SJSU ChatGPT Tweets Analysis

DATA 225: LAB 2

Anusmriti Sikdar
Data Analytics
SJSU
San Jose, USA
anusmriti.sikdar@sjsu.edu

Akanksha Tyagi
Data Analytics
SJSU
San Jose, USA
akanksha.tyagi@sjsu.edu

Anjali Mishra
Data Analytics
SJSU
San Jose, USA
anjalihimanshu.ojha@sjsu.edu

Priyanka Bhyregowda
Data Analytics
SJSU
San Jose, USA
priyanka.bhyregowda@sjsu.edu

Sahana Thoravalli Prabhuswamy
Data Analytics
SJSU
San Jose, USA
sahana.thoravalliprabhuswamy@sjsu.edu

***Abstract*—Recently, ChatGPT has been one of the most discussed topics. Twitter is a popular social media platform for people to connect and share their thoughts. Analyzing Twitter data provides deep insights into the way people think. By analyzing Twitter data on ChatGPT we can obtain valuable information. Hence, there is a need for a ChatGPT tweet analyzer application. In this paper, we're going to analyse the ChatGPT dataset using MongoDB, which is a NoSQL database. NoSQL database provides many advantages like high scalability, flexibility, performance, high availability, dost-effectiveness, and drive towards big data. The data is stored in MongoDB in the form of documents, which are similar to JSON objects. Manipulating and storing data in its natural form makes this database a simple choice, without requiring complex normalization and mapping operations.**

***Index Terms*—ChatGPT, Atlas, MongoDB**

I. Introduction

One of the most discussed revolutionary innovations in recent times in the field of Artificial Intelligence and Machine learning is the introduction of ChatGPT which is Large Language Model (LLM). People have shared their ChatGPT usage experience on social media platforms like twitter, facebook etc. In this project, we analyze the conversations involving ChatGPT using Twitter Analysis dataset from Kaggle [1]. The main purpose of this analysis is to provide insights into SJSU community's feedback and experiences with the language model using popularly used hashtags, most discussed tweets, etc, This also gives insight into users' emotions and opinions. We build an application to perform the analysis. The application system will be built using MongoDB as the NoSQL database to store and retrieve the data. The ChatGPT analyzer app can be run on any machine with a command line interface, python, and access to the internet, which will be available in most of the systems. Twitter Database in MongoDB has a reduced number of Tables or collections when compared to RDBMS, because of normalization. Even though there is data redundancy caused due to data denormalization, the performance is significantly improved.

In section II, we describe the functional requirement of the application.

II. Functional Requirement Analysis

A. Functionalites

SJSU ChatGPT Analyzer application has a client-server architecture as shown in figure 1. The Analyzer app acts as a client to the MongoDB Atlas cluster which acts as a server. A cluster in MongoDB Atlas provides a cloud-based database environment that caters to the data need of our application. The raw data which is in the form of a .csv file is extracted using a Python program and then cleaned and transformed into the required format and then loaded into the `twitter` database in the MongoDB Atlas cluster. The application runs on a command line interface and the user of our application can select options to view a few of the interesting metrics that can be obtained from the application. The developer can connect to MongoDB Atlas cluster endpoints either through a Mongo shell client or MongoDB compass or through the Python application. Only a DB admin can connect to the MongoDB Atlas and is in control of access privileges and performance tuning. Our application provides information about the following -

- Users most discussing about ChatGPT.
- Most viral tweet.
- Most used hastags.
- Most Discussed tweet with respect to replies and conversation.
- Most used source.
- Most active users.
- Number of tweets each day.
- Most used language.
- Most used media.

- Most mentioned users.
- Most liked tweets about ChatGPT.
- Users with most number of retweets.
- Most commonly used words in tweets about ChatGPT.

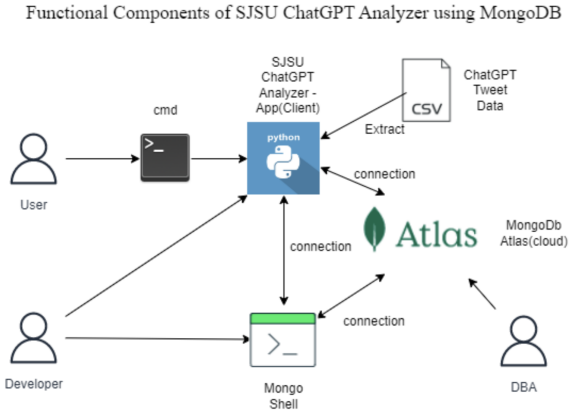

Fig. 1. Application design.

B. Limitations

Our application has following limitations -

- It provides a fixed number of metrics.
- The metrics are based on a small set of data. So they may not provide full insight into the real metrics.

III. Database Design

MongoDB is a NoSQL document-oriented database that stores information in flexible BSON documents. To achieve a well-structured and efficient database it is important to identify and model the relationship between data collections, define their attributes, and identify the appropriate schema design for storage in the database.

To design the schema for the database in MongoDB, it is important to understand the structure of data and requirements of the application. The dataset consists of a collection of tweets that have been scraped from Twitter consisting of the hashtag #chatgpt. People have been tweeting about this language model, sharing their experiences with it, and talking about articles or resources related to ChatGPT. Based on the requirements defined in section II-A, the dataset in split into various collections. The collections are chosen such that they can exist and be accessed independently. The following sections describe each collection in detail.

A. Collections

1) Users: The collection `users` stores Twitter users related information. Each document in this collection represents a single user and has a `username`, `userUrl` and `userInfo` fields. An example document in the collection is shown below.


```
{
  "_id": {
    "$oid": "64437059ddd851158e3777c7"
  },
  "username": "ciffi",
  "userUrl": "https://twitter.com/ciffi",
  "userInfo": "username=ciffi,id=19856336,
       displayname=christianfüller..."
}
```


2) User Mentions: The collection `user_mentions` stores data about other Twitter users who are mentioned in the tweets. Each document in the collection represents a single mention and has `tweetid`, `username`, `userid` and `displayname`. An example document in the collection is shown below.


```
{
  "_id": {
    "$oid": "64437090ddd851158e39baee"
  },
  "tweetId": {
    "$numberLong": "1617156291046133761"
  },
  "username": "AlexandrovnaIng",
  "userid": "2827059006",
  "displayname": "Alexandrovna"
}
```


