Design and Implementation of Comparative Analysis System of Unidimensional Directional Data

Wu Qiong[1, a], Wu Chensheng[2,] Zhang Wei[3]

[1]No.140, Xizhimenwai Street, Xicheng District, Beijing, 100044 P R CHINA

[a]wwqqppdd@163.com, [b]wu1082@163.com, [c]greenzw80@gmail.com

Key Words: web crawler; data warehouse; air quality index; traffic index

Abstract. This paper will design a set of comparative analysis system of unidimensional directional data to study correlation between traffic conditions and air pollution by directionally collecting air quality index and Beijing traffic data, and establish a mathematical model to forecast the change of air quality index in local area, so as to provide data support and theoretical basis for correctly understanding and processing the haze. The construction and application of the system will provide new ideas and support for research on air pollution prevention and control.

Introduction

A large number of overseas studies have confirmed that the short-term change of air pollution is closely related to the number of human deaths per day. In epidemiological studies, extensive studies have shown that respiratory diseases are most susceptible to air quality. As an important cause of death, respiratory disease is attracting more and more attention. At the same time, with rapid development of social economy, the urban air pollution is changing from coal smoke pollution to the combination of coal smoke and exhaust of motor vehicle, and meanwhile, due to aging in urban areas, the disease of circulatory system has become a major health hazard of residents. Therefore, this study will research and develop a set of comparative analysis system of unidimensional directional data to directionally collect the Air Quality Index (AQI) and traffic index in Beijing and carry out the modeling analysis to evaluate the intrinsic relationship between air quality index and traffic index, in order to provide data support for further modification and establishment of effective environmental laws and regulations.

Design of Comparative Analysis System of Unidimensional Directional Data

Work flow of topical crawler

The directional data collected in the system will be accurately crawled by using the topical crawler through the way of giving keywords. The crawling process is divided into the following steps:

(1) URL initialization: The queues to be accessed can be constructed as an FIFO queue, and the next crawling page of the information search comes from the queue header, and the new one is added to the queue tail. In each step, the next page is selected from the queue header for the crawler to crawl until all pages in this queue are crawled.

(2) Page reading: The web crawler first determines the type of file. For the multimedia data, it will be directly downloaded and stored to web database; and for the unstructured free text or semi-structured HTML (including Text, HTML and other formats) type, it will be analyzed continuously. When reading the corresponding page of URL, if the timeout occurs, the page is invalid, and the corresponding URL is added to the wrong queue. On the contrary, if the time is out, the web page parsing content should be read [1].

(3) Webpage parsing: After getting the page, it is necessary to parse the content to extract the required information and to guide the crawling path of the crawler in the future. Since the HTML

file is composed of “text” and “tags”, the parsing process is the process of analyzing the whole source file content and extracting the URL tag.

Design of data analysis and display module

The data mining analysis module of this system will adopt the association rule mining algorithm Apriori, K mean clustering algorithm, K center point algorithm and FP-growth algorithm of association algorithm. Different data sources will adopt different algorithms to mine and analyze valuable information. The K mean clustering algorithm is simple and has relatively fast execution and convergence process, but it is certain limited in use, so the use of K center point algorithm is to supplement and improve the K- mean deficiency. The main points are as follows: Don't use the average value of the objects in the cluster as the reference point; choose the center of the object in the cluster, that is, the center point. This partitioning approach is still based on the principle of minimizing the sum of dissimilarity between each object and its reference point. The most important feature of this system is that the scope of application is wider than that of the traditional mining analysis system, and it can be used for mining different data sources. The algorithm used in the module will be designed as follows.

Design of call relation between modules

The data crawling module is the basic module shared by other modules. All of data processing, data storage, data analysis and data query need to obtain relevant information from the data crawling module. After these modules returns network request, some personal information will often be changed, for example, after buying virtual goods, the user points will change, the server will return the spent points of users and the remaining points of users, then the module should update the points in personal information module and notify other modules to update the interface.

The message mechanism is an important part of the module call relationship. Modules cannot hold object entities of each other, so it is necessary to design a way to transfer information between modules. The most popular way is to call the program by the user on interface layer of a module, call related proxy class of business logic level by the controller, transfer the class to the business logic class so as to realize the process of business logic, and return after synchronous or asynchronous processing. The call method can be divided into two types: synchronous and asynchronous. The synchronous call mode indicates that the main module A needs to call the function of other modules B to obtain the data needed for further processing in order to carry out the next work. The asynchronous call mode indicates that the main module A generates the processing result and needs other modules B to further process or store the processing result [2].

