Ui Design:Data analyzing and visualization of Hangzhou Metro Traffic

Haotian Xue 518021910506

I. Introduction

As is mentioned in the project introduction, these yeai is witnessing a surge of urban metro traffic in man places in China, which has been facilitating our life and a the same time, causing overcrowding problems. And tha makes it necessary to make analysis of the metro traffi data from our daily life.

In this project, I made a user interface based on Qt5 t process the data from the AFC dataset and implemente some user-based functions to benefit the users. The mai functions include: downloading necessary data for th users, make a visualization of the in-out flows and make route planning based on the user's requests.

Other work includes: optimizing route advising algorithm based on some prior data, making visualization of the crowding degree of stations and integrating some Hangzhou elements into the user interface and so on.

II. Implement details

A. Developer environment

All of this project are written in Qt5(C++) in Linux(ubuntu 18.04). Other compilation environments include: Linux g++, sqlite and qmake. The more specific running environment can be referred to in README.txt.

B. Ui structure

The Ui are made up of four main components: open window, dataloader, visualization window and route planning window. All of the four parts interact with each other and all of the three windows interact with the users. The structure details are shown in Figure1.

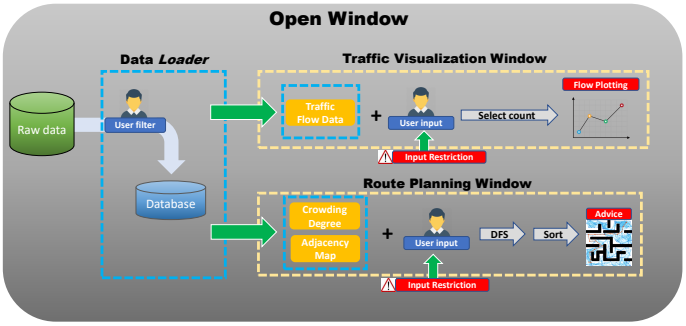

Fig. 1. Ui Structure

C. Function implementation

1) Dataloader and filter

The dataloader aims the move data in the raw .csv file into a data container which can be then used to serve the data analysis.

At first, I try to simply save the data into a container written in the memory of the computer. This actually makes the data processing really fast, however, this process will make up a fair share of memory in my computer and will cause the computer to run out of memory if it is not carefully designed.

So I choose Sqlite3 database as a container of these raw data, which from my knowledge can be more efficient espe cially on data selection.

Data filter is a kind of interaction with the users in data processing, users can make selections of their areas of interest to make downloading. It should be noted that:

1.As some data is necessary for the data analysis, I made some little restrictions on the selection.

2.The dataloader works in a new thread to make sure that you can do other work when waiting.

3.Some functional windows can be opened if and only if when the data processing is done.

4.The message box will make suggestion for a proper operation.

2) Route planning

Route planning window is created to make route advice for users. The users can input their departure station number and their destination station number and get a optimzed travelling route. I use DFS to make a search in the adjacency map, and save all the possible route. For each possible, a score is calculated based on the following equation:

$$Score[i] = w_l L[i] + w_a A[i] + w_c C[i] (i = 1, 2, ...n) \quad (1)$$

where $L[i], A[i], C[i]$ are the length, station changing numbers and the crowding degree of the ith possible route. This is base on the everyday feeling: the longer way and more station changing will make the journey time consuming, the more crowding station is always not preferred. $w_l, w_a, w_c$are the coefficients of the three factors, which represent the important of each factor. I set them to 0.5, 0.3, 0.2respectively. The lower the score, the more preferred the route.

It should be noted that the calculation of length is based on the simple assumption that(L(i, j) represents the distance between station i and station j):

$$L(i, j) > L(i^{'}, j^{'}) (|i - j| > |i^{'} - j^{'}|) \quad (2)$$

For example, the distance between station 1 and station 10 is larger than that between station 1 and 3 at most

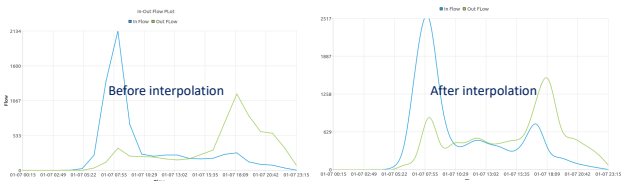

Fig. 2. Curves before and after interpolation when point number is small.

time. So the length of the route can be computed as(n_i is the station number in route i, L[i][j] is the number of the j-th station in route i):

$$L[i] = \sum_{1}^{j=n_i-1} \Big| L[i][j+1] - L[i][j] \Big| \tag{3}$$

The crowding degree is calculated simply based on the prior data in the database, and $C[i] = n_i$.After computing all of the scores, I sorted them and select top-2 which have a lower score: route1 and route2. route2 will be deleted if:

$$2S(route1) < S(route2) \tag{4}$$

3) Visualization of metro traffic flow

This part is designed to make the data more touchable for the users.In this window. users can set the stationID, time scope and time step to get a line chart of the in-out flow of the selected station. The interpolatory spline is used if the number of sample is less than 30 to make the curve more smooth. The passenger flow here are defined as the total number of people entered or leaved the certain station in the time scope. Some discussions about it can be referred to in the section 4. In this part I also implemented a visualization of the crowding degree of stations in different lines to make the users have a general view of the metro traffic overcrowding.