3) Tweets: The collection `tweets` is the main collection of the entire database design. It stores details of all the information related to a tweet. Each document in this collection represents a single tweet posted by a user and has `tweetId`, `tweetText`, `username`, `createdTime`, `lang`, `conversationId`, `replyCount`, `retweetCount`, `likeCount`, `quoteCount`, `hashtagCounts`, `Used_source` and an embedded document `urls` containing `permalinkUrl`, `outlink`, `countLink`, and `quotedTweet`. An example document in the collection is shown below.


```
{
  "_id": {
    "$oid": "64437068ddd851158e3832f7"
  },
  "tweetId": {
    "$numberLong": "1617156291046133761"
  },
  "tweetText": "@AlexandrovnaIng Prohibition
       of ChatGPT has been added to the honor
       code of my daughter's school",
  "username": "Caput_LupinumSG",
  "createdTime": {
    "$date": "2023-01-22T13:44:39.000Z"
  },
  "lang": "en",
  "conversationId": {
    "$numberLong": "1617148639993806848"
  },
  "replyCount": 1,
  "retweetCount": 0,
  "likeCount": 5,
  "quoteCount": 0,
  "hastagCounts": 0,
  "Used_source": "Twitter for iPhone",
  "urls": {
    "permalinkUrl":
        "https://twitter.com/Caput_LupinumSG..",
    "outLink": "",
    "countLink": "",
    "quotedTweet": ""
  }
```

```
}
```

4) Hastags: The collection `hastags` stores data about the used hashtags by the user in each tweet. Each document in this collection has a `tweetid` and `hastag`. An example document of this collection is shown below.

```
{
  "_id": {
    "$oid": "64437084ddd851158e38f647"
  },
  "tweetId": {
    "$numberLong": "1617156308926349312"
  },
  "hashtag": "#ChatGPT"
}
```

5) Media: The collection `media` stores data about other Twitter users who are mentioned in the tweets. Each document in the collection represents a single mention and has corresponding `tweetId`, `mediaType`, `mediaDetails`.

```
{
  "_id": {
    "$oid": "6443708cddd851158e398f54"
  },
  "tweetId": {
    "$numberLong": "1617156308926349312"
  },
  "mediaType": "Photo",
  "mediaDetails": "Photo(
    previewUrl='https://pbs.twimg.com/media/...')"
}
```

B. Denormalization

It is always important to consider how often the data changes versus how often you read data from the database. Normalization makes inserting efficient and it is easier to maintain data integrity whereas denormalization makes data reading efficient. A fully de-normalized database might achieve higher read performance but takes longer to write because every field should be updated in a document if one of the fields must be changed. Fields that do not change too frequently can be embedded and inserted into documents. If the document is too small, does not need to change regularly, grows by a small amount, needs faster reads, and does not require immediate consistency, we use the concept of embedding the document. On the other hand, if there are big subdocuments, data changes frequently, needs up-to-date information, grows by a large amount, and faster writes then, we must use the concept of referencing. In our design, the de-normalized Data Structure of the collection `tweets` has an embedded document field `urls` which is shown below.

```
"urls": {
    "permalinkUrl": "https://twitter.com/
        mochico0123/status/...",
    "outLink": "",
    "countLink": "",
    "quotedTweet": ""
}
```

We can also have an array of objects for `media_info`, `mentioned_users` and `hashtags` in tweets which uses the concept of referencing. An example is shown in figure 2.

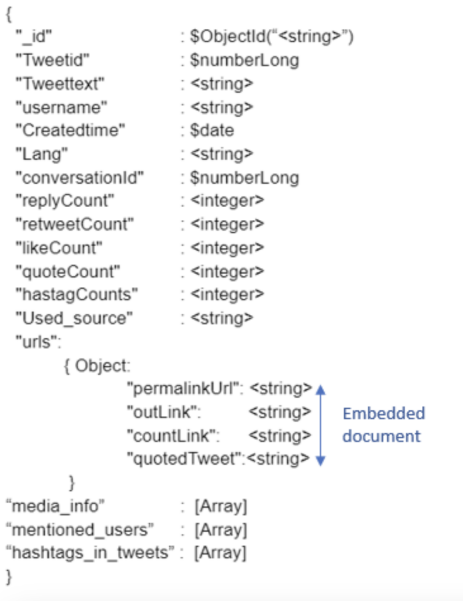

Fig. 2. Denormalization example.

IV. Connection to MongoDB atlas.

We use `pymongo` to connect to MongoDB atlas and upload all the collections there. The uploaded collections are shown in figure 3.

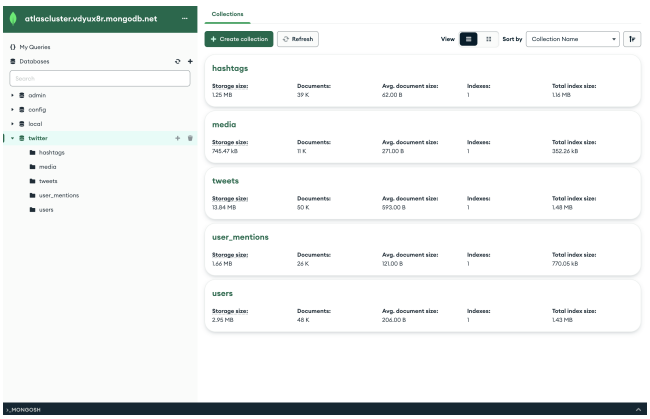

Fig. 3. Connection to MongoDB atlas.

V. Queries

In this section, we run queries for each requirement described in section II-A.

A. Number of tweets each day

The following query retrieves the number of tweets per day from `tweets` collection. The query uses the aggregation framework to group the tweets by day, using the `$dateToString` operator to extract the date from the `createdTime` field of each tweet and format it as "YYYY-MM-DD". Then, it calculates the number of tweets per day using the `$sum` operator.

```
db.tweets.aggregate([
  {
    "$group": {
      "_id": {
        "$dateToString": {
          "format": "%Y-%m-%d",
          "date": "$createdTime"
        }
      },
      "tweet_count": {
        "$sum": 1
      }
    }
  },
  {
    "$project": {
      "_id": 0,
      "tweeted_date": "$_id",
      "tweet_count": 1
    }
  },
  {
    "$sort": {
      "tweeted_date": 1
    }
  }
])
```

The results of the query are shown in figure 4.

Fig. 4. Number of tweets each day.

B. Most active users

The following query returns a list of active users along with their total tweet count, sorted by the number of tweets in descending order. It can help identify the most active users and help to better understand the user behavior and engagement on the platform.

```
db.tweets.aggregate([
  {
    "$group": {
      "_id": "$username",
      "tweet_count": {
        "$sum": 1
      }
    }
  },
  {
    "$project": {
      "_id": 0,
      "active_user": "$_id",
      "tweet_count": 1
    }
  },
  {
    "$sort": {
      "tweet_count": -1
    }
  },
  {
    "$limit": 10
  }
])
```

The results of the query are shown in figure 5.