For asynchronous calls, the implementation of Callable and Runnable interfaces in Java can be a good solution, while synchronous calls are relatively complex. It is easy to think of the solution of calling back interface from the agent level by main module A and calling the corresponding module B by the agent level, or calling the required module B for preparation in the agent implementation and processing it with the called module A. This can make the sender of the message focused on the result of the call, the call object is handed over to the agent level, but the problem is that the complex and changeable module interaction will greatly increase the content of agent level and make the logic level of architecture content complex. When the module logic is changed, the agent level should also be correspondingly changed, and this greatly increases the cost of code modification.

Function design of interface between modules

The module interface design is mainly aimed at the design of agent layer interface function in the module, which mainly includes that the module interface should include the function called among modules. The main interface functions of the main modules are defined in the diagram, and the function functions are implemented according to this interface in the development process to ensure that the modules can provide the data needed by themselves and other modules. Each module has the interface design, the data crawling module obtains the URL elements by calling the add Element interface and obtaining the data by the get Goal Content; the data processing module processes special words and natural language by nlprocessing interface; and the algorithms of data analysis depend on the completion of data processing module.

Workflow design of system module integration

After the design of each part of the system is completed individually, the module integration workflow is designed as follows:

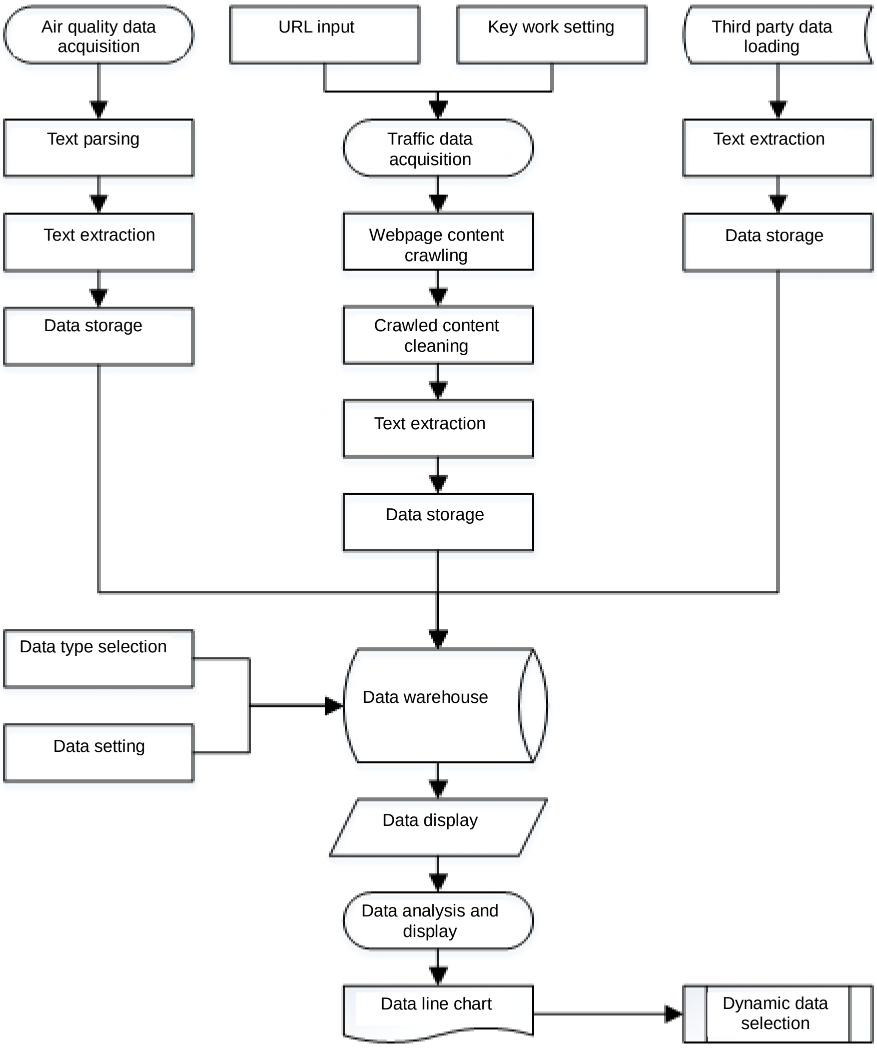


Figure 1 Workflow of Modules Integration

As shown in Figure 2, the air quality and traffic data acquisition is handled by the directional data acquisition module; the third party data loading and data warehouse process is processed by the data storage and processing module; and the data analysis and display process is processed by the data analysis and display module. The workflow among all modules is shown as the figure.

