Project Report

Data Analysis and Visualization of POI Checking-In Data

Xiao Zhenran

1 Introduction

In this project, I achieve a simple data analyzer. And it has five basic functions:

(1) Load the eligible data according to the parameters filled in the main window.

(2) Show a bar chart about 10 most popular locations for a specific user during a specific period.

(3) Show a line chart about the DAU of a specific location during a specific period.

(4) Compare two users' 10 most popular locations in one table.

(5) Compare two locations' DAU in one line chart.

And another extra function: show two users' track.

2 Implementation Details

2.1 Environment

I do all my coding and designing work on Qt IDE. The language I use is C++.

2.2 User Interface

I design a main window to catch the parameters for filter and load the data. Through the actions in the menu bar, the users can open five exclusive widgets for each function. There are 6 windows in total.

2.3 Data Storage

Firstly, I define a class named Record to store each record in the Gowalla.csv.

The Record class has five member variables to store each field of a record.

Name	Type
user_id	int
location_id	int
time	QDatetime
latitude	double
longitude	double

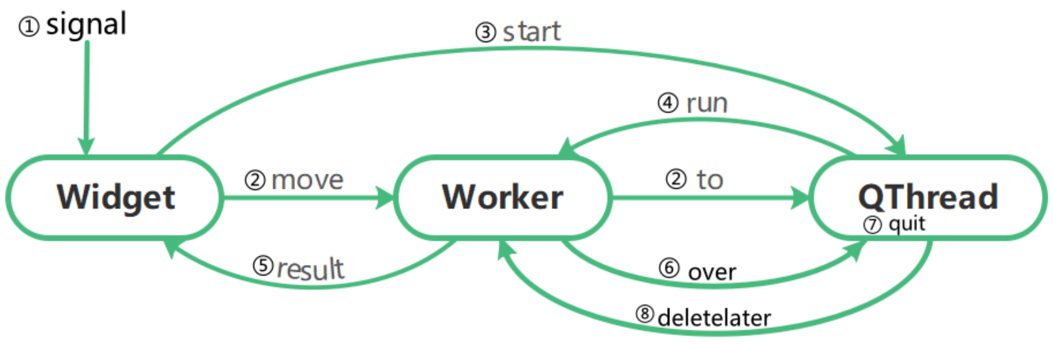
Next, I use a global object of type QHash to store the data loaded.

QHash<Key, T> is one of Qt's generic container classes. It stores (key, value) pairs and provides very fast lookup of the value associated with a key. QHash provides very similar functionality to QMap. QMap is a template class in Qt, which is a set of dictionaries based on the red-black tree algorithm. But QHash provides faster lookups than QMap.

I store the five fields in a Record. Then set it as the Hash table's value. Key is an int, the line number of the record in that csv file.

2.4 Multithread

I use multithread to separate the upper layer of displaying and the underlying layer of computing. To avoiding override run function. I write five worker classes, one for each function. And I use signal and slots to connect them:

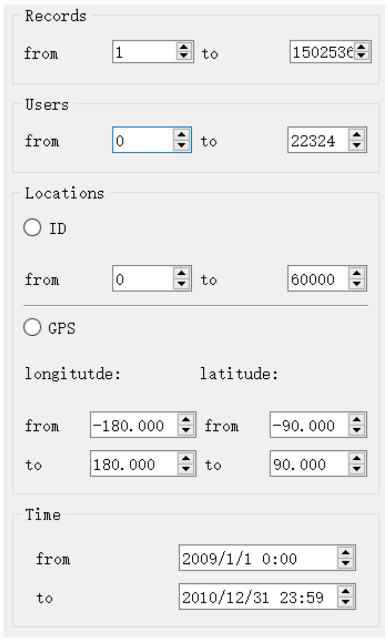


2.5 Data Loading

Users can tune the parameters in the main window to import the data field they are interested in. But there are several tips needing noticing:

(1) The two parameters in Records represent the line number. They determine the terms for loop. The display of the progress bar is also based on them.

(2) The two radio buttons determine how to filter the data about locations. If no one has been chosen, the default is ID.



Records
from 1 to 1502536

Users
from 0 to 22324

Locations
○ ID
from 0 to 60000
○ GPS
longitutde: latitude:
from -180.000 from -90.000
to 180.000 to 90.000

Time
from 2009/1/1 0:00
to 2010/12/31 23:59

2.6 Computing behind Functions

2.6.1 10 most popular locations for a user

To achieve this function, I need to know:

(1) which locations the user has been to during this period;

(2) how many times did he visit each location;

(3) what the top ten places visited most frequently are.

For (1) and (2), I need to find the record about this user during this period, take down the locations he visited and I need a counter to count the times.