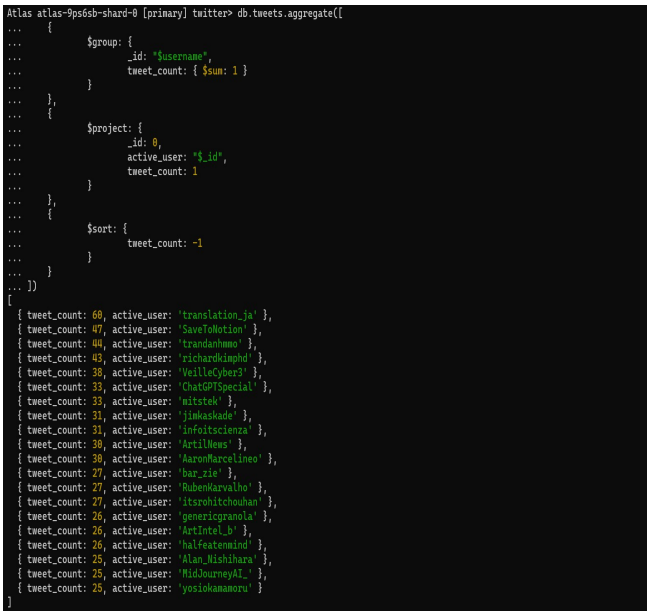

Fig. 5. Most active users.

C. Most used hashtags

The following query retrieves the 10 most used hashtags from the `hashtags` collection by grouping the tweets by `hashtag` field and using the `$sum` operator to calculate the number of times each hashtag appears in the collection. The `$sort` stage sorts the output in descending order based on the `hashtag_count` field.

```
db.hashtags.aggregate([
  {
    "$group": {
      "_id": "$hashtag",
      "hashtag_count": {
        "$sum": 1
      }
    }
  },
  {
    "$sort": {
      "hashtag_count": -1
    }
  },
  {
    "$limit": 10
  }
])
```

The results of the query are shown in figure 6.

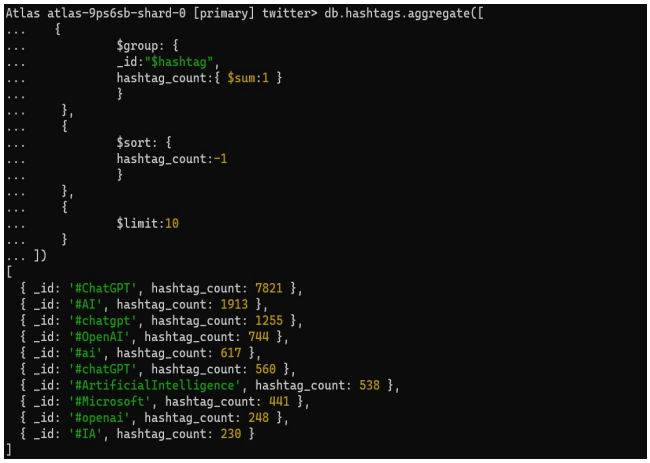

Fig. 6. Most used hashtags.

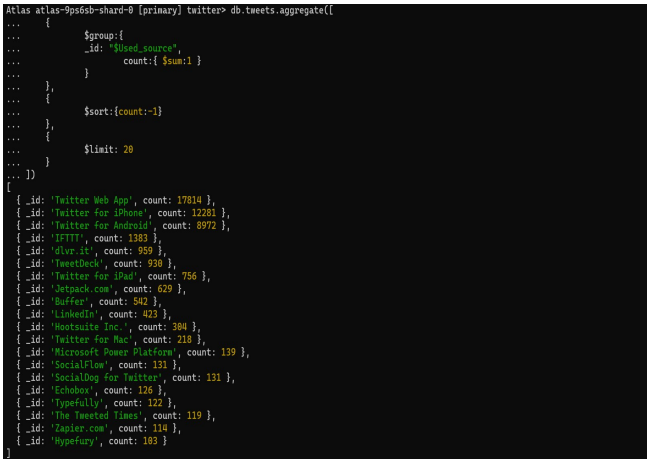

Fig. 7. Most used source.

D. Most used source

The following query retrieves the top 10 most used sources from the tweet collection by grouping the tweets by the `Used_source` field and using the `$sum` operator to calculate the number of times each source appears in the collection. The `$sort` stage sorts the output in descending order based on the `count` field.

```
db.tweets.aggregate([
  {
    "$group": {
      "_id": "$Used_source",
      "count": {
        "$sum": 1
      }
    }
  },
  {
    "$sort": {
      "count": -1
    }
  },
  {
    "$limit": 10
  }
])
```

The results of the query are shown in figure 7.

E. Most used language

The following query returns a list of languages used in the tweets collection along with the total number of tweets in each language, sorted by the number of tweets in descending order. It can help to identify the most common languages used on the platform and better understand the language preferences of the users.

```
db.tweets.aggregate([
  {
    "$group": {
      "_id": "$lang",
      "totalNumberOfTweets": {
        "$sum": 1
      }
    }
  },
  {
    "$sort": {
      "totalNumberOfTweets": -1
    }
  },
  {
    "$limit": 10
  }
])
```

The results of the query are shown in figure 8.

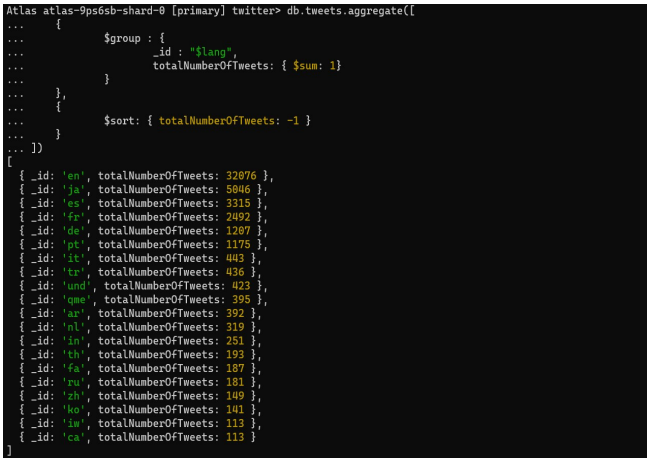

Fig. 8. Most used language.