Implementation of Comparative Analysis System of Unidimensional Directional Data

Implementation of directional data acquisition module

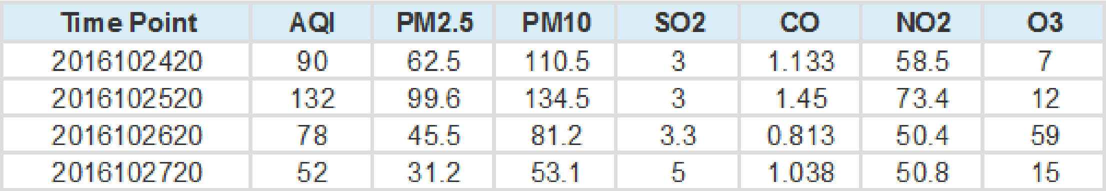
Implementation of API interface acquisition module of air quality index

The feedback data from API interface include the content shown in the following Table 1.

Table 1 Feedback Data of API Interface of AQI

Field	Field Description
aqi	The Air Quality Index (AQI) indicates the dimensionless index describing the air quality condition
area	The name of city
position_name	The name of monitoring point
station_code	The code of monitoring point
so2	The 1-hour mean of sulphur dioxide
so2_24h	The 24-hour moving average of sulphur dioxide
no2	The 1-hour mean of nitrogen dioxide
no2_24h	The 24-hour moving average of nitrogen dioxide
pm10	The 1-hour mean of particulate (no greater than 10μm)
pm10_24h	The 24-hour moving average of particulate (no greater than 10μm)
co	The 1-hour mean of carbon monoxide
co_24h	The 24-hour moving average of carbon monoxide
o3	The 1-hour mean of ozone
o3_24h	The daily maximum 1-hour moving average of ozone
o3_8h	The 8-hour mean of ozone
o3_8h_24h	The daily maximum 8-hour moving average of ozone
pm2_5	The 1-hour moving average of particulate (no greater than 2.5μm)
pm2_5_24h	The 24-hour moving average of particulate (no greater than 2.5μm)
primary pollutant	The primary pollutant
quality	The air quality index includes “perfect, good, light pollution, moderate pollution, heavy pollution and serious pollution” 6 types
time point	The time of publishing data

The data acquisition results are shown as the following Figure 2. The system directionally collects 8 fields, including time, at 20:00 every day.



Time Point	AQI	PM2.5	PM10	SO2	CO	NO2	O3
2016102420	90	62.5	110.5	3	1.133	58.5	7
2016102520	132	99.6	134.5	3	1.45	73.4	12
2016102620	78	45.5	81.2	3.3	0.813	50.4	59
2016102720	52	31.2	53.1	5	1.038	50.8	15

Figure 2 Sample of AQI Acquisition

Implementation of topical crawler module of traffic data

The data source of this system mainly comes from the Internet, so this module is mainly realized by using the network crawler technology. It is the premise of the system to achieve data analysis and management, so the reliability of the data is crucial. The design and implementation of the directional data crawling module are introduced in detail below.

At present, web crawler generally adopts breadth first and depth first strategies when crawling web pages. The breadth first indicates that the crawler crawls along the width direction of the tree until all webpages connected to the start page are crawled, then select one of the adjacent link webpage to crawl data until all webpages are crawled. This method allows the crawler to run in parallel and improves the speed and crawling efficiency. The depth first indicates that the crawler crawls non-visited nodes along the depth direction of the tree. The depth first is a recursive process, and the crawler program often consumes a large amount of computer memory in executing process. In many cases, the crawler will fall into a crash problem. Because of the incompatibility between recursion and multithreading (multithreading allows multiple tasks to run at a time, each parallel thread has its own heap search, and when a method calls itself, it needs to use the same stack. Therefore, the data crawling of this system adopts the breadth first traversal strategy [3].

