DATA VISUALIZATION WEBSITE

REPORT

-Composed By-

Kazi Nawasis Ahmed

Advanced Web Development with Big data

Middlesex University

Department of Computer Science

OBJECTIVE:

The objective for this project was to create a web interface that visualizes numerical data gathered from 3rd party web services and using machine learning to generate prediction about the data. Another target was to receive twitter data regarding the numerical data and using AWS comprehend algorithm to perform the sentiment analysis which will be also visualized in the web interface. The idea of the project was to facilitate aspiring student, traders and researchers to learn more about the cryptocurrency's trends and patterns and future prices.

INTRODUCTION:

Web Design:

The design is created in a way to have a dynamic connection between server and clients. The homepage includes five different cryptocurrencies to choose from (the chosen crypto currency for this project is BTC, ETH, REQ, BNB, REQ). When the client connects to the server, it receives a response from the server and the data (which consists of numerical data and results of sentiment analysis of the twitter data that was gathered) about a default currency is visualized. The numerical data is visualized using 'candlestick chart' and the sentiment data is visualized using 'pie chart'. Thereafter, a client can choose any other currency using the buttons displayed and the relevant data will be regarding the currency gets displayed.

The design also facilitates to update any of the data dynamically whenever server responds with new numerical or sentiment data. The plots for both get updated in real time without the requirement of client participation or activity.

Server Design and architecture:

The serverless architecture is built to automatically send data when a client connects. Client's id is stored in a database to maintain connection and send data back to client. Send a client disconnects the clients id is removed from the database. The architecture also supports the functionality to send a response to

all connected clients when the database for either numerical or sentiment data gets updated in the table.

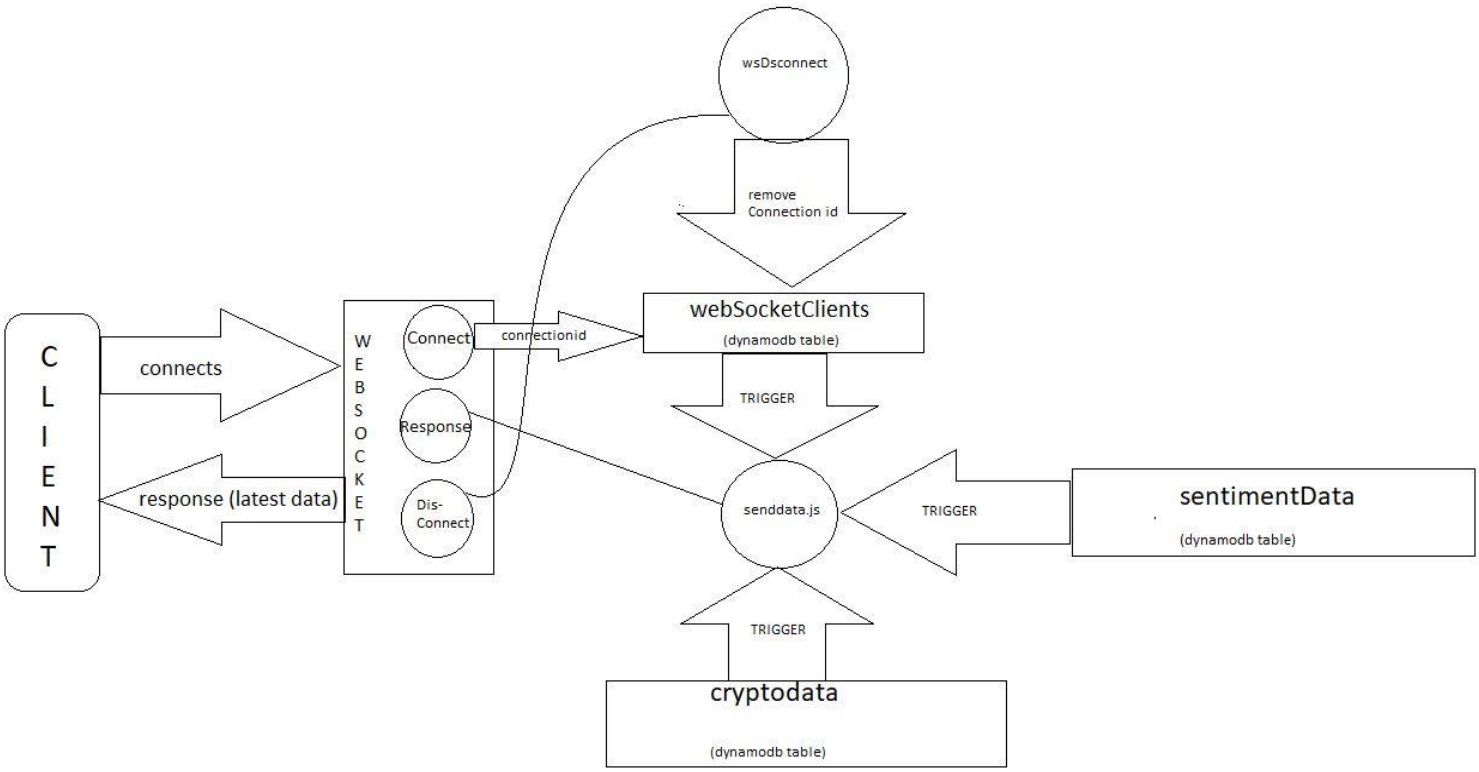


Figure. 1

Fig. 1 showing the working of the serverless architecture. When a client is connected a connection id gets stashed away in a table, that insert event triggers a function that collects latest data from the database and wraps it in an object and sends it back to the client. This response is used in client-side to visualize the data sent back from the server. In case of drop of connection or disconnection from client the connection id gets removed from the table. While

the client is still connected a to the websocket, a function is hooked with different triggers which are connected to the table containing numerical data and another table that holds the result of the sentiment analysis of the text data. When there are any changes made to either one of the data, this triggers the function which checks if new data being added to the table, in which case the function retrieves the latest data and sends a response to all the connected clients (whose id's are still stashed away in the table). And the clients get updated with the latest data automatically.