F. Most used media

The following query returns a list of media types used in the collection along with the number of times each type was used, sorted by the number of times used in descending order. It can help to identify the most popular media types on the platform and to better understand the preferences of the users.

```
db.media.aggregate([
  {
    "$group": {
      "_id": "$mediaType",
      "NumberOfTimesUsed": {
        "$sum": 1
```

```
      }
    }
  },
  {
    "$sort": {
      "NumberOfTimesUsed": -1
    }
  },
  {
    "$limit": 10
  }
])
```

The results of the query are shown in figure 9.

```
Atlas atlas-9ps6sb-shard-0 [primary] twitter> db.media.aggregate([
...     {
...             $group :{
...             _id : "$mediaType",
...             NumberOfTimesUsed: { $sum: 1}
...             }
...      },
...      {
...             $sort: { NumberOfTimesUsed: -1 }
...      }
... ])
[
  { _id: 'Photo', NumberOfTimesUsed: 10233 },
  { _id: 'Video', NumberOfTimesUsed: 522 },
  { _id: 'Gif', NumberOfTimesUsed: 407 }
]
```

Fig. 9. Most used media.

G. Most liked tweets about ChatGPT

The following query will first filter the tweets to only include those containing "chatgpt" Then, it will sort the tweets in descending order by `likeCount` and limit the results to the first document.

```
db.tweets.aggregate([
  {
    $match: {
      tweetText: /chatgpt/i
     }
  },
  {
    "$sort": {
      "likeCount": -1
    }
  },
  {
    "$limit": 1
  }
])
```

The results of the query are shown in figure 10.

H. Top 10 users who have tweeted the most about #chatgpt.

This query retrieves the top 10 users who have used the `#chatgpt` hashtag in their tweets the most number of times.

```
db.tweets.aggregate([
  {
    "$match": {
      tweetText: { "$regex": /#chatgpt/i }
}
  },
  {
    "$group": {
      "_id": "$username",
      "count": {
        "$sum": 1
      }
```

```
Atlas atlas-9ps6sb-shard-0 [primary] twitter> db.tweets.aggregate([
...     {
...             $match: {
...             tweetText: /chatgpt/i
...             }
...     },
...     {
...             $sort: {
...                     likeCount: -1
...             }
...     },
...     {
...             $limit: 1
...     }
... ])
[
  {
    _id: ObjectId("64437068ddd851158e383485"),
    tweetId: Long("1617162355112124421"),
    tweetText: 'ChatGPT passed a Wharton MBA exam. \n\nTime to overhaul education.',
    username: 'GRDecter_',
    createdTime: ISODate("2023-01-22T14:08:45.000Z"),
    lang: 'en',
    conversationId: Long("1617162355112124421"),
    replyCount: 1421,
    retweetCount: 6815,
    likeCount: 56073,
    quoteCount: 1947,
    hastagCounts: 0,
    used_source: 'Twitter for iPhone',
    urls: {
      permalinkUrl: 'https://twitter.com/GRDecter/status/1617162355112124421',
      outLink: '',
      countLink: '',
      quotedTweet: ''
    }
  }
]
```

Fig. 10. Most liked tweets about ChatGPT.

```
      }
    },
    {
      "$sort": {
        "count": -1
      }
    },
    {
      "$limit": 10
    }
])
```

The results of the query are shown in figure 11.

```
Atlas atlas-9ps6sb-shard-0 [primary] twitter> db.tweets.aggregate([
...     {
...             $match: {
...             tweetText: { $regex: /#chatgp/i }
...             }
...     },
...     {
...             $group: {
...             _id: "$username",
...             count: { $sum: 1 }
...             }
...     },
...     {
...             $sort: { count: -1 }
...     },
...     {
...             $limit: 10
...     }
... ])
[
  { _id: 'translation_ja', count: 60 },
  { _id: 'richardkimphd', count: 43 },
  { _id: 'VeilleCyber3', count: 38 },
  { _id: 'ArtilNews', count: 30 },
  { _id: 'MidJourneyAI_', count: 25 },
  { _id: 'jhengstler', count: 21 },
  { _id: 'ChatGPT_Tweeter', count: 21 },
  { _id: 'abushoyeb', count: 20 },
  { _id: 'bar_zie', count: 19 },
  { _id: 'PriorityDomains', count: 19 }
]
```

Fig. 11. Top 10 users who have tweeted the most about #chatgpt.

I. Top 10 users who have the highest number of retweets in their tweets containing #chatgpt

This query retrieves the top 10 users who have tweeted with the `#chatgpt` hashtag and have received the highest total number of retweets for their tweets.

```
db.tweets.aggregate([
  {
    $match: { tweetText: { $regex: /#chatgpt/i } }
  },
  {
```

```
    "$group": {
      "_id": "$username",
      "totalRetweets": {
        "$sum": "$retweetCount"
      }
    }
  },
  {
    "$sort": {
      "totalRetweets": -1
    }
  },
  {
    "$limit": 10
  }
])
```

The results of the query are shown in figure 12.

```
Atlas atlas-9ps6sb-shard-0 [primary] twitter> db.tweets.aggregate([
...   {
...     $match: {
...       tweetText: { $regex: /#chatgp/i }
...     }
...   },
...   {
...     $group: {
...       _id: "$username",
...       totalRetweets: { $sum: "$retweetCount" }
...     }
...   },
...   {
...     $sort: { totalRetweets: -1 }
...   },
...   {
...     $limit: 10
...   }
... ])
[
  { _id: 'DataChaz', totalRetweets: 331 },
  { _id: 'kyle_chasse', totalRetweets: 222 },
  { _id: 'Schuldensuehner', totalRetweets: 130 },
  { _id: 'StevenXGe', totalRetweets: 125 },
  { _id: 'CerfiaFR', totalRetweets: 107 },
  { _id: 'VeilleCyber3', totalRetweets: 102 },
  { _id: 'healthy_pockets', totalRetweets: 93 },
  { _id: 'ramkumarcomic', totalRetweets: 88 },
  { _id: 'lgoodtern', totalRetweets: 83 },
  { _id: 'PriorityDomains', totalRetweets: 82 }
]
```

Fig. 12. Top 10 users who have the highest number of retweets in their tweets containing #chatgpt.

