Building a Comprehensive Twitter Database: Using MySQL and ElasticSearch for Searching and Analysis

Team 15
Yunhao Li, Siheng Huang, Yanhan Chen, Yicong Li

Department of Statistics,
Rutgers University

May, 2023
Have done the pre-presentation

Retweet

We think of a tweet and its retweets as a tree, and the existing data provides a way to find the parent node of any node. That's enough for us to get the complete retweet relationship. But we don't think the data structure is intuitive. When we want to find all the retweets of a tweet, that is, all the children nodes of any node, we need to traverse the entire data set. The time complexity of this behavior is O(n). This is expensive when the amount of data in the database is very large.

Retweet

So we added a key-value pair for each tweet in the original dictionary. The value is a list of all the retweets of the corresponding tweet. For example, A is A tweet, and B and C are A's retweets. So the value of A is the list containing ID of B and ID of C. If the tweet doesn't have a retweet, the value will be an empty list. This is shown in the following table.

ID	...(Other features)	Retweet
ID of A	...	[ID of B, ID of C]
ID of B	...	[ ]
ID of C	...	[ ]

Retweet

To complete the above operations, it only takes O(n) time to traverse the data set once when we load the data.

When we have a new tweet added to the database, it only takes O(1) time to add it to the corresponding list of the tweet that were retweeted.

Introduction

We used MySQL to create three relational database tables to store information about Twitter.

- Basic information and retweet relationship
- Users information
- Popularity score

Basic info & retweet

The first table contains the basic information of all the tweets, including its ID, user ID and created time. It also contains the ID of the tweet that it retweeted (if exists) and the ID of the tweets that retweeted it (if exists).

ID	User_ID	Created_time	Retweet	Retweeted
str	str	time	list	str

In this table, ID is the primary key.

User info

The second table contains the information of all the users, including user ID, user name, location, URL and user popularity.

User_ID	User_name	Location	URL	User_pop
str	str	str	str	int

In this table, User ID is the primary key, and it is also the foreign key of the first table.

Poplarity

We use quote count, reply count, retweet count and favourite count to evaluate whether a tweet is popular. We can actually use a more sophisticated algorithm to calculate popularity. But that wasn't the main goal of our project, so we simply added up the four numbers. The larger the sum, the more popular the tweet.

ID	Quote_count	Reply_count	Retweet_count	Fav_count	Sum
str	int	int	int	int	int

In this table, ID is the primary key, and it is also the foreign key of the first table.

ES v.s. relational database

Elasticsearch is a schema-less, JSON-based search engine optimized for full-text search and high availability, using a cluster-based architecture with horizontal scalability. In contrast, relational databases use a schema-based model with tables and SQL for data manipulation, offering ACID transactions for data consistency.

Elasticsearch relies on inverted indexes and its Query DSL, while relational databases use B-trees and SQL.

Ultimately, Elasticsearch excels in search-heavy use cases, while relational databases are preferred for structured data and transactional workloads.

ES datastore

First we connected the ElasticSearch server and set up a temporary node. We then convert the raw data into json format.

It is worth mentioning that we only store the ID and text of the tweet in ES. We don't store duplicate content in MySQL and ES for saving space.

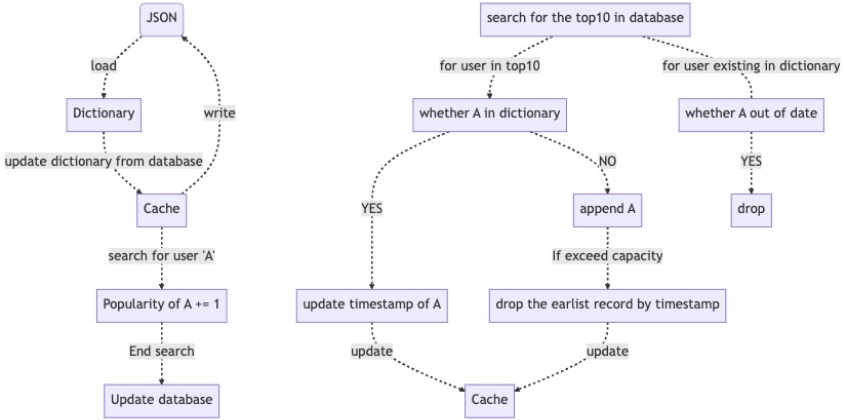
Search function

According to the characteristics of ES, we define two kinds of search functions for the implementation of subsequent search applications

The first is fuzzy search, which is helpful when we want to find documents with terms that are similar to the query term, accounting for typographical errors or slight variations. Elasticsearch uses the Damerau-Levenshtein distance method to calculate the similarity between terms.

