Social Media Analysis using Apache Spark and Kafka

Eshna Sengupta

May 12, 2019

1 Introduction

Social media has become an increasingly important tool for companies. Customers dissatisfied with products are quick to turn to social media platforms like Twitter, Facebook, or Instagram to vent and complain about products and services. If such posts go viral, it could affect a company's reputation. Analyzing such posts in real-time, and taking necessary actions is crucial for damage control. This paper discusses a real-time social media tracking system built using python, Apache Spark and Apache Kafka that performs sentiment analysis to identify negative posts and provide a real-time summary of findings. Using these tools, data from twitter that relates to the topic of interest will be extracted, stored and analyzed.

This paper is divided into four sections. Section 1 provides discussion of technologies/tools that have been used in the project. Section 2 contains implementation details, and Section 3 presents the results, Section 4 discusses Related/Future work.

2 Background

2.1 Twitter API

Twitter provides an API (application programming interface) that allows developers to interact with twitter data and gather information about tweets such as the content of the tweet, the twitter handle of the user, the number of likes and retweets, etc. When someone wants to access Twitter's APIs, they are required to first apply for a developer account and then register an application. By default, applications can only access public information on Twitter[1]. Python provides a useful open source library called *tweepy*[2] that makes it easy to communicate with the Twitter platform through its API. For this project, python will be used to interact with the Twiiter API and make requests. The results obtained from this request are in JSON format, which can be easily parsed to extract useful information.

[1]https://help.twitter.com/en/rules-and-policies/twitter-api

[2]https://www.tweepy.org

2.2 Apache Kafka

Apache Kafka was developed by LinkedIn for stream-processing and building real-time data pipelines. It is essentially a distributed publish and subscribe messaging system. Kafka servers/nodes accept and store incoming messages from data sources/publishers which are subsequently stored by Kafka into relevant topic groups. The consumers of these messages, also known as subscribers, express an interest with Kafka about one or more topics. The data stored in a topic is shared with the topic's subscribers. Kafka has five core APIs: Producer API, Consumer API, Stream API, Connect API and AdminClient API. [1] The first three APIs will be utilized for this project:

- *Producer API* lets an application can publish streams of data/records to Kafka topics.
- *Consumer API* allows an application to read streams of data/records from subscribed Kafka topics.
- *Streams API* works as a stream processor. It consumes streams of data from one or more Kafka topics, and produces an output stream to output topics.

Python provides a library, *kafka-python*, that allows easy access to these APIs.[4]

2.3 VADER Sentiment Analysis Tool

Sentiment analysis is the process of assigning a sentiment (positive, negative, neutral) to a piece of text. It is also known as opinion mining, and uses a combination of Natural Language Processing and Machine Learning techniques. Sentiment analysis can be done in two ways: Using a lexical approach, or using machine learning. In a Lexical approach, a dictionary or lexicon of sentiments, is used to map words words to sentiments. Sentiments can be represented as a categorical variable (Neutral, Negative, Positive) or a numerical variable (scores, range of sentiments). The advantage of a lexical approach is that, unlike a machine learning approach, there no need to train a model using labeled data. VADER(Valence Aware Dictionary for Sentiment Reasoning) is a rule-based sentiment analysis tool, that incorporates the lexical method and is sensitive to both polarity (positive/negative) and intensity (strength) of emotion. [5]

Performing sentiment analysis on social media content has it's challenges. Consider tweets for example, they include not just words, but emoticons like ':)' , acronyms like 'LOL' or even slang words. VADER is especially good at handling such scenarios. In [5] the authors mention that VADER achieves a very high classification accuracy and even outperforms human raters when analyzing tweets.

2.4 Apache SparkSQL

Apache Spark is an open source processing engine that makes it easy to perform low latency processing for large amounts of data. It is also able to ingest data from Apache Kafka. Spark is designed to be highly accessible and provides support and APIs for a variety of programming languages like Python, Scala and R. Spark SQL[3] is Sparks' interface for working with structured and semi structured data and is useful for querying big data. It can be used both within a Spark program, and also through external tools that connect to it through standard database connectors (JDBC/ ODBC), such as business intelligence tools like Tableau.

3 Implementation

The implementation is divided into the following tasks:

- Obtain twitter credentials
- Create a kafka producer that publishes messages to a topic
- Create a kafka consumer that consumes messages from the kafka topic.
- Process content of the tweets and store it to SQLite database.
- Use Spark SQL to query data from SQLite database and create visualizations.

3.1 Set up

This project was built using:
Python 3
Kafka version 2.2.0 (Released March 22, 2019)
Spark 2.4.1 built for Hadoop 2.7.3 (Released March 31, 2019)
Anaconda version 2019.03, build channel: Python 3.7

3.2 Twitter credentials

The following twitter credentials are required to access twitter data through an API: *Consumer API key, Consumer secret API key, Access Token, Access Token secret*

As of 2019, Twitter changed the rules regarding the use of it's APIs in order to protect the privacy of it's users. With the new rules in place, one must first apply for a developer account with Twitter detailing how they plan on using their API. Once the developer account is approved, an application must be created at `https://apps.twitter.com/app/new` and the above information is issued.

OAuth or Open Authorization is commonly used for token-based authentication and authorization on the Internet. Through OAuth an end user's account information can be used by third-party services without exposing the user's password. The python *tweepy* library supports oauth authorization through the tweepy.AuthHandler class. It is used in this project.