J. Most viral tweet

This query finds tweets that contain the hashtag `#chatgpt` and then calculates the ”viral count” by adding up the number of replies, retweets, likes, and quotes for each tweet. Then, it sorts the tweets in descending order based on their viral count and returns the tweet with the highest viral count, which is considered the most viral tweet about chatGPT.

```
db.tweets.aggregate([
  {
    $match: { tweetText: { $regex: /chatgpt/i } }
  },
  {
    "$addFields": {
      "viral_count": {
        "$add": [
          "$replyCount",
          "$retweetCount",
          "$likeCount",
          "$quoteCount"
        ]
      }
    }
  },
  {
    "$sort": {
      "viral_count": -1
    }
  },
  {
    "$limit": 1
  }
])
```

The results of the query are shown in figure 13.

```
Atlas atlas-9ps6sb-shard-0 [primary] twitter> db.tweets.aggregate([
...   {
...     $match: {
...       tweetText: { $regex: /#chatgpt/i }
...     }
...   },
...   {
...     $addFields: {
...       viralcount: {
...         $add: ["$replyCount", "$retweetCount", "$likeCount", "$quoteCount"]
...       }
...     }
...   },
...   {
...     $sort: { viralcount: -1 }
...   },
...   {
...     $limit: 1
...   }
... ])
[
  {
    _id: ObjectId('64437068ddd8511158e38aac0'),
    tweetId: Long('1617565447670464204 7'),
    tweetText: '🚨 FLASH | Quelques jours après avoir supprimé 10 000 emplois, #Microsoft va investir plusieurs milliards de dollars dans OpenAI, la maison-mère de #ChatGPT.',
    username: 'CerfiaFR',
    createdTime: ISODate('2023-01-23T16:50:30.000Z'),
    lang: 'fr',
    conversationId: Long('1617565447670464204 7'),
    replyCount: 29,
    retweetCount: 107,
    likeCount: 1199,
    quoteCount: 21,
    hastagCounts: 2,
    Used_source: 'TweetDeck',
    urls: {
      permalinkUrl: 'https://twitter.com/CerfiaFR/status/1617565447670464204 7',
      outLink: '',
      countLink: '',
      quotedTweet: ''
    },
    viralcount: 1356
  }
]
```

Fig. 13. Most viral tweet.

K. Most commonly used words in tweets about ChatGPT

The following query searches for tweets containing the word ”chatgpt” (case-insensitive) in their text. Then it splits each tweet’s text into individual words and groups them by word, counting the number of occurrences of each word. The resulting list is sorted in descending order of frequency and limited to the top 10 most used words. This can provide insights into the topics and themes associated with the use of the hashtag `#chatgpt` on Twitter.

```
db.tweets.aggregate([
  {
    $match: { tweetText: { $regex: /chatgpt/i } }
  },
  {
    "$project": {
      "words": {
        "$split": [
          "$tweetText",
          " "
        ]
      }
    }
  },
  {
    "$unwind": "$words"
  },
  {
    "$group": {
      "_id": "$words",
      "count": {
        "$sum": 1
      }
    }
  },
  {
    "$sort": {
      "count": -1
    }
  },
```

```
  {
    "$limit": 10
  }
])
```

The results of the query are shown in figure 14.

Fig. 14. Most commonly used words in tweets about ChatGPT.

L. Most Discussed tweet with respect to replies and conversation

The following query helps to identify the most active conversations in the tweets collection. The `$match` filters the tweets to include only those with a non-null `conversationId`. The `$group` groups the tweets by `conversationId` and calculates the total number of replies for each conversation using `$sum` operator. Then, the conversations are sorted by the total number of replies in descending order using -1.

```
db.tweets.aggregate([
  {
    "$match": {
      "conversationId": {
        "$ne": null
      }
    }
  },
  {
    "$group": {
      "_id": "$conversationId",
      "count": {
        "$sum": "$replyCount"
      }
    }
  },
  {
    "$sort": {
      "count": -1
    }
  },
  {
    "$limit": 1
  }
])
```

The results of the query are shown in figure 15.

Fig. 15. Most Discussed tweet with respect to replies and conversation.

M. Most mentioned users

The following query will return the top 10 users who have been mentioned the most in the tweets, along with the count of how many times each user has been mentioned.

```
db.user_mentions.aggregate([
  {
    "$group": {
      "_id": "$username",
      "count": {
        "$sum": 1
      }
    }
  },
  {
    "$sort": {
      "count": -1
    }
  },
  {
    "$limit": 10
  }
])
```

The results of the query are shown in figure 16.

Fig. 16. Most mentioned users.

VI. ChatGPT Application

We have also implemented an interactive python app. Users can select from the available options which are as discussed in the functional requirements to get the necessary metrics as shown in figure 17, 18, 19 and 20.

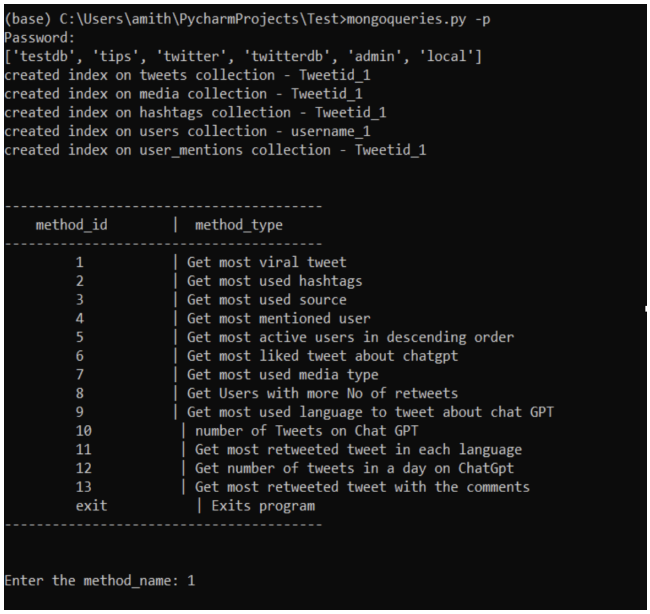

Fig. 17. App example 1.

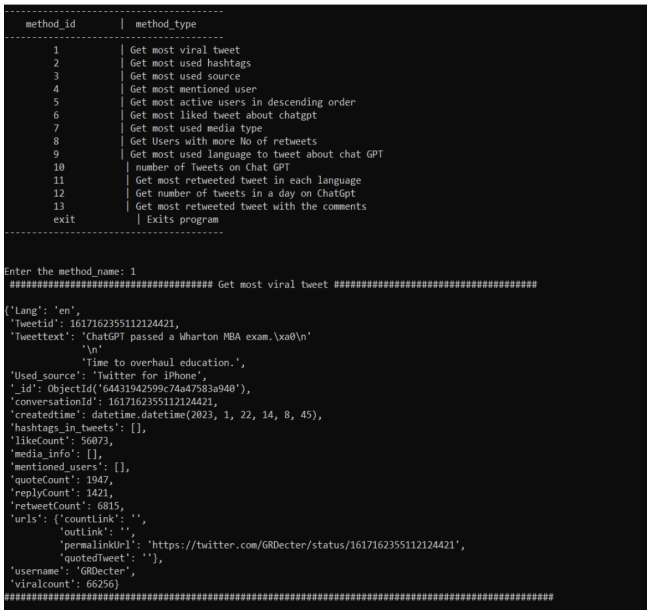

Fig. 18. App example 2.

