Twitter Search Application

Team 5 – Neeti Patel, Aalap Patil, Abhishek Bahad, Rushabh Karnavat

1. Introduction

The objective of this project is to design and implement a search application for a Twitter dataset that allows users to perform searches based on usernames, hashtags, keywords, top ten metrics, and time ranges. The data are stored using two different database management systems, MySQL and MongoDB. Indexing is used on some fields to optimize the query performance. A cache system is implemented using Least Recently Used (LRU) technique that stores popular data for faster access, thereby enhancing the query response times. According to the text in the tweets, sentiment analysis is performed. Finally, a simple and user-friendly UI is developed that allows users to perform different types of searches.

2. Dataset

The Twitter dataset is a JSON file that contains information about tweets. A few embedded JSON objects in the dataset have further details about the users and the tweets.

The insignificant variables have been removed from the dataset in the data-cleaning process. Refer to section 12 for the variables included in the tables and collections.

3. Project Architecture

Figure 3.1 demonstrates the project architecture. The user and retweets information is stored in a relational database management system, MySQL, while the tweets and comments are stored within a non-relational database, MongoDB. When a search is initiated, the application queries the appropriate database for the request and returns the results. Additionally, the searched data is cached and updated on a Least Recently Used (LRU) basis (ref. section 5.6).

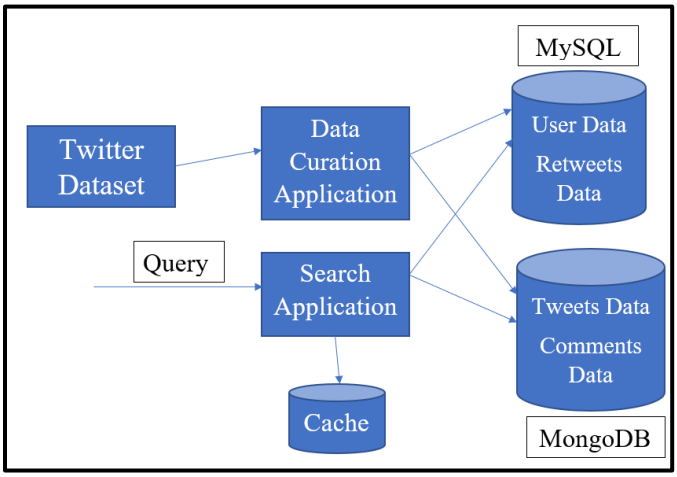

Figure 3.1 Project Architecture

4. Data Model and Datastores

4.1 Modeling and Storage of the User and Retweets Data

The user and retweets data are stored in MySQL. The primary reason is that user and retweets data have a fixed schema and that these details are not updated often, so they are stored on disk. The data is inserted into the MySQL database one record at a time. Refer to section 12 for the columns included in the user and retweets tables.

For the user table, it is checked whether the user already exists in the table. That record is added if the user does not exist in the table. If the user already exists, the timestamp variable is checked, and if it is more recent, then the user record is updated. Additionally, if there are more users' details in the 'retweeted_status' or 'quoted_status' variables of the record, it is checked whether those user records exist in the user table. A new record is created for that user if they do not exist. If they exist, the timestamp variable is checked, and only the most recent record is kept in the user table. There are a total of 90336 users in the table.

4.2 Modeling and Storage of the Tweets and Comments Data

If the text starts with "RT," the program considers it a retweet and retrieves the tweet ID, the original tweet ID from the retweeted status, and the user details of the person who retweeted the original tweet.

The tweets data are stored in MongoDB. The database contains two collections – tweets and comments and is schema-less which may be updated frequently, so MongoDB is used. The data is inserted into the MongoDB database one record at a time.

The tweets or comments data insertion is checked whether it is a retweet by checking the first two characters in the 'text' field. Retweeted tweets or comments get inserted in their respective collections.

To insert data for the tweets in the tweets collection, it is checked if a given tweet falls in the category of tweets or comments. For this, the 'in_reply_to_status_id_str' field is being checked. If this field is 'None,' then it is considered an original tweet tweeted by a user. Otherwise, it is regarded as a comment on a tweet by a user. After applying conditions, if the text is qualified as a tweet, it is inserted if it is not already in the database. Afterward, if the 'truncated' field is marked as 'True,' then the field

'extended_tweet' is inserted, and if the field 'is_quote_status' is 'True,' then the field 'quoted_status_id_str' field is inserted in the tweets collection.