[3]https://spark.apache.org/sql/

3.3 Kafka Producer

Pykafka is a kafka client for python that implements kafka producers and consumers in python. Given that a kafka instance is running on a localhost, pykafka can be used to connect to it. The cluster that pykafka connects to must have a topic defined on it in order for pykafka to create a producer and start producing messages.

Twitter has two types of APIs : Streaming API and REST API. In contrast to the REST API, the streaming API returns a continuous stream of responses.

For this project, an instance of tweepy.Stream is used to establish a streaming session and send messages to StreamListener instance. The *on_data* method of StreamListener receives all messages and it is within this method that we create the kafka producer. Our kafka producer publishes all tweets about the company "Tesla" to the topic "twitter".

3.4 Kafka Consumer and Tweet Processing

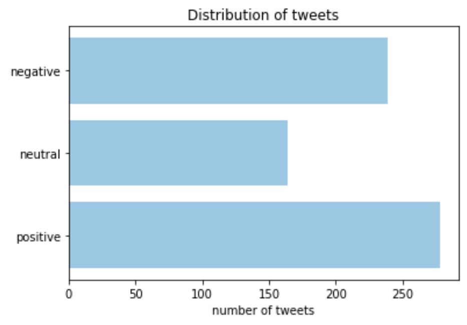
After we have our kafka producer running, we are ready to start consuming messages. Again, this is done using pykafka, through the SimpleConsumer consumer instance. The messages are Tweet objects rendered in JSON. A tweet object has several attributes. Some of them are: **id, created_at, text**.[3] provides detailed information about the Tweet Object. Since our main aim is to perform sentiment analysis, we clean the text portion of the tweet to get better results from our analyzer. Tweets posted by a user can either be an original post or a retweet of a post by another user. In the latter case, the tweet object text field starts with a "RT". This is of no use in sentiment analysis and can be stripped out. We do the same for any hashtags or twitter handles included in the text. The cleaned text is then supplied to our Vader Sentiment Analyzer, which returns a sentiment (Positive, Negative, Neutral) for the tweet. A SQLite database is populated with tweet attributes: *username, followerCount, originalTweet, cleanTweet, and Sentiment.*

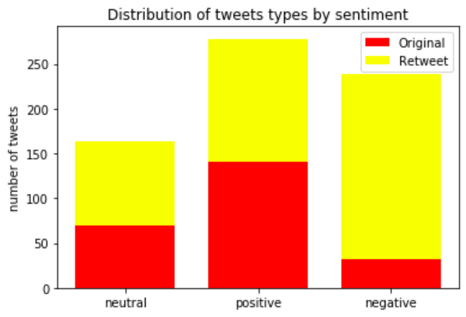
3.5 Querying SQLite Database using SparkSQL

Spark SQL can operate on a variety of data sources such as Parquet, ORC, JSON and AVRO Files, Hive tables. However, to read data from a SQLite database, Spark SQL must connect to it using JDBC (Java Database Connectivity).

4 Results

Once the data is populated in the SQLite database, to gather meaningful insights, Spark SQL and the python graphing package *matplotlib* is used. The data exploration is done on a jupyter notebook running on Spark. We notice that Most of the tweets are positive, however there are also a large number of negative tweets. On further investigation, it is found that majority of the tweets are actually retweets and not original tweets. It is highly likely that a popular twitter user posted a negative tweet and several of his followers retweeted it.





Popular users with large follower counts on social media accounts are considered “influencers”, and companies usually approach them for promotions and advertisements. Negative comments from influencers should not be taken lightly. So from our data, we also want to find names of popular users who post negative tweets. The most retweeted negative tweet is also useful to know. In our case, the most retweeted negative tweet is **RT @Scotty49er: So today I drove a Tesla for the first time, and now all other cars are ruined for me. Our current gas car now feels like a...**, which was tweeted 168 times. It is interesting to note that this tweet is actually not negative. Our sentiment analyzer was not able to classify it correctly. It is might have been confused due to words like “ruined”, which has a negative connotation.

5 Related Work

5.1 Spark Streaming and Storm

Apache Spark Streaming is an extension of the Spark Core API that allows the processing of real-time streaming data in a scalable way. In the future, this application can be extended to process the streams of messages on the go using spark streaming. For example, we could compute a real time count of the sentiments. Spark Streaming has several advantages: [2]:

- **Scalability:** Spark Streaming is able to scale to hundreds of nodes easily.
- **Ease of Use:** Using Apache Spark is easy because it allows the user to write streaming jobs the same way batch jobs are written.
- **Fault Tolerance:** high availability is maintained by efficiently recovering from failures.
- **Spark Integration:** Since it is built on spark, Spark Streaming lets users reuse the same code for batch processing, join streams against historical data, or run ad-hoc queries on stream state.

Another stream processing technology that we could consider is Apache Storm. Storm is a popular choice for real-time analytics. It processes incoming messages one by one, and provides minimum latency. It has good parallel processing capabilites and provides high throughput. In addition to that it is scalable. However, Spark provides better programming language support and integration than Storm and would therefore be more suitable for this use case.

References

[1] Apache kafka documentation. "`https://kafka.apache.org/documentation/#adminapi`".

[2] Apache spark streaming. "`https://spark.apache.org/streaming/`".

[3] Tweet object - twitter developers. `https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object.html`.

[4] Steven Van Dorpe. Kafka-python explained in 10 lines of code. "`https://towardsdatascience.com/kafka-python-explained-in-10-lines-of-code-800e3e07dad1`", Aug 2018.

[5] C. Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. 2014.