2020

Twitter Analysis

CAB432

Assignment 2

Matthew Weir, Ethan Tracy

N10509020, N10520813

9/30/2020

Contents

Introduction

Purpose & description

The app we have created is a Twitter Analyser, where users can enter search terms and get a real-time stream of results as they are posted. Once inspecting the results, users are also provided with the sentiment analysis of the chosen tweet. Using this tool, it is possible to make observations of public opinion on a topic as events unfold, which can provide users with a unique insight on live events. The most noteworthy part of our application would have to be in its architecture, where we utilize two separate servers and socket.io connections in order to bypass the Twitter API's limitation of only allowing one stream to be open at a time.

Services used

Twitter filtered stream API (v.1.1)

An event listening stream that emits whenever a new tweet matches the search queries.

Endpoint: https://api.twitter.com/2/tweets/search/stream/

Docs: https://developer.twitter.com/en/docs/twitter-api/tweets/filtered-stream/quick-start

Natural

Provided a few tools that we used to retrieve sentiment analysis given a query.

Docs: https://www.npmjs.com/package/natural

React-tagcloud

Provided us with a way to display text in a word cloud.

Docs: https://www.npmjs.com/package/react-tagcloud

Socket.io

Provided us with low latency permanent communication between our client/server over a socket.

Docs: https://socket.io/

Aws ElastiCache (Redis cache)

Provided us a Redis cache from within AWS to allow all the instances to connect to easily.

Docs: https://aws.amazon.com/elasticache/

Use cases

US 1

As a	User
I want	To search a term/query
So that	I can see a feed of all tweets containing that term/query

US 2

As a	User
I want	To be able to see a stream of what other users have queried
So that	I can see what other users are searching

US 3

As a	User
I want	To be able to click a past users' button
So that	I can see all the users that have come through the stream previously.

US 4

As a	Financial trader
I want	To analyze current sentiment on an asset of my choice
So that	I can make more informed trading decisions