To get the data for queries related to a time range, the 'created_at' field is inserted in a particular format

'%a%b%d %H:%M:%S%z%Y,'

where %a: Abbreviated weekday name, %b: Abbreviated month name, %d: Day of the month as a decimal number, %H: Hour in 24-hour format as a decimal number, %M: Minute as a decimal number, %S: Second as a decimal number, %z: Time zone offset from UTC in the form of +HHMM or -HHMM, %Y: Year as a four-digit decimal number. This field is converted into epoch time and inserted as a 'timestamp' in the tweets collection. Lastly, if a tweet is already present in the collection, the popularity of the respective tweet gets updated. Refer to section 12 for details about the fields.

Similarly, the comments collection is created for the users' comments on the tweets. For the fields in this collection, refer to section 12. There are 36304 tweets and 13606 comments in the collection.

4.3 The Popularity System

A popularity system has been defined using the number of replies, likes, followers, quote tweets, retweets, and comments by giving weight to each factor. See Table 4.3.1 for the assigned weights.

Factor	Weight (Per 1 count)
No. of Replies	+3
No. of Likes	+5
No. of Followers	+10
No. of Quote tweets	+20
No. of Retweets	+20
If the author is Verified	+10000

Table 4.3.1 Weights given to different factors.

4.4 The Time System

This system first displays the most recent tweets according to the user's search request. When the search application displays tweets, they can be filtered by the last day, week, and month.

The current time (in seconds) is set as when the most recent tweet is in the Twitter dataset. For ordering by day, the last 24 hours are considered with respect to the set current time. The results are then sorted according to popularity [4.3]. Similarly, for ordering by week and month, the last seven days and the last 30 days with respect to the set current time are considered, respectively.

5. Search Application Design

'Tkinter' is used for the User Interface design because it has an object-oriented interface.

5.1 Ordering of Search Results

If a search starts with '@,' then the application searches the user account's screen name in the user table and orders the results based on the number of followers.

If a search starts with '#,' the application searches for that hashtag in the tweets collection and orders the results as tweets based on popularity [4.3].

If a keyword search is done, the application searches for tweets with that specific word in the tweets collection and retrieves results based on popularity [4.3].

If a user presses the 'top 10 users' button, the search application retrieves the ten users from the user table with the most followers in the database.

If a user presses the 'top 10 tweets' button, the search application retrieves the ten tweets from the tweets collection according to popularity [4.3] in the entire database.

If a user presses the 'by day,' 'by week,' or 'by month' button, the search application retrieves the top tweets of the day, week, or month respectively, according to popularity [4.3] in the entire database.

When the 'Retweets' or 'Comments' button is used, the data is searched in the Retweets table or the Comments collection, respectively. Retweets are ordered by popularity and comments by timestamp.

5.2 Query Translation Process for Datastore

The search query text is obtained from the search box using the get() function and stored in a string variable. The variable is then passed on, running the query and retrieving results. If the variable begins with '@,' the data is searched in the user table. A similar process is followed if it starts with '#' or plain text, but the data is searched in the Tweet Collection.

5.3 Drill-down Options in the Search Results

When a user is searched for and their username is selected, the system displays their information and a button to show the user's tweets. Clicking the 'Show Tweets' button displays all tweets posted by that user.

When a tweet is searched by hashtag or text, its tweet ID, author's username, and tweet text are displayed. By clicking on the author's ID, we can see their information.

Clicking on a specific tweet ID shows the tweet along with its comments, users who have retweeted, sentiment, and the number of likes. See Figure 5.3.1 for in-detail drill-down options.

