Title: Mini-Project 2
Authors: Abdelhadi Abou Hachem, Oluwaseyifunmi Bamijoko, Syrus Muhammed, Rachit Anugula

General Overview

For the purpose of learning the concept of working with data files without SQL databases, we developed this program that interfaces with MongoDB for sorting and processing Twitter data and allows users to perform functions on the data. In Phase 1, the tweets are loaded from a JSON file into a MongoDB collection, taking a JSON file and MongoDB server port # as input, then it connects to the server and creates a database “291db” and a collection “tweets”. In Phase 2, the user is able to perform various tasks on the MongoDB database, such as searching for tweets and users, listing top tweets and users based on specific criteria, and composing a tweet. The search functions support keyword matching and are case-insensitive.

Algorithm Details

Data Loading

The “load-json.py” is a Python program that uses MongoDB’s utility (mongoimport) to construct a collection and create a database called “291db”. This method is able to handle large JSON files by inserting data in batches which is more efficient and requires less memory usage.

Search Algorithms

The search_tweets function performs a case-insensitive search on tweets based on the user’s entered keywords. It uses a regex-based search on the “content” field to find tweets with the provided keywords. Search_users is similar to search tweets but it finds keywords in the user’s display name and location fields.

Sorting and Listing

List top tweets allows users to list top n tweets based on 3 fields (retweetCount, likeCount, quoteCount) and sort them in descending order of the chosen field.
List top users allows users to list top n users based on their “followersCount”, results are sorted in descending order as well.

Insert Algorithms

The compose_tweets function prompts the user to input the text that they would like to post as a tweet. It then performs a sequence of queries to insert a document into the “tweets” collection.

Indexes for Optimization

Compound text indexes are implemented by using MongoDB’s ‘create_index’ function, which improves search performance. Instances include “user.username”, as well as “content” and “location” fields.

Testing Strategy

Members would test database connection, data insertion, and search queries as well as make sure the load-JSON works properly and the file is imported successfully.
Group members would each test every function to verify that the outputs would match the expected results, ensuring the correct implementation of every function. Using the provided JSON files we assessed our project’s functionality using varying sized datasets.

<u>Group work breakdown</u>

Group members communicated through Discord to organize tasks, and held multiple in-person meetings to collaborate and ensure that the project was on track. The group's method of editing the source code was done through Github, members would specify the tasks they were working on in order to maximize efficiency.

Name	**Work Item**	**Estimated Time/Progress**
Abdelhadi Abou Hachem	1- List top tweets 2- Code optimization /Enumeration 3- Debugging 4- Project Report	1) 5 hours 2) 2 hours 3) 5 hours 4) 2 hours
Bamijoko Oluwaseyifunmi	1- Load-json.py 2- Search_users 3- search_tweets 4- Debugging	1) 4 hours 2) 3 hours 3) 3 hours 4) 2 hours
Syrus Muhammed	1 - compose_tweet() 2 - Debugging/Testing 3 - Project Report	1) 3 hours 2) 5 hours 3) 2 hours
Rachit Anugula	1 - list_top_users() 2 - debugging/testing 3 - Project report	1) 4 hours 2) 4 hours 3) 2 hours