To to exactly, in the above description, I defined the in(out) flow as the total number of passenger entered(leaved) the certain station in the time step. When the number of point is less than 30, the interpolation will work: assume we have n point now, notated as (x_1, y_1), (x_2, y_2), ..., (x_n, y_n), now we should make a cubic polynomials connect between the adjoint two points notated as $f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$, they should follow:

$$\begin{cases} f_i(x_i) = y_i & i = 1, 2, ..., n \\ f_i(x_{i+1}) = y_{i+1} & i = 1, 2, ..., n-1 \\ f_i^{'}(x_i) = f_{i-1}^{'}(x_i) & i = 2, 3, .., .n \\ f_i^{'}(x_{i+1}) = f_{i+1}^{'}(x_{i+1}) & i = 1, 2, ..., n-1 \end{cases} \tag{5}$$

Then the curve will be smoother when the number of point is too small.(Fig.2.)

Another problem is that When the time step is relatively large (longer than 3 min), the plot will be smooth and is comfortable for users to get information from it, but when the time step is shorter than 2 minutes, the curves begin to vibrate(Fig 6).

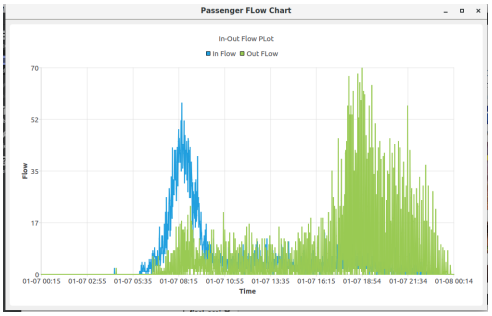

Fig. 3. The curves vibrates when the time step is too short.

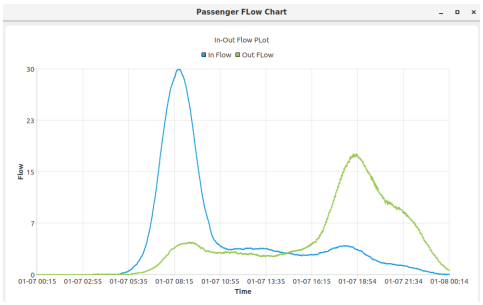

Fig. 4. Curves after applying the filter. The time range and step is the same as those in Fig.2. We can see that the trend makes sense though the absolute values change a lot.

Actually, the trend of curves is much more important for the users than the exact shape of curves. So I thought about using filters to get the trend of the data(D sized of n). The method can be written as (6) and (7):

$$D_i = \frac{1}{2w+1} \sum_{j=i-w}^{i+w} D_j \tag{6}$$

where w is half the size of window, if the index is out of range, the value will be 0.

after applying the raw data list to the filter 5 times, we get the trend curves(Fig.3.), which makes sense.

D. Polished Ui design

In order to make the users have a better experience and at the same time integrating into it the characteristics of Hangzhou, I made a polishment of the user interface especially on background of windows and the widgets' outlook. I selected some of the typical city elements of Hangzhou including the West Lake the and modern city's night view. It actually took me a lot of time to make the layout and color of widgets like buttons, text and input boxes matching the background. More designs can be seen in the Result part.

III. Results

In this part, I will show some images from the user's angle of this Ui(from Figure 5 to 8). Some time-consumption(TIME CMD) data will also be presented in this part(Table 1).

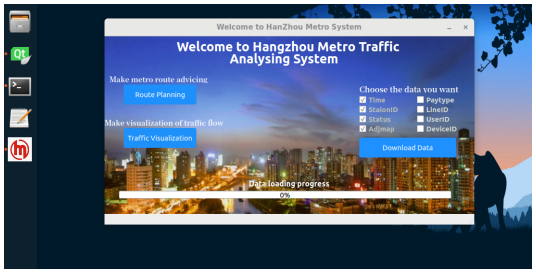

Fig. 5. Ui design of openwindow:Night view of Hangzhou

Fig. 6. Ui design of data visualization window: Hangzhou traffic

IV. Discussion

A. Some findings from the raw data

1.Through the analyzing process, I find that the data of station 54 is actually in absence.

2.We can analyse from the data that station 68-81 belong to lineA, 1-35 belong to lineB, 35-69 belong to lineC. So station 35 is the interchange station between lineC and lineB, and station 69 is the interchange station between lineA and lineC.