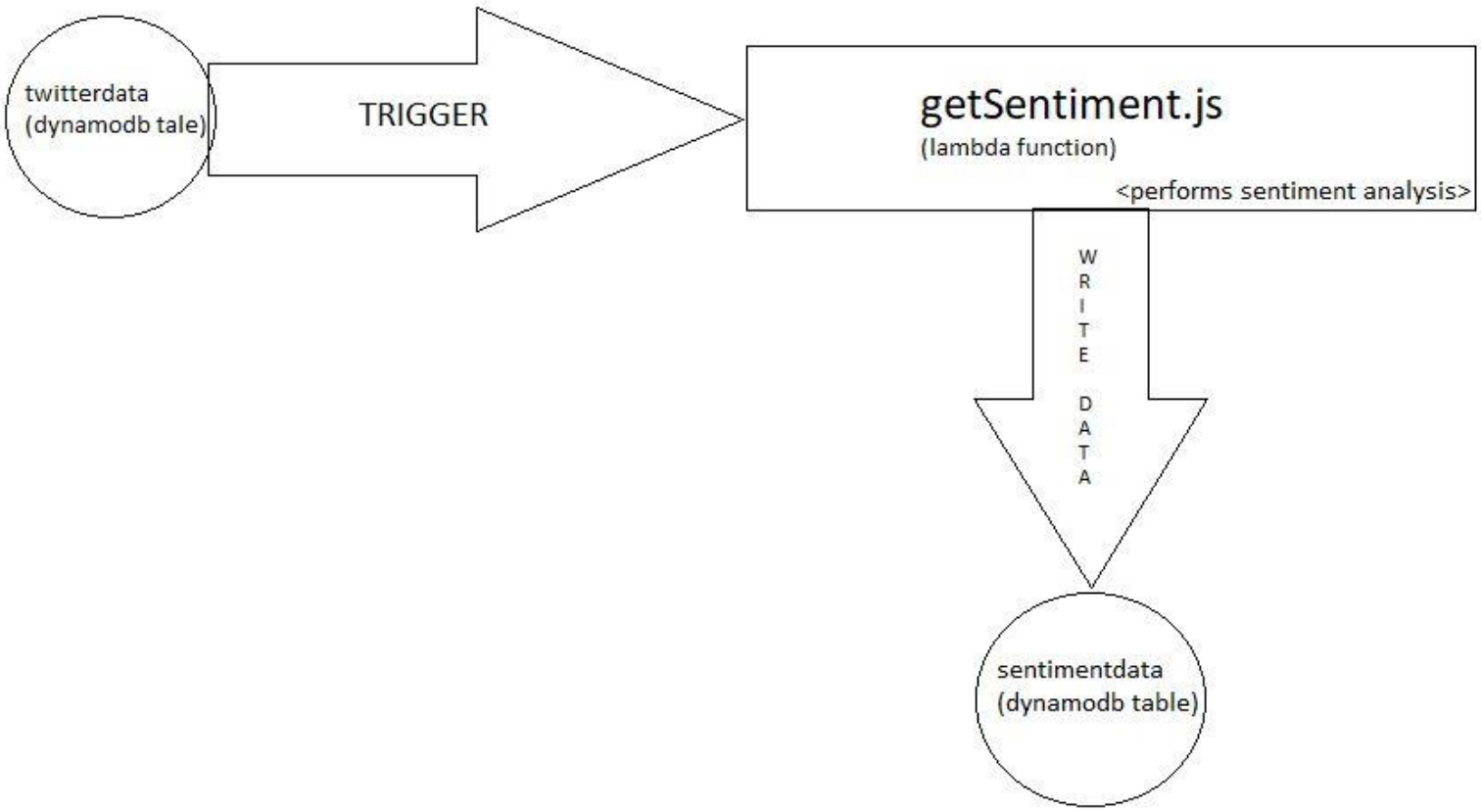


Figure. 2

There is another set off function that works together to perform the sentiment analysis of the text data stored in the database. When new data is gets inserted into 'twitterdata' table that triggers a function which performs the sentiment analysis of these text data and stores it into another table 'sentimentdata' as shown in figure 2.

The updating of the sentiment data table triggers a function that sends a response to all the connected clients which is described in figure 1.

Links:

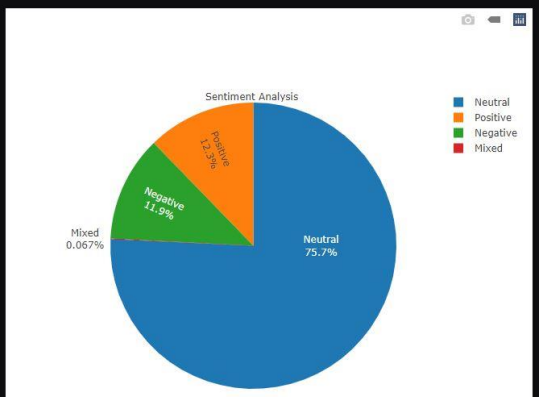
Public link for the website: https://cst3130cw2sub2.s3.amazonaws.com/index.html

Public link for the synthetic data plot:

https://chart-studio.plotly.com/~kazi123/0/synthetic-data-for-m00704320/#/

SCREENSHOT:

FrontEnd:



Plot | Data | Python & R | Forking History