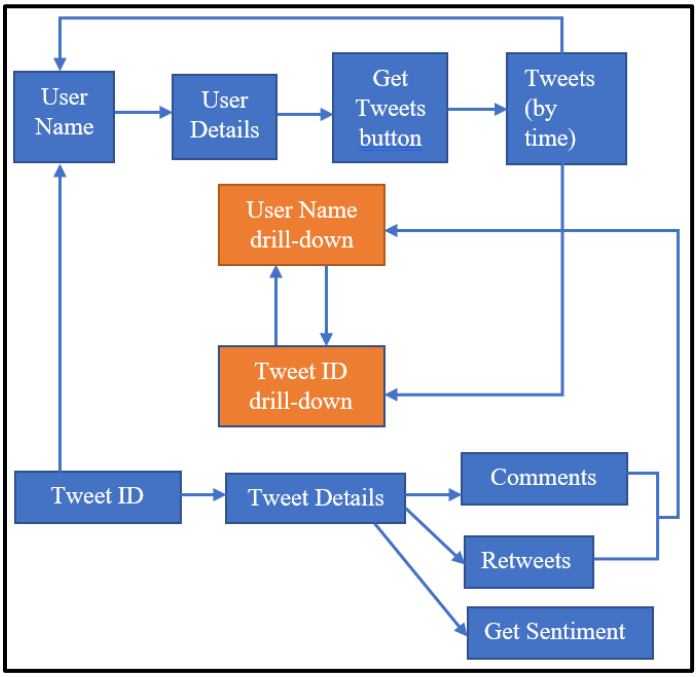

Figure 5.3.1 Drill-down options

6. Caching and Indexing

An LRU cache has been implemented using Python Dictionaries for the search application. Whenever a user utilizes the search box and clicks the search button, the application checks whether the searched element is in the cache. If the component is present in the cache, the results are retrieved directly from the cache. If not, the search text is added as a key, and the top 10 results retrieved from the database are added as a value in the cache dictionary. The implementation checks whether the cache is full while adding the element. The least recently used element is evicted from the dictionary if it is full, and a new element is added. Additionally, during each addition operation, the dictionary is written and saved to a file for checkpointing purposes. The next time the search application runs, the saved file can be loaded and ready to retrieve the cache. Using this LRU cache, the retrieval time of the search application has been improved by a factor of 10^4. See Images 6.1 to 6.6 for the observed time difference after caching.

To improve the efficiency and speed of the retrieval process, indexing has been implemented on the tweet ID in the comments collection and the user ID in the tweets collection. By applying indexing to these fields, the search application can quickly locate and retrieve relevant tweets and comments, resulting in a more responsive and efficient search experience for users.

Image 6.1 Keyword results before caching

Image 6.2 Keyword results after caching

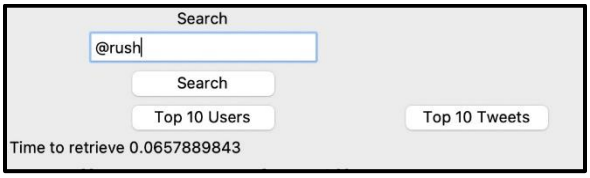

Image 6.3 Username results before caching

Image 6.4 Username results after caching

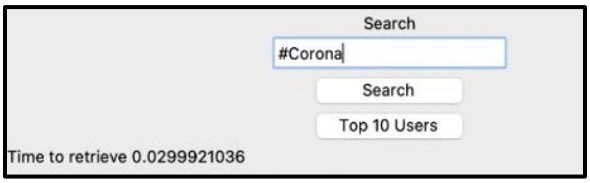

Image 6.5 Hashtag results before caching

Image 6.6 Hashtag results after caching

7. Results of the searches

7.1 User Search

When a search begins with '@,' the application will display results. The results will be sorted in descending order based on the total number of followers of the users. The search will show all the results if there are fewer than ten total results. If there are more than ten results, only the top 10 will be displayed with an option to load more.

7.2 Hashtag Search

When a search begins with #,' the application will display tweets according to popularity [4.3]. There are distinct options for getting results based on time [5.2].

7.3 Keyword Search

When a keyword search is initiated, the application will display tweets that include that keyword sorted according to popularity [4.3].

8. Sentiment Analysis on Tweets

8.1 Evaluation Metrics for sentiment analysis

The 'truncated' field is checked, and if the value is 'True,' the 'full_text' field of the 'extended_text' object is used for sentiment analysis. If the value is 'False,' the 'text' field is used.908848

To analyze the sentiment of tweets in the search application, the NLTK library is utilized to eliminate stop words from the text of each tweet. Next, TextBlob is used to calculate the sentiment polarity of each tweet,

which helps categorize it as Positive, Negative, or Neutral based on the polarity score. These values are then saved in the database for each tweet. The tweet's sentiment is retrieved and displayed whenever a user presses the 'Get Sentiment' button in the search application. This process allows users to quickly and easily determine the sentiment of tweets relevant to their search queries.

8.2 Limitations

There are limitations to using this sentiment analysis program.

- This basic Sentiment Analysis program does not classify based on human emotions like joy, excitement, anger, etc.
- Tweets in other languages are classified as "Neutral" since the program cannot give a polarity score to the words in other languages.
- The text in the tweets is often short, so an accurate analysis is complex.
- The program evaluates tweets based on content, not context, making accurate predictions difficult.