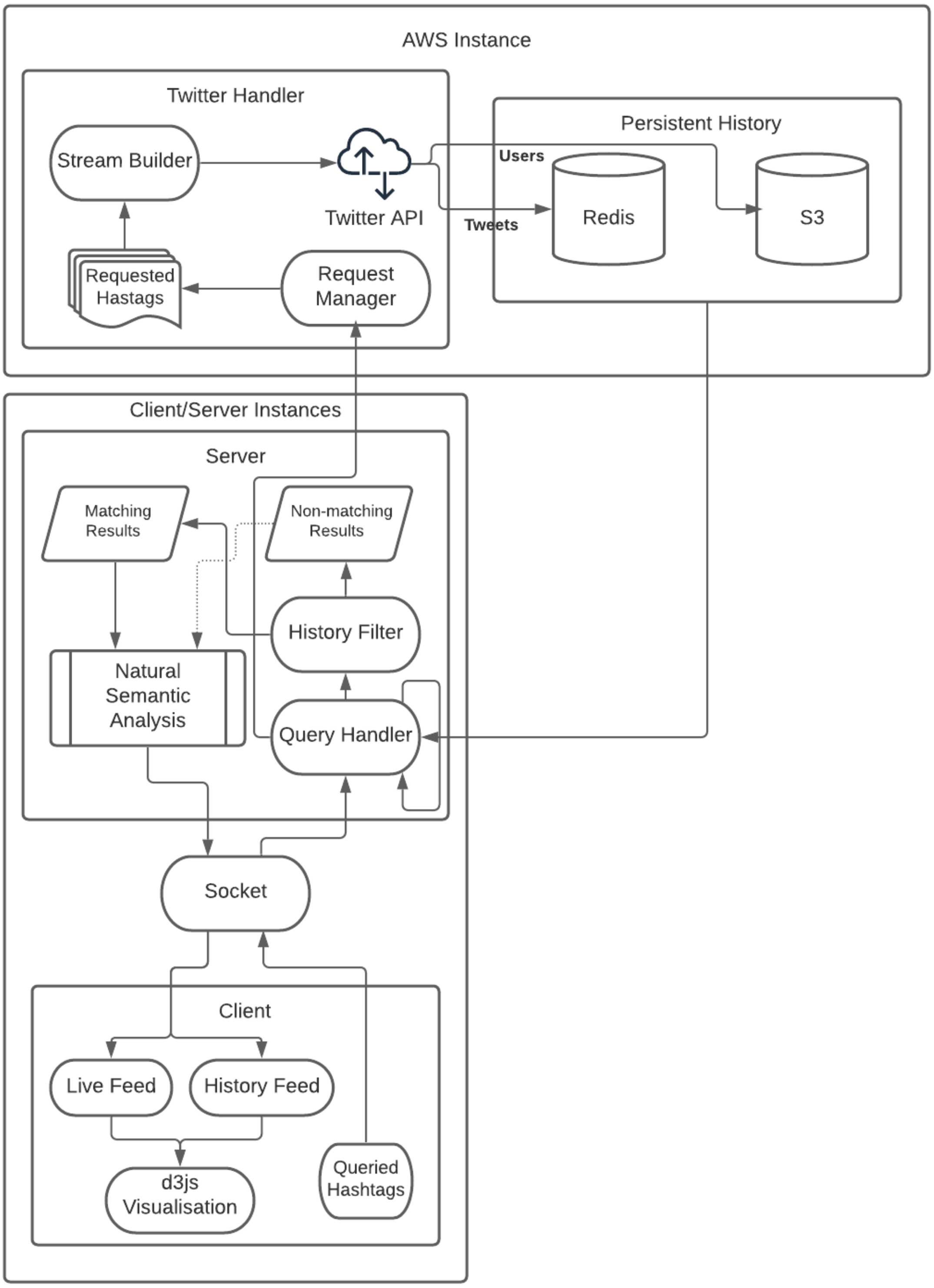
US 5

As a	Bachelor fan
I want	To observe reactions from other fans while the new episode is playing
So that	I can be more engaged in the show

Technical breakdown

Architecture

As previously mentioned, the architecture of our program is quite unique. This was needed in order to circumvent the Twitter APIs limitation of only allowing a single stream at a time, which for an application that needs be able to scale to many concurrent users just wouldn't work using a more traditional client-server architecture. The architecture that we ended up with can be seen in the diagram below.



Client / server demarcation of responsibilities

The crux of the application's architecture is that we have two servers, one which interfaces with the client and one that handles the twitter stream. The client-server is responsible for filtering and sending the client results, managing active queries and running sentiment analysis. The twitter handler has the job of handling the twitter stream and aggregating all client queries into one request which is later filtered by the aforementioned client-facing server(s). As tweets come into the twitter handler, they are saved to persistent history, with the Redis instance hosted the twitter-handler instance getting the full tweet object and S3 getting a modified, stripped-down version. When a client-server updates its queries in the twitter handler, the queries are added to a list of queries which are all searched at once. It then falls to the client-server to look for new tweets, which is done by checking for the existence of an incremented key value in the Redis, and then determining whether the tweet matches that client's queries or if it belongs in the history panel. The client-server then does the semantic analysis using the *natural* NLP library before emitting the result. When the socket for a client is closed, the client-server tells the twitter handler to drop that client's queries.

Response filtering / data object correlation

Tweet	tweet
string	created_at
int64	id
string	id_str
string	text
bool	truncated
int64	in_reply_to_status_id
string	in_reply_to_status_id_str
int64	in_reply_to_user_id
string	in_reply_to_status_id_str
string	in_reply_to_screen_name
User	user
Coordinates	coordinates
Places	place
int64	quoted_status_id
string	quoted_status_id_str
bool	is_quote_status
Tweet	retweeted_status
int	quote_count
int	reply_count
int	retweet_count
int	favorite_count
Entities	entities
Extended Entities	extended_entities
bool	favorited
bool	retweeted
bool	possibly_sensitive
string	filter_level
string	lang
Rules[ ]	matching_rules
Coordinates	coordinates

Twitter API Response

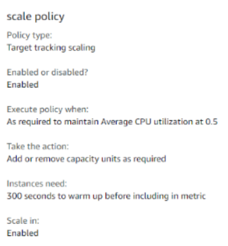
Tweet	tweet
string	name
int	sentiment
string	picture

Filtered S3 Store

Scaling and Performance

The application can be scaled, the instances load and boot successfully and are able to serve users, however due to the small amount of load that could be generated it was hard to get a scaling policy that allowed the application to scale in. With a bit of tweaking and monitoring the applications normal load and setting the CPU utilization lower the application would scale in.