The other is full-text search, which is designed to search for relevant documents across large volumes of text, considering various factors such as term frequency, document frequency, and the length of the document.

Flow chart


JSON
load
Dictionary
write
update dictionary from database
Cache
search for user 'A'
Popularity of A += 1
End search
Update database
search for the top10 in database
for user in top10
for user existing in dictionary
whether A in dictionary
whether A out of date
NO
YES
YES
append A
drop
If exceed capacity
update timestamp of A
drop the earlist record by timestamp
update
update
Cache

Initialization

We defined a cache to store the most popular user information. Formally this is a dictionary, and it's the same as the user information table in MySQL, including user ID, user name, location, URL, and popularity. In addition, a unique feature is included, time to live, which represents the user's timestamp.

We set the cache size to 20 and initialize the dictionary representing the cache. We also define a function to check the length of the dictionary, and if it is possible to exceed 20, we use a set of rules to exclude certain records.

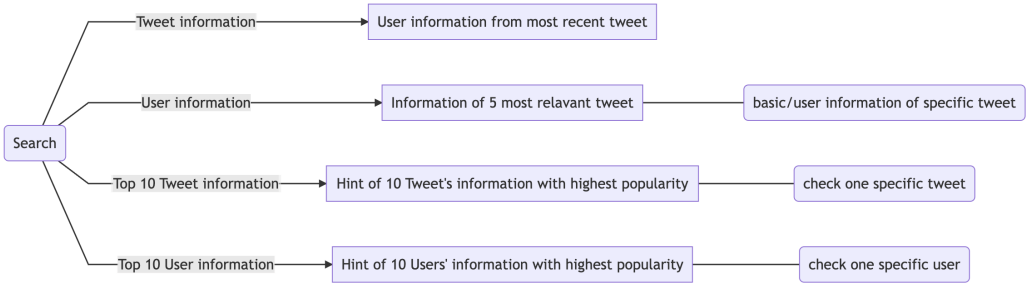
Update

Since our database was static, we calculated a user's popularity based on search operations only. When a user's information is searched, its corresponding user popularity increases by one. It's worth noting that we clear out the user popularity every day.

The cache is loaded each time the search application is started. When we first started the application, we added all the information of the 10 most popular users and time-stamps of the moment to the cache. With each subsequent update, if there is a new popular user, we will remove the record with the earliest timestamp. At the same time, we also update the timestamp of the records that already exist in the cache.

Finally, we set an 'out of date' rule. If the timestamp of a record in the cache is 3 days ago, we remove the record.

Introduction



Search for user

If you try to search for user information, the app looks in the Cache first. If no corresponding information is found, the system searches the table of user information.

If the search is successful, the app will provide basic information about the user and the five most recent tweets he or she has post. You can choose to continue to see the details of a particular tweet.

Search for user

```
Please select one of the following four options for searching

1. tweet
2. user
3. Top ten tweets
4. Top ten users

Please enter a single number: 2

Please enter the user you want to search: sara

+------+---------------------+------+----------+-----+------------------+
| S.No |       User ID       | Name | Location | URL | User Popularity |
+------+---------------------+------+----------+-----+------------------+
|  1   | 1000006582896295938 | sara |          |     |       13         |
+------+---------------------+------+----------+-----+------------------+

If you want to search for the tweets the user posted, you can enter the corresponding number before the user id
If you want to search for other informations, Please enter "C"
If you want to quit, please enter "Q"
```

Search for tweet

If you try to search for tweets, the application firstly checks the length of the search text, and if it's a single word, fuzzy search is preferred because it's more accurate. If it's a phrase or sentence, then full-text search is used.

ES automatically arranges search results in descending order of text relevance. You can choose to see the details of a particular tweet, including when it was posted, user information, list of retweets and so on. The application will then fetch this information from the basic information table in MySQL. It's worth noting that the list of retweets we created while working with the raw data comes into play here.