I use QHash class to define a hash table to be as the counter. The key is int, to record the location's id, and the value is int as well to count the times. I choose it because if I insert all the locations' ids as key into the hash table, when I need to add the count of one location, I can quickly search it in the hash table through its id.

The counting process are divided into two parts:

(1) Prepare a counter. Traverse the loaded data to find out eligible records and insert each location's id as key and 0 time as value into the counter, a hash table.

(2) Count. Traverse the data again to count.

Then for (3), I need to find out ten most popular places, namely 10 key-value pairs with the biggest value. Because when iterating over a QHash, the items in the it are arbitrarily ordered. I need to bring them out to sort. But that is not intelligent. I can just move the item one by one to a min heap of which the size is 10. When the heap is full, if the next item's value is bigger than the top item's value of the min heap, just pop the top out and push the next item in.

Therefore, I use ten objects of type QPoint to receive each item in the hash table, an object of type QVector as the min heap based on Quicksort. I rewrite the Quicksort algorithm for QVector<QPoint> class in *globalvar.cpp*. Until the final vector is prepared, I set it as a parameter of a signal and emit the signal to the upper layer for visualization.

2.6.2 DAU

To achieve this function, I need to know:

how many times a day the location was visited during this period.

I need to find the record about this location during this period. And I a counter to count the times for each day.

I use QMap class to define this counter. The key is QDate, to record each day's date during this period. The value is int, to count the times. Why I choose QMap this time? Because when iterating over a QMap, the items are always sorted by key, in ascending order. It is perfect of QMap that after counting I don't need to process it more and it can be directly sent to the upper layer.

The counting process are divided into two parts:

(1) Prepare a counter. Traverse from the starting date to the ending date, insert each date as key and 0 as value into the counter, a QMap. The increment of the date is achieved by the function addDays() belong to QDate class.

(2) Count. Traverse loaded data and use the function date() to get the date part of a QDatetime object to find out eligible records and add the times of this day which is searched out through key.

Until the final map is prepared, I set it as a parameter of a signal and emit the signal to the upper layer for visualization.

2.7 Visualization

I use QtCharts mode to achieve most of the visualization.

For function (2), I use a bar chart to show the result.

For function (4), I use a table widget to show the result.

For function (3) and (5), if the data points are fewer than 10, thet will use interpolation to smooth the line. (But I think the image was strange when there are two adjacent points both with value of 0.) Another weak point, I have tried many methods but can't set an axis with text for a line chart. I can only give first day "1", second day "2" as abscissas just like this.

3 Results

The application can work well in some conditions, but it can't work well in all conditions.

4 Discussions

4.1 Drawbacks

My application can’t process too many data quickly. When I try to load all the 1502536 records, the Qt Creator closed itself in the middle. The progress bar would continue to work until 100%. But because the Qt Creator had closed, I can’t see the debugging information. If I load 500000 records, the Qt Creator would not close during the loading progress. But when I try a function, it takes 20-30 seconds to give the result and during this period the Qt Creator would closed as well. (I have tried running another classmate’s project realized by database on my laptop and it runs so silky.)

When you have drawn an image in a widget, if you want to draw anther one for other data then, you need to close the widget and open a new one. If not, no new thread is available for computing and the program will be closed forcefully.

There is no error handling in my code. I feel sorry that the application can’t deal with the users’ misinput.

4.2 Some Thinking

4.2.1 About data storage

My initial plan was to use QSqlite database to store the data and QSqlQuery to manipulate the database to filter the data. But when I finished writing the code for loading data and started to debug, I found that a database can only be used in the thread in which I created it. It can’t be used in other threads. I couldn't find a solution for that in a short time. Thus, I switched to the current way of storing data. Actually, it is perfect to store and process a large amount of data with database. Its query efficiency is very high. Although it is a pity that I did not grasp it well this time, I will continue to learn database in the future.

4.2.2 About records

The records in the .csv file are sorted by user’ ID. It makes the data analysis based on users much easier than that based on POI.

5 Summary

I have learned much from this project.

Firstly, I have learned a lot about the specific use of Qt through searching the Internet, consulting classmates and doing experiments.

Secondly, I experienced the whole process of completing a project on my own. Compared to the time I spent coding, I spent as much or more time thinking about different problems I faced during the process. Before I started coding, I spent at least a week to search information to figure out the solution for each function. I also wrote a planning case before working on code. When I encountered difficulties in the process, I had to come up with the most cost-effective solution as soon as possible, because I didn't have a lot of time to waste.

Thirdly, my problem-solving skills have improved. My logical thinking ability has been improved as well.

Finally, sincere thanks to my professors and TAs.