9. Conclusions

In conclusion, the search application designed and implemented for the Twitter dataset allows users to perform searches based on various criteria, such as accounts, hashtags, keywords, top ten metrics, and time range. The application utilizes two different database management systems, MongoDB and MySQL, to store data efficiently, and indexing is used to optimize query performance. Implementing a cache system using the LRU technique also enhances query response times. The sentiment analysis of the tweets is performed using the TextBlob library. A simple and user-friendly UI developed using Tkinter allows users to explore the search results further and access additional details. Lastly, the application also times the searches to compare the differences between searches with and without a cache. Overall, this project aims to provide an efficient and effective way for users to search through Twitter data.

10. References

1) https://developer.twitter.com/en/docs/twitter-api
2) https://www.mongodb.com/docs/manual/
3) https://docs.python.org/3/library/tkinter.html
4) https://dev.mysql.com/doc/

11. Team Contributions and Roles in the Project

Aalap Patil was responsible for designing the project architecture and the UI application and implementing data storage in MongoDB.

Abhishek Bahad was responsible for implementing the retrieval queries for MySQL and MongoDB.

Neeti Patel was responsible for implementing the data storage of MySQL, Sentiment Analysis, and Documentation.

Rushabh Karnavat was responsible for the designing of the cache and search application.

11.1 GitHub Repository

https://github.com/patilaalap/694-2023-Team5

11.2 Github Usernames

1) neetipatel
2) patilaalap
3) rpkarnavat
4) abhishekbahad

11.2 Project Demonstration

Use this Google Drive link to watch a demonstration of the search application – https://drive.google.com/file/d/1Q1zJ7xRhzcle_ZhuxsRMhqkKrq3tq5Bi/view?usp=sharing

12. Appendix

- Columns in the User Table –
  1) id_str - Unique identifier for a User.
  2) name - The name of a user, as a user-defined it.
  3) screen_name - Alias that a user identifies itself.
  4) description - Description of a user's profile.
  5) verified - A user with a verified account.
  6) followers_count - The number of followers a user account has.
  7) friends_count - The number of users a user account follows.
  8) created_at - The DateTime, a user's account, was created on Twitter.
  9) timestamp - Epoch time of a tweet by a user.
- Columns in the Retweets Table –
  1) id_str - The unique identifier for a Tweet.
  2) original_tweet_id - Tweet id of the original tweet on which the retweet has been made.
  3) user_id_str - Unique identifier for a User.

4) screen_name - Alias that a user identifies itself.

- Fields in the Tweets Collection –
  1) created_at - Time when a Tweet is created.
  2) id_str - The unique identifier for a Tweet.
  3) text - The actual text of the status update.
  4) truncated - Indicates whether the value of the text parameter was truncated.
  5) quoted_status_id_str - It is the Tweet ID of the quoted Tweet.
  6) is_quote_status - Indicates whether a tweet is a Quoted Tweet.
  7) reply_count - Number of times a Tweet has been replied to.
  8) retweet_count - Number of times a Tweet has been retweeted.
  9) favorite_count - Indicates approximately how many times a Twitter user has liked a Tweet.
  10) hashtags - The field contains all the hashtags from a tweet.
  11) extended_tweet - It contains additional information about a tweet's content when it contains more than 140 characters.
  12) popularity - The field has been defined using the number of replies, likes, followers, quote tweets, retweets, and comments by giving weight to each factor.
- Fields in the Comments Collection –
  1) created_at - The date and time when a comment was posted on Twitter.
  2) id_str - A string representation of the unique identifier of a comment.
  3) text - The content of a comment.
  4) truncated - A boolean value indicating whether a comment text was truncated because it exceeded the maximum length allowed by Twitter.
  5) quoted_status_id_str - The unique identifier of the tweet that a comment is quoting.
  6) is_quote_status - A boolean value indicating whether a comment is a quote tweet.
  7) quote_count - The number of times a comment has been quoted.
  8) reply_count - The number of times a comment has been replied to.
  9) retweet_count - The number of times a comment has been retweeted.
  10) favorite_count - The number of times the comment has been favorited.
  11) hashtags - Represents hashtags parsed out of a Tweet text.
  12) extended_tweet - A subfield that contains additional information about

a comment's content when it exceeds the 140-character limit.

13) is_reply_to_status_id_str - The unique identifier of the tweet that a comment is replying to.
14) is_reply_to_user_id_str - The unique identifier of the user that the comment is replying to.
15) is_reply_to_screen_name - The username of the user that the comment is replying to.