General System Overview

Welcome to our Twitter Replicator Project made using Python and MongoDB!! Choose an option to get started!

1. Tweet Search:

Think of keywords as your magic words. Type them into the search bar and watch the tweets that contain these words appear before your eyes. Each tweet will tell you its own story, revealing its id, date, content, and the username of its author.
Want to know more about a tweet? Just select it and you can check all its other aspects, including the url, replyCount, retweetCount, likeCount, and quoteCount.

2. User Search:

Looking for someone? Enter their name into the search bar and all users with matching usernames will appear.
Each user will introduce themselves by their username, displayname, and location.
Want to know more about a user? Just select them and all their details will be revealed.

3. Top Tweets:

Want to know what's trending? Enter a number 'n' and select a field (retweetCount, likeCount, quoteCount) to see the top 'n' tweets based on the selected field.
Each tweet will present its id, date, content, and the username of the person who posted it.
Want to know more about a tweet? Just select it and all its details will be revealed.

4. Top Followed Users:

Enter a number 'n' and you'll see the top 'n' users ranked by followersCount.
Each user will step forward and present their username, displayname, and followersCount.
Want to know more about a user? Just select them and all their details will be revealed.

5. Compose a Tweet:

Enter your tweet content and press enter. Your tweet will be added to the database with the date field set to the system. All other fields will be set to null or zero.
Now everyone can see what you have said.

6. Exit Program:

Sad to see you go! Start up the program again at any time to interact with others. Bye!!

Algorithm Details

Upon starting the application, the `load_data()` function was initiated to load the dataset into the database and create the necessary indexes. The user was then presented with a menu of options. If they chose to search for tweets, they entered keywords and `search_tweets` was called to return matching tweets. For more details about a tweet, another implementation was used.

If the user wanted to search for other users, they entered a keyword and search_users was called. To view more details about a user, main file implementation was used.
To list the top users, the user entered a number 'n' and top_users(n) was called. Similarly, to list the top tweets, the user entered a number 'n' and a field, and top_tweets was called.
Finally, to compose a tweet, the user entered the content and `compose_tweet(content)` was called to insert the tweet into the database. This walkthrough ensured all functionalities were thoroughly tested according to the provided rubric. It's important to remember that testing is an iterative process, and it was necessary to retest after fixing any discovered issues.

To speed up our searches, we created indexes on attributes that we will query on. We created indexes in load_json.py. Indexes were put on “content”, "retweetCount", "likeCount", "quoteCount", "user.displayname", and "user.followersCount". This is because these are the attributes that we use to query when searching for tweets/users and searching for the top ‘n’ tweets/users.

Testing Strategy

The test began with Phase 1, where a dataset of 420k entries was loaded into the database. The time taken to load the data and create necessary indexes was measured, ensuring it was significantly less than 3 minutes (~ 45 sec). The correctness of the database and the efficiency of the code were verified with the different modes of indexing done.

In Phase 2, several functionalities were tested. The search for tweets was tested with various keywords. It was verified that all tweets matching the keywords were returned and that for each matching tweet, the id, date, content, and username were shown. A tweet was selected to verify that all fields were shown.

The search for users was tested with a keyword. It was verified that all authors matching the keyword were returned and that for each user, the username, displayname, and location were displayed. A user was selected to verify that all fields of the user were shown.

The listing of top users was tested with different values of ‘n’. It was verified that the top ‘n’ users based on followersCount were listed. A user was selected to verify that all fields of the user were shown.

The listing of top tweets was tested with different values of ‘n’ and different fields. It was verified that the top ‘n’ tweets based on the selected field were listed. A tweet was selected to verify that all fields were shown.

Finally, the composition of a tweet was tested. A tweet was composed and it was verified that the tweet was inserted into the database with the date field set to the system date and username set to “291user”. All other fields were set to null or zero.

This test plan ensured that all functionalities of the database were thoroughly tested according to the provided rubric. It’s important to remember that testing is an iterative process, and it was necessary to retest after fixing any discovered issues.

Project break-down and group coordination

All coordination for group work, meeting times and file exchanges were done through Github and Discord. We also met to complete the synchronization of our programs during the weekends in the university.

1. main.py: This file is the main file which handles all of the front end screen that the user would see.

The main screen interface with the options and interface for user selection were implemented by Mikael Nineza (1 hr). The error handling done mainly during the main file, including user inputs and troubleshooting was done by Mikael Nineza (2 hrs). Connection to the database of the main file, as well as references made to the databases throughout this file were done by Pratham Prajapati (1 hr). Search tweets, search users, list top tweets, list most followed users, compose a tweet options in the main file were provided and coded for by Mikael Nineza (2 hrs). Troubleshooting for options implementation was done by Pratham Prajapati (1 hr).

2. Load-json.py: This file contains all code for importing a database and indexing in for faster reading and computations and was done by Pratham Prajapati (2 hrs)
Troubleshooting for this file in terms of indexing, was done my Mikael Nineza and Pratham Prajapati (2 hrs)

3. Functions.py: This file contains all internal code for all of the functions made and implemented.

The coding for the functions, search_tweets, search_users, top_tweets and compose_tweets was done by Aakarsh Gupta (5 hrs). Troubleshooting for MongoDB language was done by Aakarsh Gupta (1 hr). Database handling and fixing of top_users was done by Mikael Nineza (2 hrs).

4. Project Report PDF: Contains all relevant documentation about the development and group collaboration procedure.

Completed by Aakarsh Gupta (2 hrs)