Search for tweet

```
Please select one of the following four options for searching

1. tweet
2. user
3. Top ten tweets
4. Top ten users

Please enter a single number: 1

Please enter the text you want to search: unemployment

0: ID: 1254039207498178563, Score: 11.221596, Text: Special Program For People Who Do Not Usually Qualify for Unemployment:

Did you get turned down for unemployment?… https://t.co/iSkaQE0wqs
1: ID: 1254047517983084545, Score: 11.221596, Text: RT @SleepyGirlGuide: Special Program For People Who Do Not Usually Qualify for Unemployment:

Did you get turned down for unemployment? Or…
2: ID: 1254048648113147905, Score: 11.221596, Text: RT @SleepyGirlGuide: Special Program For People Who Do Not Usually Qualify for Unemployment:

Did you get turned down for unemployment? Or…
3: ID: 1254055810637991938, Score: 10.902409, Text: @INCIndia Bas ek baar corona se jeet jaayein
Gdp,unemployment har cheez se jeet lenge
Because gdp and unemployment is not life threatening
4: ID: 1254052361779269634, Score: 9.4077215, Text: @ahsankhan568 @MalickViews @BPTWithMalick @FaheemYounus @humnewspakistan Agar corona s inflation unemployment
incre… https://t.co/vT4mjaeES2

If you want to know about the tweet, Please enter the number before ID
If you want to search for other informations, Please enter "C"
If you want to quit, please enter "Q"
```

Top 10 users and tweets

For the top10 most popular users and tweets, we will provide lists along with the corresponding drill down search.

The top10 users were obtained directly from cache, and the top10 tweets were obtained from the tweet popularity table.

Top 10 users

```
Please select one of the following four options for searching

1. tweet
2. user
3. Top ten tweets
4. Top ten users

Please enter a single number: 4
```

S.No	User ID	Name	Location	URL	User Popularity
1	7436623968922282881	twomad	Vancouver, British Columbia	https://www.youtube.com/threemad	38
2	349747452	Rindradana	The Way of The Strong	http://www.abuget.com	33
3	1000034375973646337	clarih 😻	Bonsucesso		28
4	1000026406112251905	Corona supplier banker	Hell		21
5	1000027886915637250	cheche			21
6	1000006582896295938	sara			13
7	104534482	Linda	Nebraska		13
8	100004211	azakiya tamilmagan	Chennai, India		11
9	29280466	Rick Aaron		https://twitter.com/search?q=from:RickAaron	9
10	21899941	linda	in my little slice of heaven	http://nutmeg237.blogspot.com	8

```
If you want to search for the tweets the user posted, you can enter the corresponding number before the user id
If you want to search for other informations, Please enter "C"
If you want to quit, please enter "Q"
```

Top 10 tweets

```
Please select one of the following four options for searching

1. tweet
2. user
3. Top ten tweets
4. Top ten users

Please enter a single number: 3

+-------+---------------------+------------------------------------------------------------------------------------------------------------------
-------+
| S.No |      Tweet ID       |                                              Text
       |
+-------+---------------------+------------------------------------------------------------------------------------------------------------------
-------+
|  1   | 1238264431320215553 |                                      *corona virus enters my body*
       |
|      |                     |
       |
|      |                     |                         The 4 Flintstone gummies I ate in 2005: https://t.co/3STfdIQtaT
       |
|  2   | 1240334979701395458 | When this Corona shit passes we have to promise each other that we're going to tell our kids that we survived a zombie apocalypse i
n 2020 |
|  3   | 1237436114887041024 |                THIS MAN IS A GENIUS he figured out the Corona virus problem 😮 https://t.co/EZP7IqTtxV
       |
|  4   | 1240066150278430720 |                 yeah it's a new generation we gon call them quaranteens https://t.co/rJSKmIf7Qp
       |
|  5   | 1239689136358985729 |                           "corona time "😭😭😭😭 https://t.co/iXBMHVcFoY
       |
|  6   | 1237982659760005120 |                                              Nobody:
       |
|      |                     |               Me and the homies on our $41 corona trip to the Bahamas: https://t.co/UOoRRJLiGr
       |
|  7   | 1243533984371523584 |                         Due to Corona, we officially have three days of the week
```

Thanks

Yunhao Li	yl1841@scarletmail.rutgers.edu
Siheng Huang	sh1641@scarletmail.rutgers.edu
Yicong Li	yl1851@scarletmail.rutgers.edu
Yanhan Chen	yc1241@scarletmail.rutgers.edu