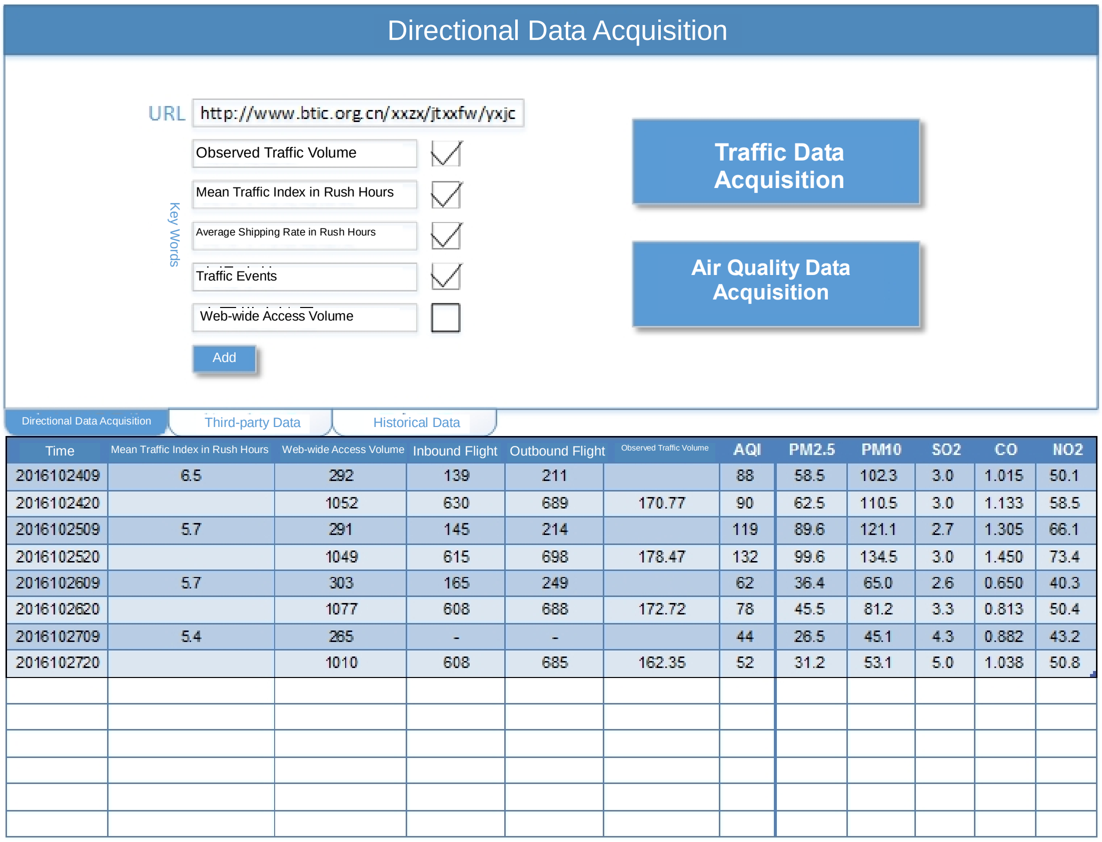
When a crawler used in this system accesses a site, it first checks whether there is a Robots.txt file in the root directory of the site. If yes, the crawler will determine the scope to access in accordance with the content of the file; and if the file does not exist, the crawler will crawl the data along the link.

This crawler uses nonrecursive method to realize the crawling process. When the program is implemented, 4 queues are constructed: waiting queue, run queue, completion queue, and error queue. The waiting queue is a collection of initial URL of crawler and URL newly discovered by crawler. The run queue is a collection of URL that the crawler is processing. The completion queue is a collection of URL that has been crawling. The error queue is a collection of URL when the crawler program is in page parsing error or data reading timeout. When the program is executing, one URL can only be in one queue at one time, which is called as a URL state. The program changes from a state to a state according to a state chart.

A URL will experience four states from the to-be-processed state to the completely-processed state: first, in the waiting queue, the URL waits to be processed by the Robot, and the newly discovered URL is added to the queue. When Robot starts processing the URL of a web page, the URL is sent to the running queue for processing. At this time, if the Robot makes an error when crawling a webpage, then the URL of this webpage will be sent to the wrong queue, and the URL in wrong queue cannot be moved to other queue; and if the Robot successfully crawls a webpage, then the URL of this webpage will be sent to the completion queue, and the URL in completion queue cannot be moved to other queue, wherein, when the URL in waiting queue is being moved to the run queue, it should be first compared with the URL in the completion queue to prevent repeated crawling; and after one URL in run queue is processed, the URL in waiting queue should be added to the run queue according to the first-in-first-out rule, and the corresponding URL in the queue should be deleted.