However even with the CPU utilization execute policy set to maintain the application at 0.5 the application was not scaling down from three instances with the application idling at just over 0.6.


scale policy

Policy type:
Target tracking scaling

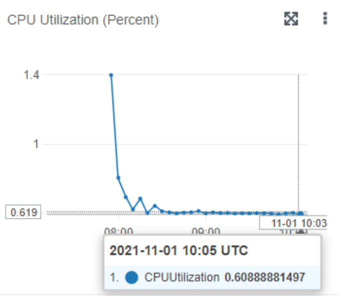
Enabled or disabled?
Enabled

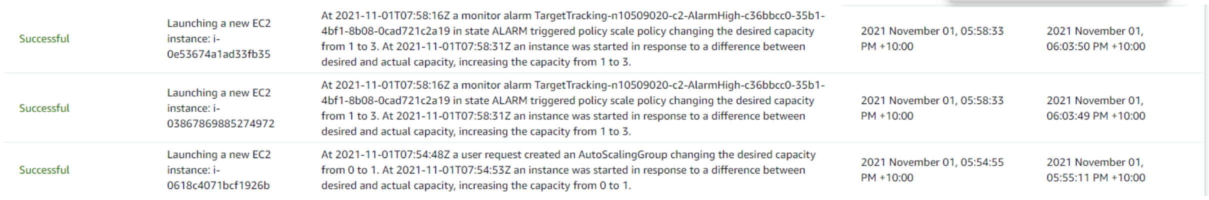
Execute policy when:
As required to maintain Average CPU utilization at 0.5

Take the action:
Add or remove capacity units as required

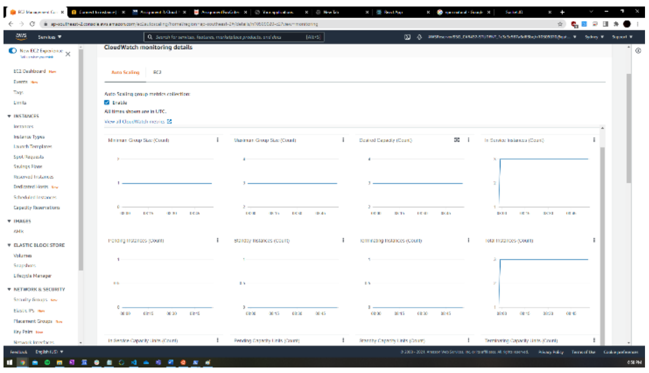
Instances need:
300 seconds to warm up before including in metric

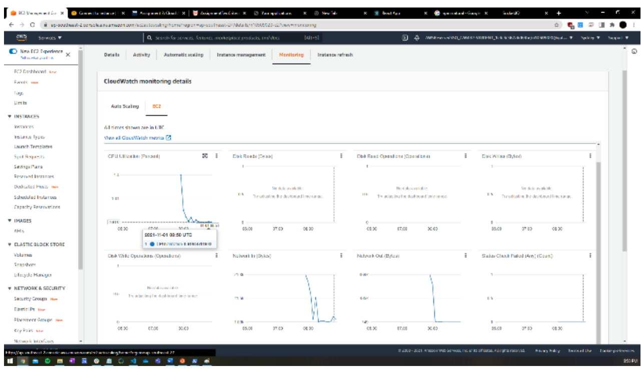
Scale in:
Enabled





Successful	Launching a new EC2 instance: i-0e53674a1ad33fb35	At 2021-11-01T07:58:16Z a monitor alarm TargetTracking-n10509020-c2-AlarmHigh-c36bbcc0-35b1-4bf1-8b08-0cad721c2a19 in state ALARM triggered policy scale policy changing the desired capacity from 1 to 3. At 2021-11-01T07:58:31Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 3.	2021 November 01, 05:58:33 PM +10:00	2021 November 01, 06:03:50 PM +10:00
Successful	Launching a new EC2 instance: i-038678698885274972	At 2021-11-01T07:58:16Z a monitor alarm TargetTracking-n10509020-c2-AlarmHigh-c36bbcc0-35b1-4bf1-8b08-0cad721c2a19 in state ALARM triggered policy scale policy changing the desired capacity from 1 to 3. At 2021-11-01T07:58:31Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 3.	2021 November 01, 05:58:33 PM +10:00	2021 November 01, 06:03:49 PM +10:00
Successful	Launching a new EC2 instance: i-0618c4071bcf1926b	At 2021-11-01T07:54:48Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 1. At 2021-11-01T07:54:53Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 1.	2021 November 01, 05:54:55 PM +10:00	2021 November 01, 05:55:11 PM +10:00


Application scaling up to three instances


Low application load

Performance when the application is just running as a single instance with multiple users querying high traffic streams holds up well and the application does not encounter any latency issues.