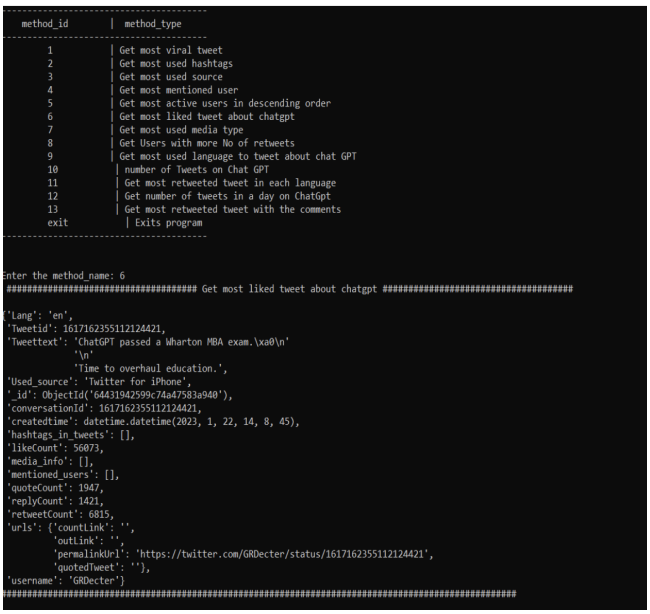

Fig. 19. App example 3.

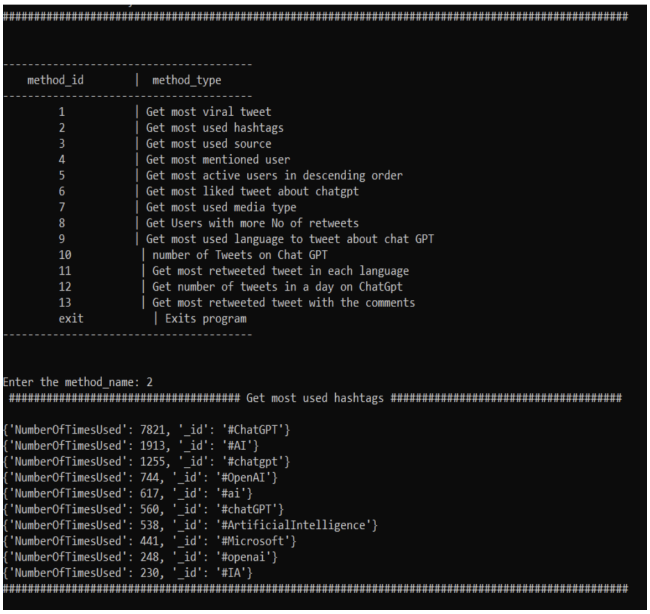

Fig. 20. App example 4.

VII. Data visualization

We have visualized the tweet data in MongoDB atlas. The chart is shown in Figure 21. The chart can also accessed at the following public url: https://charts.mongodb.com/charts-lab-2---chatgpt-tweet-ana-ifsvx/public/dashboards/64433f00-8295-4353-892c-04104f0e8333.

VIII. Performance analysis

We use `explain("executionStats")` method to analyse the performance of each query. The `explain` method results present the query plans as a tree of stages. figure

