DOI:10.18686/ahe.v8i6.13523

Urban Carbon Emission Detection Platform Based on Python

Desai Fan, Yan Kai, Xinmiao Yan, Xiaoqian Wei, Zhigang Zhu*
City Institute, Dalian University of Technology, Dalian, Liaoning 116600

Abstract: With the increasingly severe global climate change, reducing greenhouse gas emissions has become the consensus of all countries in the world. China is undergoing rapid urbanization, and it is urgent to strengthen urban carbon emission management. This paper analyzed the characteristics and management requirements of urban carbon emission data, proposed the overall architecture design of the platform, adopted Python for modular development, built a structured carbon emission data warehouse, and implemented a variety of carbon emission accounting and analysis algorithms such as IPCC emission factor method, STIRPAT model and ARIMA time series prediction. The platform is also tested and evaluated from the aspects of system performance and application effect, and good feedback is obtained in the actual application of departments.
Keywords: Urban carbon emission; Testing platform; Python; Data analysis

Fund Project:
College Student Innovation and Entrepreneurship Project of Liaoning Province (Project number: S202413198013, Project name: Urban Carbon Emission Detection Platform based on python)

With the increasingly prominent issue of global climate change, reducing greenhouse gas emissions has become a general consensus of the international community. As a major source of greenhouse gas emissions, cities play a key role in the fight against climate change. As the largest developing country in the world, China's urbanization process is advancing rapidly, and the problem of urban carbon emissions is becoming increasingly prominent. Accurate and timely understanding of urban carbon emissions is of great significance for formulating targeted emission reduction policies and promoting sustainable urban development. As urban carbon emissions involve energy consumption, industrial production, transportation, buildings and other fields, data sources are extensive and diverse, which brings great challenges to carbon emissions accounting and tracking. Traditional manual collection and calculation methods cannot meet the needs of real-time dynamic monitoring, and advanced information technology is urgently needed [1]. Python has rich libraries and strong community support, which is very suitable for building data-intensive applications. This paper aims to explore the use of Python technology, develop a full-featured, easy-to-use urban carbon emission detection platform, and realize the automatic collection, rapid processing and intuitive display of carbon emission data.

1. Platform architecture and technical route

The urban carbon emission detection platform designed in this paper adopts a hierarchical architecture based on Python, which mainly includes three levels: data layer, algorithm layer and application layer. In the data layer, Python data collection and processing libraries such as Requests, BeautifulSoup and Pandas are used to obtain carbon emission related data from authoritative data sources in energy, industry, transportation, construction and other fields, and data cleaning, conversion and integration are carried out to form a structured carbon emission data warehouse. Based on Python's scientific computing libraries NumPy and SciPy, the algorithm layer implements a series of algorithm models for carbon emission accounting, prediction and analysis, including IPCC emission factor method, STIRPAT model, ARIMA time series prediction, etc. At the application layer, the Web interface is built by Flask, a Python Web framework, and visualization libraries such as Echarts and Plotly are integrated to visually present the analysis results of carbon emission data in the form of interactive maps and multidimensional charts, providing users with visual carbon emission monitoring

and decision support functions. The whole platform follows the modular design idea, and realizes decoupling and communication between all layers through standard interfaces to facilitate the expansion and maintenance of the system. Python parallel computing library Multiprocessing is used to optimize the execution efficiency of key algorithms, and Unittest is used for unit testing.

2. Data collection and integration

The data collection and integration module of the urban carbon emission detection platform is responsible for extracting and integrating carbon emission related data from multi-source heterogeneous data. Using Python's Requests library to write a general data climbing framework, you can automatically climb according to the configuration file of energy, industry, transportation, construction and other fields of authoritative data sources, such as the National Bureau of Statistics, the Ministry of Ecology and Environment, the city statistical yearbook. For structured data such as CSV, Excel, etc., Pandas is used for data reading and pre-processing; For unstructured data such as HTML and PDF, BeautifulSoup and PyPDF2 are used for parsing and information extraction. Outliers and missing values were processed by means of mean filling and regression interpolation, and carbon emission data from different sources and formats were mapped to a unified data model. Finally, the data merging and linking operation of Pandas is used to integrate the cleaned carbon emission data according to time, space, industry and other dimensions, and the carbon emission data warehouse is constructed. The data warehouse adopts the star model design, including a fact table and multiple dimension tables, supporting flexible data slicing and drilling operations.