Test plan

Task	Expected Outcome	Result	Screenshot/s
Search for query	Results displayed and interactable	PASS	1
Analyze tweet sentiment	Accurate sentiment displayed in word cloud, and in the form of a smiley face the for sentiment of the text.	PASS	2
View history from other concurrent users	History panel populating appropriately	PASS	3
Remove queries once user disconnects	When a user leaves the site, their entry in the twitter-handler should be removed	PASS	4
No live feed data	Bouncing waiting icon until data	PASS	5
No history feed data	Bouncing waiting icon until data	PASS	6

View users page	Users are listed with their sentiment emoji displayed	PASS	7
View users page loading	Bouncing waiting icon until data	PASS	8
Refresh button	Refresh page back to home page.	PASS	-
User tries to add a space character in the input field	Error message displays		9
User tries to add a space character in the input field, error message disappears.	Error message disappears	PASS	-
Can't connect to backend	Nothing crashes and it waits for a connection.	PASS	-
Can't connect to Redis server	Nothing crashes and error message displays on server to connect to Redis.	PASS	-
Can't connect to s3	Nothing crashes and error message displays on server to check s3 credentials	PASS	-

Difficulties / Exclusions / unresolved & persistent errors /

By far the most challenging part of our implementation came from the twitter API and finding ways to bypass its limitations. We had conceptualized the idea of having a dedicated server for handling the twitter stream quite early on, however the specific details of this implementation were undefined. An example of this was how we were going to be continuously looking for new tweets from both the client and client-server's perspective. We ended up using the socket.io library to open connections between the front and back end, which allowed us to wait for events to handle incoming data.

Another difficulty came from moving the sentiment logic from the client to the client-server. When we were developing the prototype of this app, we had all the sentiment logic on the client. For the final build we were planning to move this over to the client-server, as this would help us more easily demonstrate scaling. When we did this however, we ran into many small issues, which in all just meant taking time to rewrite parts where necessary as well as rethink some of the logic behind when we fetch from the client-server.

Deploying the app to AWS via a docker container proved to be very difficult and we had to overcome many issues,

- Applications not finding the .aws file, even though it should have been able to find it automatically as it was in the correct security group. This was fixed this by gaining elevated privileges and adding the file in manually
- Docker containers not being able to see the external Redis cache. Initially this was fixed by setting the hostname/port to connect to the Redis cache correctly however the AWS security group didn't allow it to communicate with the server.

  We ended up using the AWS ElastiCache solution which proved a great success, and it was easy to implement.

- Configuring the docker containers to auto start using pw2, Resources were sparse however while researching we found that docker containers can have a run time setting the tells the container to continue running even after reboot.

The only difference between our brief and our final product is in our usage of S3, which was very loosely defined in our brief and ended up serving another purpose that we had not originally planned for. That use case is the users page, which lists all twitter users that have appeared in the stream and displays an emoji representing the sentiment of their tweet. S3 was a great choice for this because we did not need super low latency like Redis, and S3's listObjectsV2 function makes the process of trawling the bucket very easy.

We were able to finish our implementation without any outstanding bugs that we are aware of.