Implementation sample of directional data acquisition function

The final sample of directional data acquisition module of this system is shown as the following figure:



Directional Data Acquisition

URL http://www.btic.org.cn/xxzx/jtxxfw/yxjc

Key Words
- Observed Traffic Volume ☑
- Mean Traffic Index in Rush Hours ☑
- Average Shipping Rate in Rush Hours ☑
- Traffic Events ☑
- Web-wide Access Volume □

Time	Mean Traffic Index in Rush Hours	Web-wide Access Volume	Inbound Flight	Outbound Flight	Observed Traffic Volume	AQI	PM2.5	PM10	SO2	CO	NO2
2016102409	6.5	292	139	211		88	58.5	102.3	3.0	1.015	50.1
2016102420		1052	630	689	170.77	90	62.5	110.5	3.0	1.133	58.5
2016102509	5.7	291	145	214		119	89.6	121.1	2.7	1.305	66.1
2016102520		1049	615	698	178.47	132	99.6	134.5	3.0	1.450	73.4
2016102609	5.7	303	165	249		62	36.4	65.0	2.6	0.650	40.3
2016102620		1077	608	688	172.72	78	45.5	81.2	3.3	0.813	50.4
2016102709	5.4	265	-	-		44	26.5	45.1	4.3	0.882	43.2
2016102720		1010	608	685	162.35	52	31.2	53.1	5.0	1.038	50.8

Figure3 Sample of Directional Data Acquisition Function

Implementation of data storage and processing module

Implementation of data processing technology

The data preprocessing mainly realizes data extraction, conversion, cleaning and loading, and aims to convert decentralized, non-integrated, hard-to-access and variable data sources, variable platforms, poor data quality, redundant data, and non-analyzable data into structured and predefined data warehouse model by ETL process and load the data to the data warehouse [4].

If above works are achieved and completed individually, the work will be time-consuming. Therefore, this module will be added with an ETL processor to assist the completion of part of work, so that the processing is simpler and faster and the energy can be more concentrated in the design of the core part. Here, we use the current popular IBM Datastage processor to complete part of the data preprocessing work.

The Datastage implements an ETL process through Job, which can run multiple instances by specifying different parameters at runtime. Each tool provides a graphical interface with simple operation, good custom development and more flexible batch process. The Datastage at least can set parameters for each job and can refer to the parameter name in job.

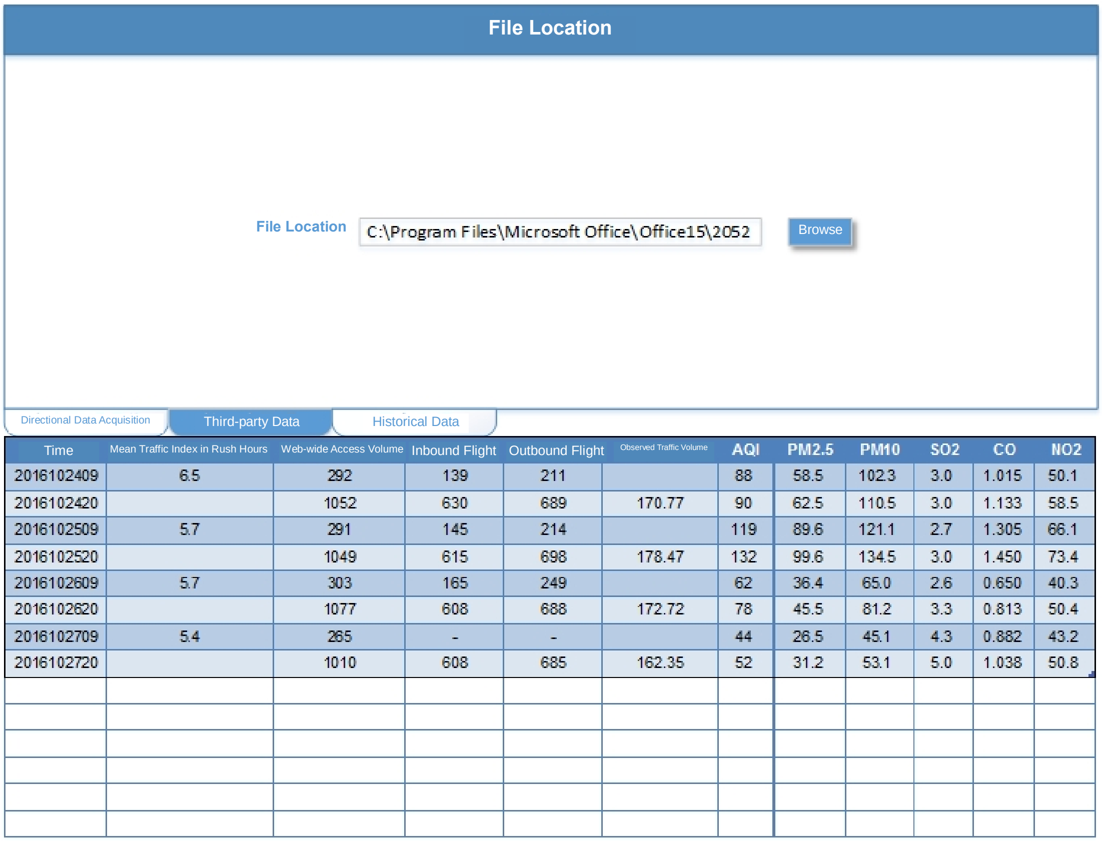
In addition to the above three items, there are many other functions and available useful data. All of the data lists can be downloaded directly. With these data, it is unnecessary to filter the web log, and the task now becomes processing the existing data: first, accounting the downloaded data list to obtain detailed browsing behavior of each user in website, then processing the data of each user to avoid the data cleaning and user session identification and reconstruction.