3.According to the plot, around 8:00 am it comes the first summit of passenger flow in a day, and in about 6:00 pm , the second summit comes. This result makes sense for the two summits are actually the time for citizens to go to work and go home respectively.

B. About the speed of dataloader

At first, I used inner memory instead of database to store the data in the ubuntu18.04 in VisualBox. However, the program will stopped for "out of memory" when I load the data. Asking for help, I found that the inner memory of my visual environment is less than 6G, which is uncapable to store so many data in the memory.

Then I turn to sqlite3 database, when using transaction function, the inserting speed can achieve less than 1.5s for each csv file. The query time is much faster when the united index is utilized.

In most time, database is a more general method to instore data than simply put data into memory since the memory of computers vary from each other. Using structure like B+ Tree, It can greatly improve the query process.

Some better polishment must can be implemented to make the inserting process faster.

TABLE I
TIME CMD

Function	TimeCMD(avg)
Data loading(1 file)	1007 ms
Data inserting(All)	3.5 min
Route planning(avg)	1.6 ms
Visualize Flow(24 point avg)	637 ms
Visualize Crd(avg)	15 ms

C. The design of route planning system

The real route planning system is much more complex for it counted into more factors including: real-time crowding degree, weather, speed restriction, events, just to name a few. And all of the importance of these factors should be carefully computed even using machine learning method or deep learning method.

In my implementation, I tried my best to make the planning more practical. However, some problems still exist. For example, the coefficeint may not be consistent with the real importance, the route adviser includes too limited information, the calculation of especially the length between two certain stations is not always valid if the two stations are adjoint but do not share the adjoint station numbers.

D. About the trend fitting in flow visualization

As I have mentioned in section II, the filters were used when the the time step is too short. At first, I used the following iterating method as the fitler in consideration that 0 takes a fair part in the flow list:

$$(D_1, D_2, ..., D_n) := (J(\frac{L_1 + R_1}{2}), J(\frac{L_2 + R_2}{2}), ...J(\frac{L_n + R_n}{2}))$$

where $L_i(R_i)$ is the first list number left(right) to Di, $x := J(y)$ means if x is not 0, x remains unchanged; otherwise x is changed to y. However, when applying this kind of filter, I found that the trend changed greatly: the summit of the curve moved greatly, which may be caused by the great number of 0.

I also thought about using polynomial regression method to fitting the trend and actually it took me a long time to do that. I tried to fit a polynomial with degree n(n=10, 5, 3). The real data is: (x_1, y_1), (x_2, y_2), $...(x_n, y_n)$, the target polynomial is: $f(x) = a_0 + \sum_{i=1}^{n} a_i x^i$

When fitting the trend, the non-zero points are much more important than the zero ones. So I defined the loss function as:

$$Loss(x, y) = \sum_{i=1}^{n} \beta(y_i)(f(x_i) - y_i)^2 \tag{7}$$

in the loss function:

$$\beta(x) = \begin{cases} 0.1 & x = 0 \\ 1 & otherwise \end{cases}$$

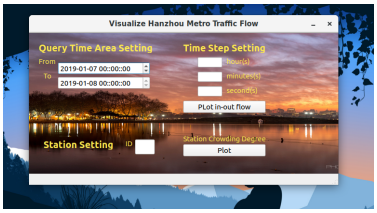

Fig. 7. Ui design of route planning window: Beautiful West Lake

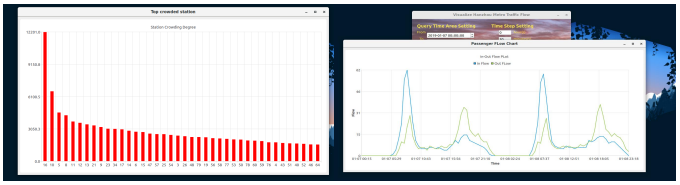

Fig. 8. Data visualization from the user's angle.

so we can calculate the partial derivative and operate SGD on each parameters.

$$\partial_i = \frac{\partial Loss(x,y)}{\partial a_i} = \sum_{i=1}^{n} 2\beta(y_i)x_i^i(f(x_i) - y_i) \tag{8}$$

$$a_i = a_i - LearningRate * \partial_i \tag{9}$$

V. Acknowledgement

CS241 is a new course for SJTU computer science major, I acquired more than I can image at first from the patient teaching offered by Mr Ling and Mr Jin.

Through this course I gain a overview of the principle and thoughts of C++.I learned how to make basic GUI through Qt. This course also expose me to some algorithm and data analyzing tools: Genetic Algorithm, Dynamic Programmming, Interpolation method, Multithread Programming, just to name a few.

Instead of just extending theoretical explanation, this course let me to make a implementation of some of the basic method taught in class. The final project can be deemed as a comprehensive practice in all of these work.

I will also express my sincere gratefulness to TA Zhu, and TA Sun for their great efforts to helping with my learning in this course.