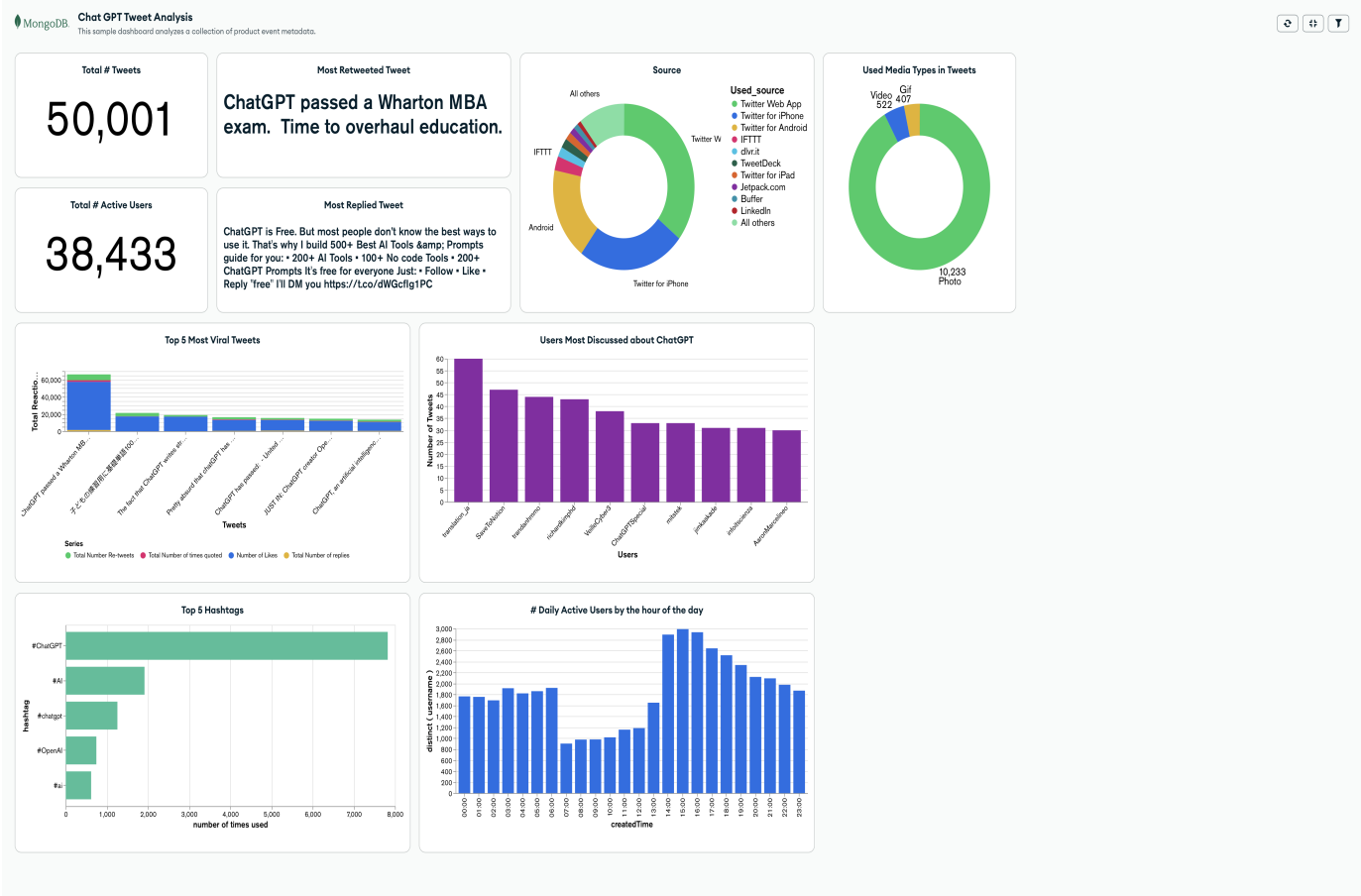

Fig. 21. Data visualization using MongoDb atlas.

23 shows the comparison of the execution times of all the queries. The detailed analysis of each stage is provided in the jupyter notebook. We see that the queries to get the most commonly used words takes the most amount of time. Detailed analysis shows that MongoDB uses `COLLSCAN` to scan all the documents in the collection, which results in high execution time. The execution stats are shown below.


```
{
  "executionStats": {
    "executionStages": {
      "advanced": 46546,
      "executionTimeMillisEstimate": 65,
      "inputStage": {
        "advanced": 46546,
        "direction": "forward",
        "docsExamined": 50001,
        "executionTimeMillisEstimate": 35,
        "filter": {
          "tweetText": {
            "$options": "i",
            "$regex": "chatgpt"
          }
        },
        "isEOF": 1,
        "nReturned": 46546,
        "needTime": 3456,
        "needYield": 0,
        "restoreState": 60,
        "saveState": 60,
        "stage": "COLLSCAN",
        "works": 50003
      },
      "isEOF": 1,
      "nReturned": 46546,
      "needTime": 3456,
      "needYield": 0,
      "restoreState": 60,
      "saveState": 60,
      "stage": "PROJECTION_DEFAULT",
      "transformBy": {
        "_id": true,
        "words": {
          "$split": [
            "$tweetText",
            {
              "$const": " "
            }
          ]
        }
      },
      "works": 50003
    },
    "executionSuccess": true,
    "executionTimeMillis": 878,
    "nReturned": 46546,
    "totalDocsExamined": 50001,
    "totalKeysExamined": 0
```

```
  },
  "queryPlanner": {
    "indexFilterSet": false,
    "maxIndexedAndSolutionsReached": false,
    "maxIndexedOrSolutionsReached": false,
    "maxScansToExplodeReached": false,
    "namespace": "twitter.tweets",
    "parsedQuery": {
      "tweetText": {
        "$options": "i",
        "$regex": "chatgpt"
      }
    },
    "planCacheKey": "5B027F7E",
    "queryHash": "5B027F7E",
    "rejectedPlans": [],
    "winningPlan": {
      "inputStage": {
        "direction": "forward",
        "filter": {
          "tweetText": {
            "$options": "i",
            "$regex": "chatgpt"
          }
        },
        "stage": "COLLSCAN"
      },
      "stage": "PROJECTION_DEFAULT",
      "transformBy": {
        "_id": true,
        "words": {
          "$split": [
            "$tweetText",
            {
              "$const": " "
            }
          ]
        }
      }
    }
  }
}
```


In figure 22, we should execution times from the server. The spike corresponds to the query. Figure 27, 25, 26, 27, 28 and 29 show various system metrics.

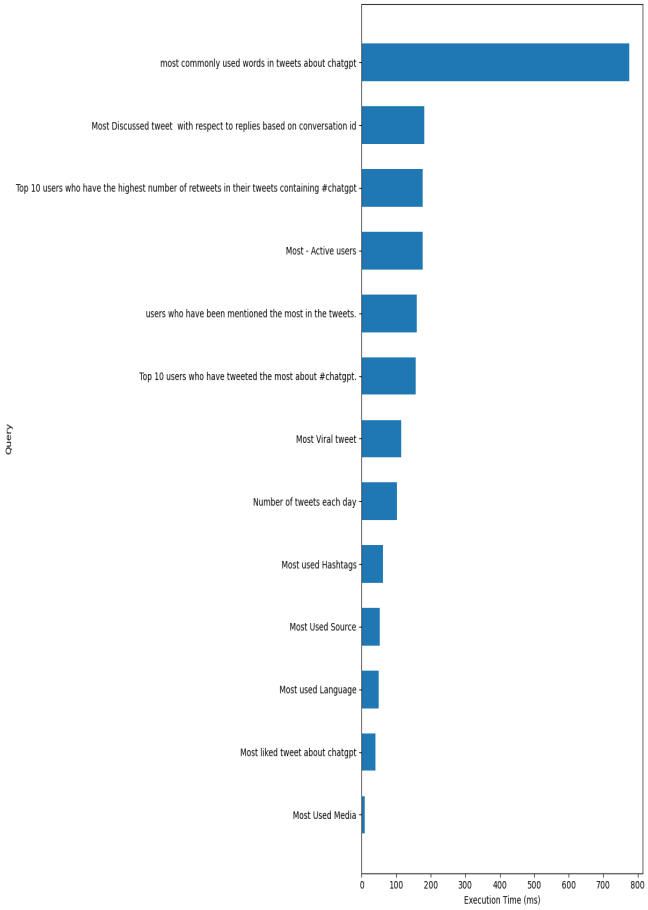

Fig. 23. Timing analysis of each query.

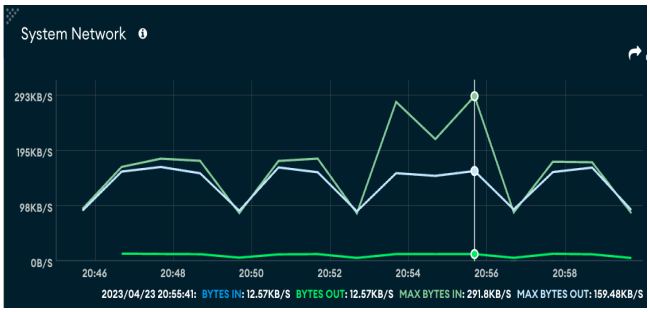

Fig. 24. System network usage.