The Data formatting is the last step of data preprocessing. Once the processing of the preceding part is completed, the data can be formatted to prepare the next step of data mining.

These data can be stored in an association database to provide log data query and apply to common pattern mining, and tree structure index can be used to make queries more efficient [5].

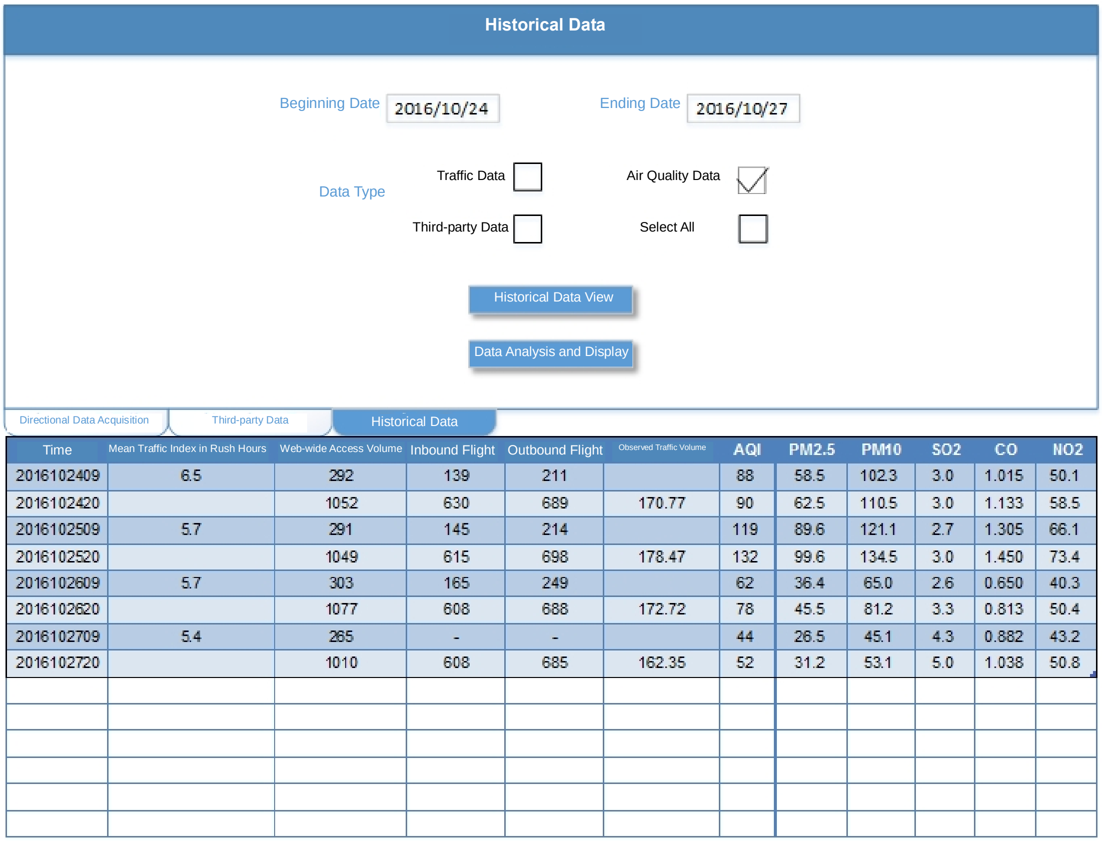
Implementation samples of data storage and processing function

