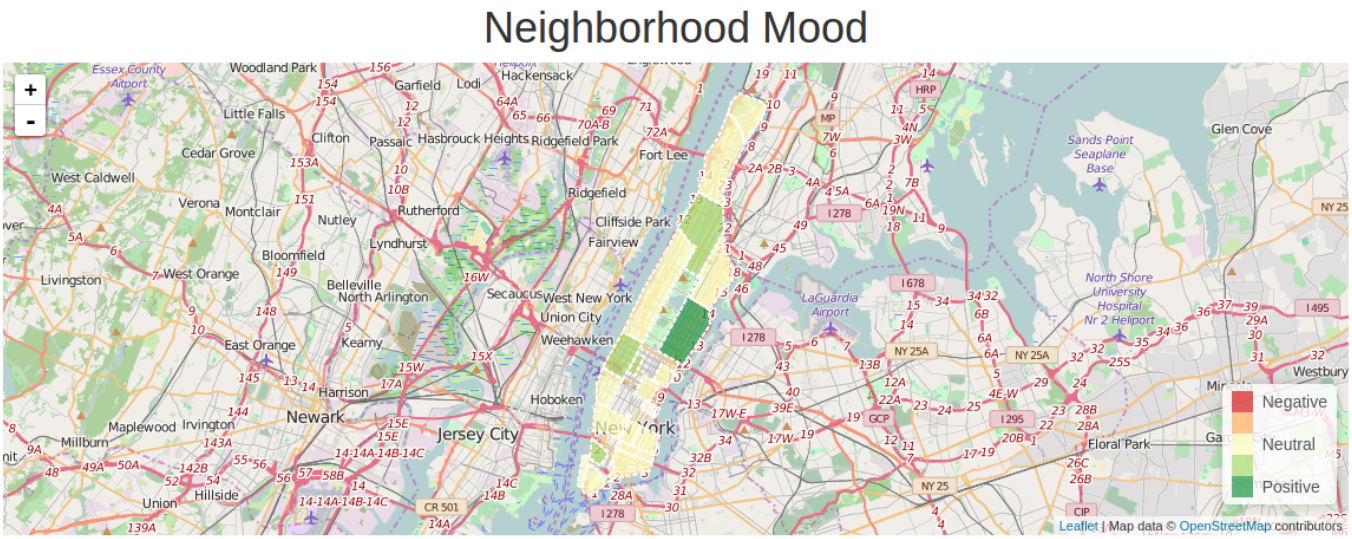
Goodbye Spark Streaming, Hello AWS Kinesis Analytics

Mark Jarrell Follow
Aug 29, 2016 · 3 min read



We will create a dashboard that prototypes a real-time, big data app using AWS.

AWS Kinesis Analytics allows us to use SQL in place of more complex offerings such as Apache Storm and Apache Spark Streaming.

I made this app to learn new techniques and am hoping others can find it useful.

Architecture

> *twitter streaming api [provides raw tweets] ->*
> *raw_tweets_to_kinesis [continuously running python program] ->*
> *tweets [kinesis stream] ->*
> *lambda_handler [lambda function] ->*
> *processed_tweets [kinesis stream] ->*
> *kinesis analytics [kinesis analytics continuously running sql] ->*

> *sentiment_by_neighborhood [kinesis stream] ->*
> *send_results_to_dynamodb [lambda function] ->*
> *neighborhood_sentiments [dynamodb] ->*
> *dashboard [html/js/flask app]*

The **raw_tweets_to_kinesis** Python application listens in on a long lasting HTTP stream. It receives all tweets originating from NYC in real time and pushes them to the **tweets** Kinesis Stream.

Kinesis Streams are conceptually pub sub message queues. They can handle a lot of data at low latency which makes them great for big data.

The **lambda_handler** lambda function is a Python application listening to the **tweets** Kinesis Stream. When **tweets** receives new messages, **lambda_handler** (a) calculates the tweets' sentiments using TextBlob. Sentiment is a number representing the mood of the tweet ranging from -1 (negative) to 1 (positive). **Lambda_handler** also (b) figures out which NYC neighborhoods the tweets belong to. It does this using a KML file containing the neighborhoods' GPS coordinates. This is conceptually a list of coordinates that creates a polygon representing the neighborhood. **Lambda_handler** outputs the sentiments and neighborhoods for the tweets as JSON messages to another Kinesis Stream called **processed_tweets**.

> ***Author's Side Node***

> *Determining which neighborhood a tweet belonged to took some trial and error.*

> *First I tried using a Python library called Shapely, but it was difficult to install on AWS Lambda due it its C dependency.*

> *I then tried SymPy, a library written in 100% Python. It proved to be slow and kept timing out.*

> *A function found online after Googling 'point in polygon python' did the trick.*

Kinesis Analytics is an AWS product that allows SQL statements to be run on live data sources (Kinesis Streams in our case). Kinesis Analytics can eliminate the need for Storm, Spark Streaming, etc for plenty of use cases. Our Kinesis Analytics SQL statement runs every time a new tweet comes into **processed_tweets**. It then looks at all tweets pushed to **processed_tweets** in the last 30 minutes, groups by neighborhood, and averages sentiment. Any changed [neighborhood, average_sentiment] values are pushed to another Kinesis Stream called **sentiment_by_neighborhood**.

A lambda function called **send_results_to_dynamodb** listens in on **sentiment_by_neighborhood**. It updates or inserts a row into our DynamoDB called **neighborhood_sentiments** where the key is the neighborhood. This means that there will only be one "row" per neighborhood in the database.

Now we have a DynamoDB database with two "columns", neighborhood and sentiment. The database is always kept up to date thanks to our lambda architecture.

Our dashboard simply polls this database every 30 seconds and fills in the neighborhood colors based on the sentiment values. Our dashboard uses Leaflet and OpenStreetMap for the map.

Code

Demo

https://neighborhood-sentiment.herokuapp.com/

Credits

This tutorial is based on https://blogs.aws.amazon.com/bigdata/post/Tx1Z6IF7NA8ELQ9/Building-a-Near-Real-Time-Discovery-Platform-with-AWS. This tutorial also utilizes AWS Kinesis Analytics.

The html/javascript/flask dashboard is inspired by Leaflet's Choropleth tutorial: http://leafletjs.com/examples/choropleth.html.

The dashboard was created from https://github.com/zachwill/flask_heroku for easy deployment to Heroku.

The point in polygon code comes from http://www.ariel.com.au/a/python-point-int-poly.html.

About Me

GitHub: https://github.com/jarrellmark

LinkedIn: https://www.linkedin.com/in/mark-jarrell-1bb52924