3. Data analysis algorithm and model design

The core of the urban carbon emission detection platform is a series of carbon emission accounting, forecasting and analysis algorithms implemented based on Python. For carbon emission accounting, IPCC emission factor method [2] is adopted to calculate CO2 emissions from fossil fuel combustion based on urban energy consumption and emission factors of various energy sources:

$$E_i = \sum_j AD_{i,j} \times \mathrm{EF}_j \times \mathrm{O}_i$$

Where Ei represents the CO2 emission of city i, ADi, j represents the activity level of energy j consumed by city i, EFj represents the emission factor of energy j, and Oi represents the oxidation rate of city i. The matrix operation of numpy is used to optimize the computing efficiency of the emission factor method, and the energy consumption and emission parameters of each city are stored with Python dictionary. The STIRPAT model is used to analyze the driving factors of carbon emission. The model can be expressed as follows:

$$I = \mathrm{a}P^{\mathrm{b}} A^{\mathrm{c}} T^{\mathrm{d}} e$$

Where I represents carbon emissions, P represents population, A represents per capita GDP, T represents technical level, a, b, c, d are parameters to be estimated, and e is the residual term. The parameter estimation and statistical test of STIRPAT model are implemented by using Python's statsmodels library, and the results are visually analyzed by using R's ggplot2.

In addition, urban carbon emissions are predicted based on ARIMA time series model. The mathematical expression of ARIMA (p, d, q) model is as follows:

$$\Phi(\mathrm{B})(1-\mathrm{B})^{\mathrm{d}} X_t = \delta + \theta(\mathrm{B}) Z_t$$

Where φ (B) is the P-order autoregressive coefficient polynomial, θ (B) is the Q-order moving average coefficient polynomial, δ is the constant term, (1-B) d is the D-order difference operator, and Zt is the white noise sequence. Using Python's statsmodels.tsa module to achieve ARIMA model fitting and prediction:

```
from statsmodels.tsa.arima_model import ARIMA
# Fit ARIMA(1,1,1) model
model = ARIMA(co2_data, order=(1, 1, 1))
results = model.fit()
# Carbon emissions projections for the next 5 years
forecast_data, std_err, conf_int = results.forecast(5)
```

4. Data visualization design

In order to visually display the results of carbon emission data analysis and improve the usability and interactivity of the platform, on the basis of Python Web framework Flask, a series of data visualization functions are designed by integrating various visualization tools and libraries: (1) Echarts was used to draw the time trend chart of total urban carbon emission and per capita carbon emission,

showing the horizontal comparison between different cities and the vertical change characteristics of each city; (2) In view of the spatial distribution characteristics of carbon emissions, an interactive map visualization module was developed based on Python's Folium library. By crawling and processing the geographical base map data of OpenStreetMap, carbon emissions of each city are mapped onto the map, and different emission levels are marked with progressive colors [3]. (3) Use Plotly to draw a three-dimensional column chart and radar chart of carbon emissions to analyze the internal mechanism and driving factors of urban carbon emissions from the perspectives of energy structure and industrial structure.

5. Platform performance and application effect evaluation

In order to evaluate the performance of the urban carbon emission detection platform, the platform was tested from the aspects of data processing efficiency and query response speed. After the optimization of database indexing, caching and other technologies, the data retrieval speed of the platform is increased by 30%, and the response time for common data analysis tasks is controlled within 5 seconds, which basically meets the needs of practical applications. At the same time, in order to evaluate the practical application effect of the platform, the relevant staff of the ecological environment Bureau of the city where the project is located are invited to try out the platform and give feedback. The survey results show that users give a high evaluation to the functions and interface of the platform, and believe that the platform is convenient to operate, the chart is clear, and the carbon emission data and analysis results provided are of reference value for management work. In particular, the scenario analysis function can help them quickly simulate and evaluate the effects of different emission reduction policies, and provide beneficial support for the formulation of action plans for carbon peak.

6. Conclusion

As centers of human activity, cities play an indispensable role in addressing the challenge of global climate change. The development of urban carbon emission detection platform based on information technology is an important exploration to promote urban low-carbon transformation and achieve sustainable development. The urban carbon emission detection platform built based on Python technology in this paper provides a new idea and a new method for fine, dynamic and intelligent urban carbon emission management. The development and application of this platform not only helps to improve the timeliness and accuracy of urban carbon emission accounting, strengthen the tracking, monitoring and trend prediction of carbon emissions, but also provides data support and decision-making reference for tapping urban carbon emission reduction potential and formulating low-carbon development policies.

References:

[1] Yang Yin-run, Dong Na. Synergistic effects of building carbon emissions in urban agglomeration [J]. Environmental Science,1-12.

[2] Xia Zhongwei, Fan Chang, Zhang Goli. Estimation of methane emissions from rice fields in China based on an optimized IPCC method [J]. Acta Ecologica Sinica,2024,(21):1-10.

[3] YAN Shengyu, WANG Zhaolong, Wu Jin, et al. Measurement method of vehicle carbon emission in provincial expressway network [J]. China Environmental Science,1-12.

About the author:

Desai Fan, male, Han nationality, 2003-09, born in Heze, Shandong Province, majoring in computer science and technology, City College, Dalian University of Technology

* Corresponding author:

Zhigang Zhu, male, Han nationality, 1984-10, born in Shenyang, Liaoning Province, Associate professor title, teacher, Master degree, City College of Dalian University of Technology, research interests: Software engineering, Computer Science and Technology.