The final samples of directional data acquisition of this system are shown as the following two figures:


File Location
File Location
C:\Program Files\Microsoft Office\Office15\2052
Browse
Directional Data Acquisition
Third-party Data
Historical Data

Time	Mean Traffic Index in Rush Hours	Web-wide Access Volume	Inbound Flight	Outbound Flight	Observed Traffic Volume	AQI	PM2.5	PM10	SO2	CO	NO2
2016102409	6.5	292	139	211		88	58.5	102.3	3.0	1.015	50.1
2016102420		1052	630	689	170.77	90	62.5	110.5	3.0	1.133	58.5
2016102509	5.7	291	145	214		119	89.6	121.1	2.7	1.305	66.1
2016102520		1049	615	698	178.47	132	99.6	134.5	3.0	1.450	73.4
2016102609	5.7	303	165	249		62	36.4	65.0	2.6	0.650	40.3
2016102620		1077	608	688	172.72	78	45.5	81.2	3.3	0.813	50.4
2016102709	5.4	265	-	-		44	26.5	45.1	4.3	0.882	43.2
2016102720		1010	608	685	162.35	52	31.2	53.1	5.0	1.038	50.8

Figure 4 Sample 1 of Data Storage and Processing Function



Historical Data

Beginning Date 2016/10/24 Ending Date 2016/10/27

Data Type

Traffic Data □ Air Quality Data ☑

Third-party Data □ Select All □

Time	Mean Traffic Index in Rush Hours	Web-wide Access Volume	Inbound Flight	Outbound Flight	Observed Traffic Volume	AQI	PM2.5	PM10	SO2	CO	NO2
2016102409	6.5	292	139	211		88	58.5	102.3	3.0	1.015	50.1
2016102420		1052	630	689	170.77	90	62.5	110.5	3.0	1.133	58.5
2016102509	5.7	291	145	214		119	89.6	121.1	2.7	1.305	66.1
2016102520		1049	615	698	178.47	132	99.6	134.5	3.0	1.450	73.4
2016102609	5.7	303	165	249		62	36.4	65.0	2.6	0.650	40.3
2016102620		1077	608	688	172.72	78	45.5	81.2	3.3	0.813	50.4
2016102709	5.4	265	-	-		44	26.5	45.1	4.3	0.882	43.2
2016102720		1010	608	685	162.35	52	31.2	53.1	5.0	1.038	50.8

Figure 5 Sample 2 of Data Storage and Processing Function

Conclusion

This paper designs and develops a set of comparative analysis system of unidimensional directional data to directionally collect air quality index and traffic data in Beijing. Firstly, provide data crawling method from data acquisition module and realize the module function; secondly, realize the data storage and processing module of the system, establish the data warehouse and pre-process the data by Datastage; finally, realize the data analysis and presentation module and try to seek potential correlation between air quality index and traffic data by data mining technology. This study provides data support for further modifying and establishing effective environmental laws and regulations.

References

[1] Zhang Weiming, Principle and Application of Data Warehouse, Publishing House of Electronics Industry, 2002.

[2] Qi Guohui, Development and Direction of OLAP Technology-The Way of Data Warehouse, http://www.dwway.com, 2003

[3] Wang Hongming, Ajax-based Web Information Extraction System Design and Implementation, Sciencepaper Online, February 2010.

[4] Shen Wenqin, Li Qingchao and Shao Zhiqing, Incremental Crawling and Shadowing Update Strategy in Search Engine, Journal of East China University of Science and Technology: Natural Science Edition, Vol.30, No.3 June 2004, Page 284-287.

[5] Zhang Xiaoxiang, Deep Experience to Inside Development of Java Web -- Core Foundation, Publishing House of Electronics Industry, 2006, Page 349-358.