Extensions

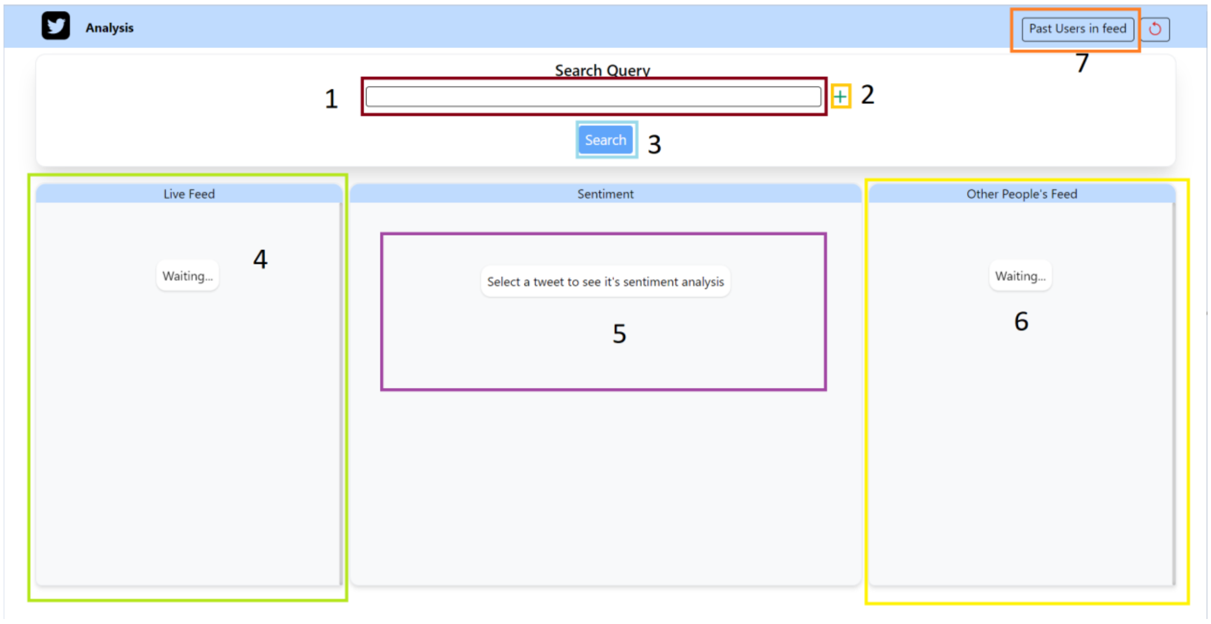
Within the Past Users page, it would be great if you were able to search for a user and see all over their tweets and have a sentiment analysis of all of that users' tweets. This would fill out the app and give it a much richer experience. As the users are stored in a state object, doing client-side filtering on the object would not be difficult, there for it would be a very quick search feature.

User guide

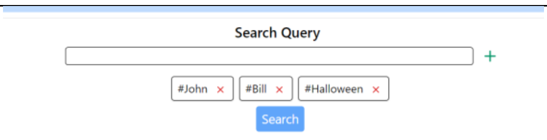
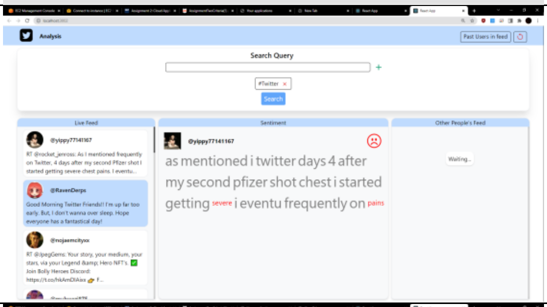
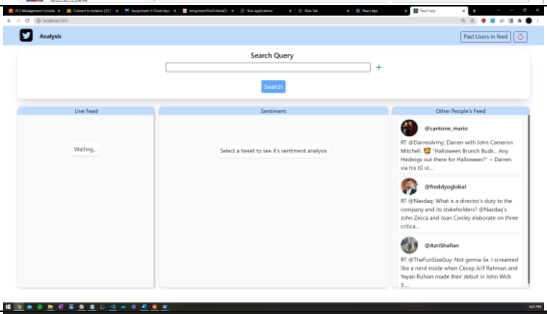
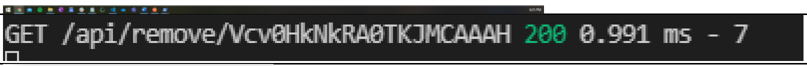
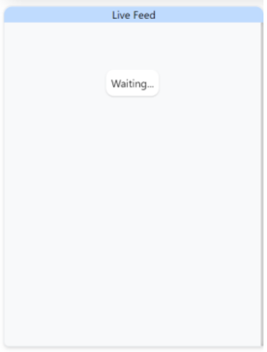
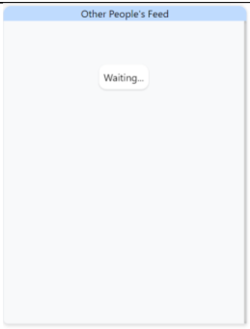
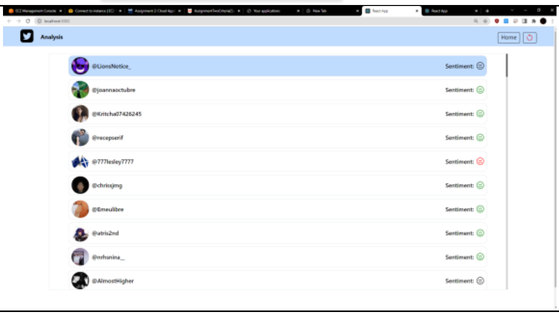
Using the application is quite straight forward. When you first load the page, you are greeted by a search box, three empty panels and a header. To start receiving tweets, enter a query into the search box (1), click the '+' on the right side of the box (2) and click search to send your quer(y/ies) (3). To remove hashtags, simply click on the cross beside its tag below the search bar.