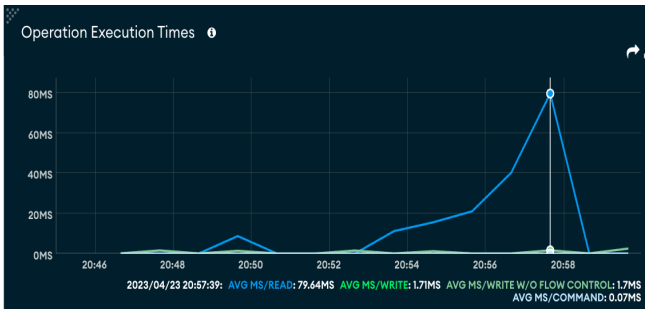

Fig. 22. Execution times.

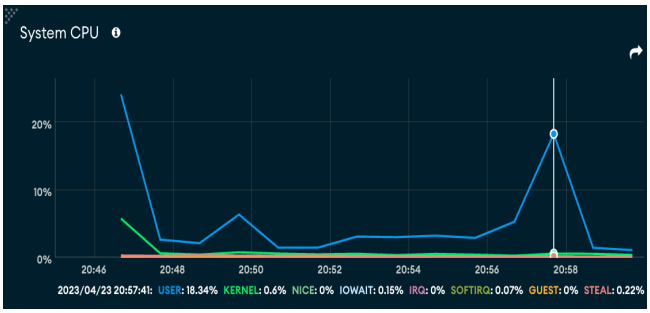

Fig. 25. CPU usage.

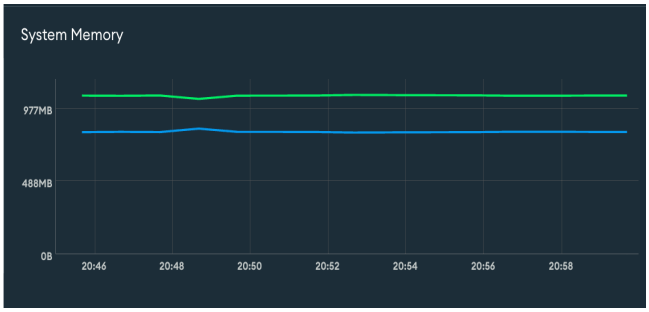

Fig. 26. Memory usage.

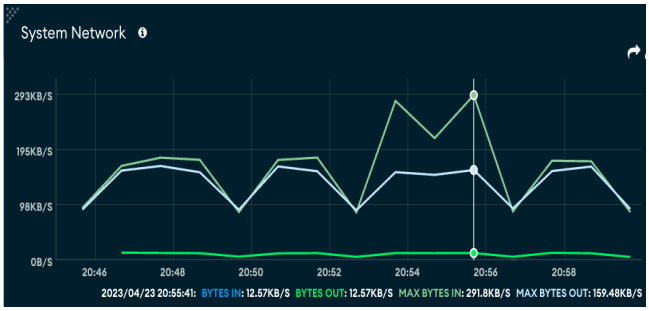

Fig. 27. Network usage.

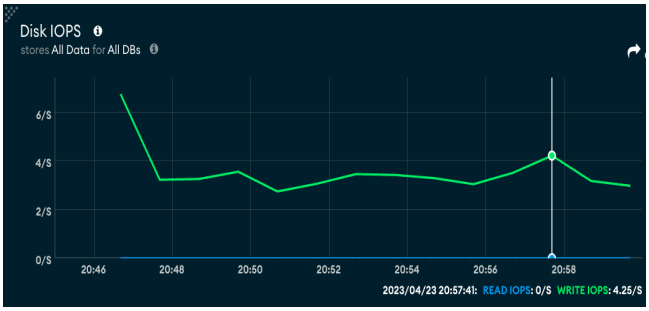

Fig. 28. Disk IO usage.

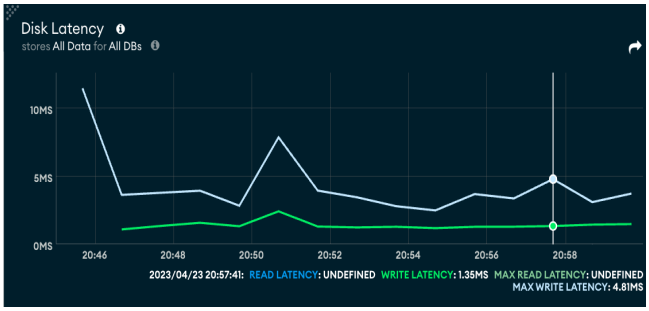

Fig. 29. Disk latency.

A. Comparison with MySQL

In figure 30, we compare the run times of each query on both MySQL and MongoDB. We find that most of the queries run faster on MongoDB. However, we must take into account the factor that these queries were run on different clusters.

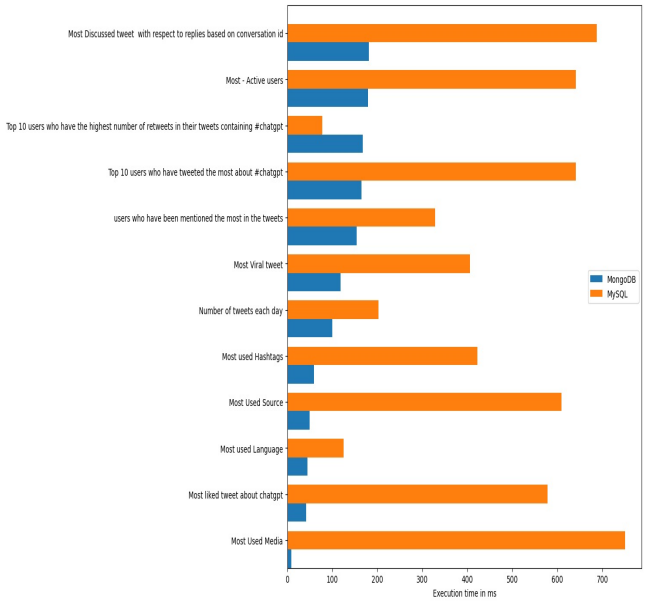

Fig. 30. Comparison of each queries on MySQL and MongoDB.

REFERENCES

[1] ChatGPT Twitter Dataset: https://www.kaggle.com/datasets/tariqsays/chatgpt-twitter-dataset.
[2] MongoDB: https://www.mongodb.com/docs/manual/introduction/