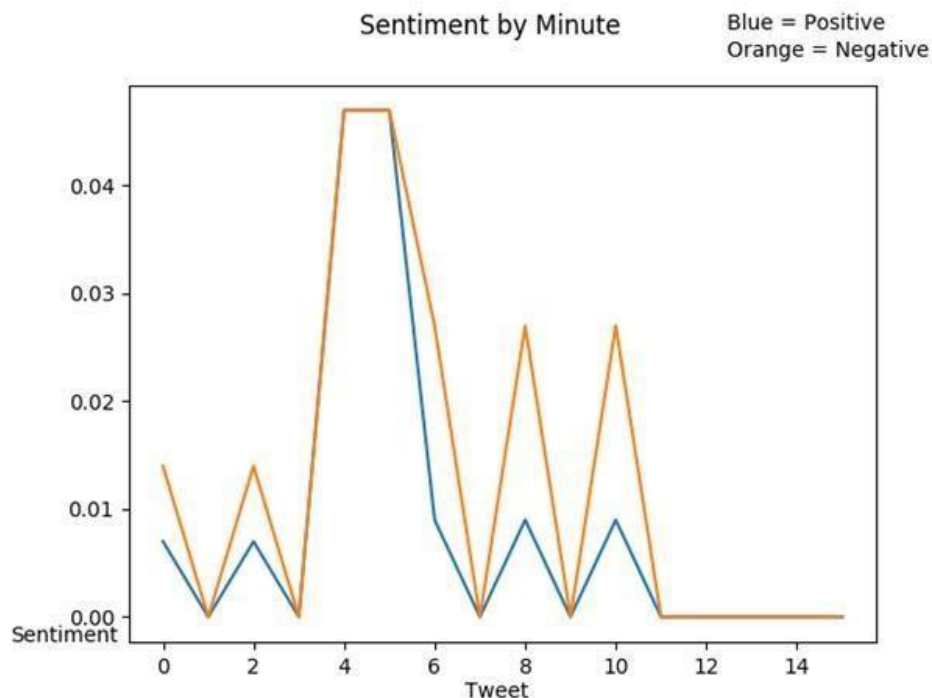
Using SparkMllib for finding relationship between stock price and sentiment:

We then utilized SparkMLlib to combine the minute by minute positive sentiment, negative sentiment and stock price data and did a linear regression to find if there is a relation between stock price and the sentiment of the people.

For doing the above process,

1. We first combined the sentiment and stock price data in python.
2. Next, we converted this pandas dataframe to a spark dataframe
3. Then using Vector Assembler, we transformed the dataframe to a labelled point vector, keeping the label as the stock closing price and positive – negative sentiment as the features
4. Next, we used LinearRegression from the pyspark.ml.regression library to run a linear regression with Stock Price being the dependent variable and Positive and Negative sentiment being the independent variables
5. Using model.summary we found out that the p values for both positive and negative sentiment were not sigNiFicant and the model explained only 18% of the variation in stock price with respect to sentiment

Conclusion

This use case demonstrates a practical method to quantify and utilize live streaming consumer opinions. In general, a live Banana dashboard such as this can be used to assess immediate reactions to a new product, a particular event, a marketing campaign, etc.   And to measure whether the product or

campaign had an impact, we can utilize sentiment analysis and spark MLlib to identify any relation. To further enhance this project, we could expand on the live streaming by including the sentiment on the banana dashboard.

References

Alpha Vantage API Documentation. (2017). Retrieved December 07, 2017, from https://www.alphavantage.co/documentation/

Analyzing Social Media and Customer Sentiment With Apache NiFi and HDP Search. (n.d.). Retrieved December 05, 2017, from https://hortonworks.com/tutorial/analyzing-social-media-and-customersentiment-withapache-NiFi-and-hdp-search/

Apache NiFi Team. (n.d.). Apache NiFi Overview. Retrieved December 07, 2017, from https://NiFi.apache.org/docs/NiFi-docs/html/overview.html

Authenticating to the Cloud Translation API | Translation API | Google Cloud Platform. (n.d.). Retrieved December 07, 2017, from https://cloud.google.com/translate/docs/auth

Cloud Translation API Client Libraries | Translation API | Google Cloud Platform. (n.d.). Retrieved December 07, 2017, from https://cloud.google.com/translate/docs/reference/libraries#clientlibrariesusage-python

Pradhan, M. (2016, May 25). Twitter Trends Analysis using Spark Streaming. Retrieved December 07, 2017, from http://www.awesomestats.in/spark-twitter-stream/