Mini Project 2 Project Report

By: Laura Kirezi, Xavier Salm, Shubhkaran Dhillon, Pollo Latysheva

General Overview

This project allows users to load a json file and search the contents of the file using MongoDB. It is broken up into two programs. The first program, load-json.py, allows the user to load any JSON file by inputting the name of the file and it also prompts the user to input the MongoDB server port connection number. This program then creates a MongoDB document store which stores all the content from the JSON file. From there, the user is able to operate the loaded document store using the twitter2.py program. This program allows the user to search for tweets using keywords, search for users, list a certain number of users or tweets and compose a tweet.

Algorithm Details

Phase 1: load-json.py

After connecting to the given port and using the given json file name (without the .json extension at the end, the program adds that), the program first starts by preparing the json file to converted to mongodb by turning it into a valid json file (adding opening and closing square brackets, adding commas, removing the any trailing newline characters at the end of the json file), Then, in order to scale well with larger code bases and not consume too much memory, the json file is imputed in batches of 10000 into the mongodb database. After that, indexes are created to make future queries faster, specifically, indexes for the following fields are made: retweetCount, likeCount, quoteCount, user.displayname, user.location, content, and user.followersCount.

Phase 2: twitter2.py

Search Tweet Function

The user is prompted to input one term or multiple terms. The program then turns the inputted terms into a list to accommodate for multiple keywords. Then, it iterates through the keywords list and compares the keywords to the words in the content of the tweets provided in the JSON file. This function utilizes regular expressions pattern matching in order to optimize searching large data collections by matching particular string characters instead of multiple search queries. This is especially helpful for tweets that contain lots of different characters such as hashtags. The function then displays the ID, Username, Date and Content of each matching tweet. Then, the program prompts the user to select one of the matching tweets and displays all the fields of the selected tweet. The function continues to prompt the user to search for keywords until they choose to exit.

User Search Function

The user is prompted to enter a keyword (case-insensitive). Based on the provided keyword, the system conducts the search that matches displayname or location. The results get displayed without any duplicates, and then the user is asked whether they would like to see full information about any particular user. If “n” (case-insensitive) is entered, the user gets returned to the main menu. If anything but “y” (case-insensitive) is provided, the user sees the error message and gets returned to the main menu.

Upon entering “y” (case-insensitive), the user is prompted to provide a username of the user they want to take a look at. If the user provides a non-existing username, the system generates an error message and returns the user to the main menu. If an existing username is provided, the user can see all fields of the user for the displayed record, alongside the maximum followersCount which is displayed as a new variable maxFollowersCount at the very beginning. After the display, the user is automatically returned to the main menu.

List Top Tweets Function

The user is prompted to select which count they want the top tweets of, and how many tweets they would like to see (n). Then, a query is performed to find the top tweets by the given count, limiting the results by n. This query is then turned into a list for ease of indexing to specific entries. Once this list of top tweets is printed to the user, the function enters a while loop to allow the user to select as many tweets as they want to see more info. To display more information, the user gives a number *index* where index-1 is the correct entry in the list. From there, the id value of the list is accessed and used to search the database for the tweet with that id, and the entire result is printed.

List Top Users Function

The topUsers function uses a MongoDB aggregation pipeline to find and display information about top Twitter users based on followers count. It prints basic details for each user and then enters a loop, allowing the user to select a username for more detailed information. The function interacts with the user until they choose to exit, at which point it returns to the main menu.

Compose Tweet Function

The `composeTweet` function inserts a new tweet into a MongoDB database. It takes tweet content as input, generates the current UTC time, and creates a tweet document with relevant fields such as date, content, username. The tweet is then inserted into the database, and the function prints the ID of the inserted tweet. Note: The global variable `db` is assumed to be set elsewhere in the program.

Testing Strategy

To test our program, we used several different databases that were provided in the assignment, including the large farmers-protest-tweets-2021-03-5.json. When figuring out the edge cases tests, we relied on the provided assessment rubric.

Load-json.py

- Ensured that the right JSON file is being loaded into document store and added each item of the file is added correctly
- Tested for time efficiency using small files (10.json) and large files (farmers-protest-tweets-2021-03-5.json)
- Ensured that the larger file took around 1min to load data
- Tested that the indexes were created correctly
- Tested that the program could use invalid json files (missing square brackets, missing commas, new line at EOF)

Search Tweet Function

- Searching w/ one keyword and multiple keywords
- Ensured the tweets matched all the keywords (AND semantics) and no word fragments
- Ensured the necessary information was displayed (ID, date, username, content)
- Tested case insensitivity and
- Tested selecting a matching tweet and ensured all the fields were displayed (including replyCount, retweetCount, etc.)
- Test with newly composed and added tweet
- Tested that words would only match if they exactly matched the term searched
- Ensured that the a tweet would match only if it contained the exact terms, in order

User Search Function

- Testing the keyword search for displayname and location
- Ensuring that all the users that match the keyword are displayed correctly (username, displayname & location)
- Ensured there were not duplicates

- Tested that the user could select a user by entering their username and see all the fields
- The maximum followersCount (as a new variable maxFollowersCount) is displayed as the very first separate entry

List Top Users Function

- Tested if the user is prompted for number, n and if the function displays the correct number of users
- Tested if the users that are displayed are based on followersCount
- Ensure all the correct information is displayed: username, displayname & followersCount
- Test if user is prompted to select a user and all the correct information is displayed

List Top Users Function

- Check if user is correctly prompted to input number, n and to select retweetCount, likeCount or quoteCount
- Test that the right information is displayed based on the user input
- Ensure that result is ordered in descending order for each field option (retweet, like, quote)
- Test if user is prompted to select a tweet and all the correct information is displayed

Compose Tweet Function

- Test that the user can input a tweet
- Ensure that the tweet is inserted correctly into the database
- Check to see if the correct information is stored in the correct fields: date, tweet content and username “291user”
- Ensured that the new document was correctly formatted, and thus could be actually searched in other queries.

Main Menu Function

- Tested that it only worked with the port connection that is currently running
- Tested that all the commands brought user to the right function
- Ensured that the exit command closed the program effectively

Source Code Quality

Commented the code, created additional functions for elements of the code that would end up being repeated more than once, While we did not use docstrings, there is a comment at the start of each function outlining the purpose of that function. Proper variable names were used throughout the code.

Group Work Strategy

Division of Tasks:

- Laura: tweetSearch function (7 hours, 100%), General Testing of twitter2.py (30 mins, 40%) Design Document: General Overview, Phase 2 Algorithm Details & Testing Strategy (1 hour, 60%)
- Xavier: load-json.py (6 hours, 100%), mainmenu function (.5 hours, 100%), topTweets function (3 hours, 100%), Design Document: Phase 1 Algorithm Details, Source Code Quality (20 minutes, 100%)
- Shubhkaran: composeTweet function(3 hours, 100%), topUsers function(6 hours, 100%)
- Pollo: userSearch function and its helper function userInfo (8 hours, 100%), General Testing (40 mins), edits to the design document (30 mins).

Method of Coordination: We communicated over Discord calls to allocate tasks at the beginning, did regular check-ins throughout , worked on final testing and debugging as well as worked on the project report.