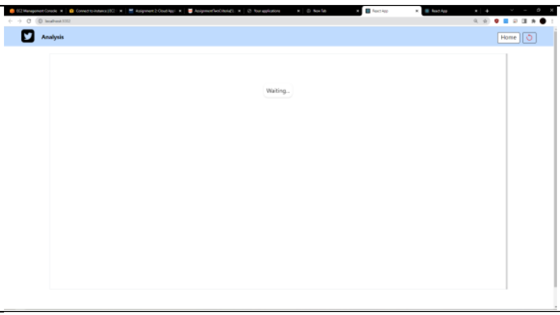
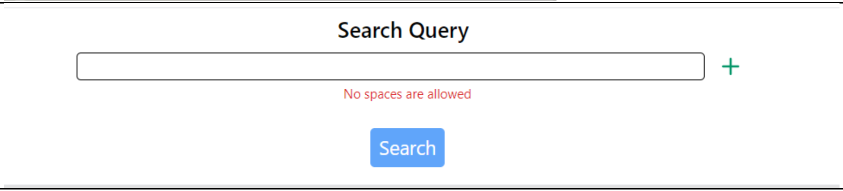
Once the stream finds tweets matching your query you will see them populating the feed panel (4). If there are any concurrent users, you will also see their tweets appearing in the history panel (6). You can view the sentiment analysis by clicking on any of the tweets that you see (5). This will display a word cloud and emoji representing the calculated sentiment.

To view the users page, click on the button in the top right in the header (7). This will fetch and display a list of users that have appeared in the feeds, as well as showing the sentiment of their tweets.



Appendices

1	 Search Query #John × #Bill × #Halloween × Search
2	 Analysis · Past Users in feed · Search Query · #Twitter × · Search · Live Feed: @yippy77141167 RT @rocket_jenross: As I mentioned frequently on Twitter, 4 days after my second Pfizer shot I started getting severe chest pains. I eventu… @RavenDerps Good Morning Twitter Friends!! I'm up far too early. But, I don't wanna over sleep. Hope everyone has a fantastical day! @nojaemcityxx RT @JpegGems: Your story, your medium, your stars, via your Legend &amp; Hero NFT's. ✅ Join Bolly Heroes Discord: https://t.co/hkAmDIAixx 👉 F… · Sentiment: @yippy77141167 as mentioned i twitter days 4 after my second pfizer shot chest i started getting severe i eventu frequently on pains · Other People's Feed: Waiting...
3	 Analysis · Past Users in feed · Search Query · Search · Live Feed: Waiting... · Sentiment: Select a tweet to see it's sentiment analysis · Other People's Feed: @cantone_mario RT @DarrenArmy: Darren with John Cameron Mitchell. 🥰 "Halloween Brunch Buds… Any Hedwigs out there for Halloween?" ~ Darren via his IG st… @freddyoglobal RT @Nasdaq: What is a director's duty to the company and its stakeholders? @Nasdaq's John Zecca and Joan Conley elaborate on three critica… @AzriShafian RT @TheFunSizeGuy: Not gonna lie. I screamed like a nerd inside when Cecep Arif Rahman and Yayan Ruhian made their debut in John Wick 3…
4	 `GET /api/remove/Vcv0HkNkRA0TKJMCAAAH 200 0.991 ms - 7`
5	 Live Feed · Waiting...
6	 Other People's Feed · Waiting...
7	 Analysis · Home · @LionsNotice_ Sentiment: · @joannaoctubre Sentiment: · @Kritcha07426245 Sentiment: · @recepserif Sentiment: · @777lesley7777 Sentiment: · @chrissjmg Sentiment: · @6meulibre Sentiment: · @atris2nd Sentiment: · @nrhsnina__ Sentiment: · @AlmostHigher Sentiment:

8	
9	Search Query No spaces are allowed