International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 9 Issue 5, Sep-Oct 2025 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Research on Knowledge Graph Visualization Based on *Computer Networks and In-depth Introduction to Computers Networks*

Hang Dong, He Xu, Yi Cheng, Guanyu Liu, Qilun Lyu

Beijing Wuzi University, Beijing, China

ABSTRACT

With the development of educational informatization, learners are shifting from traditional linear reading to structured, visual, and interactive exploration. Taking Computer Network and a simple introduction to computer networks as research objects, this paper constructs a knowledge graph using Neo4j and designs a browser-based visualization system with ECharts.

The system adopts an embedded Neo4j solution, preloading .dump files and packaging with a built-in JRE for one-click cross-platform deployment. It realizes functions such as graph loading, interactive display, dynamic refresh, and health check. The paper details the design of backend APIs, frontend rendering logic, and exception handling mechanisms. Multiple rounds of testing confirm that the system can stably load and interact with graphs, supporting scenarios with up to 1000 nodes and 1500 edges, This not only helps learners better organize fragmented knowledge and grasp the logical relationships between concepts but also effectively enhances the efficiency of knowledge review.

KEYWORDS: *Knowledge Graph Visualization; Embedded Neo4j; ECharts; Flask Framework; Computer Network Teaching, Cross-platform Deployment; User Interaction; Data Rendering.*

How to cite this paper: Hang Dong | He Xu | Yi Cheng | Guanyu Liu | Qilun Lyu "Research on Knowledge Graph Visualization Based on Computer Networks and In-depth Introduction to Computers Networks" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-9 | Issue-5, October 2025, pp.461-465, URL: www.ijtsrd.com/papers/ijtsrd97511.pdf

IJTSRD97511

INTRODUCTION

In the traditional learning mode, students usually need to read textbooks page by page, which makes it difficult for them to quickly locate the connections between knowledge points and is not conducive to forming an overall understanding. Especially in the computer discipline, where there are a large number of knowledge points and complex connections between concepts, learners often encounter the situation of "remembering local knowledge but not understanding the overall structure". Computer Networks and A Concise Course in Computer Networks start from computer communication and basic knowledge respectively, and systematically introduce contents such as network layering, protocols, operating systems, and computer composition, making them very suitable for constructing cross-chapter knowledge graphs [1].

Through knowledge graphs, knowledge points in textbooks are transformed into nodes, and connections between concepts are presented in the form of edges. Learners can explore the knowledge system as if browsing a map. Visualization can not only display concept definitions and attributes, but also reveal the sequential dependency relationships, inclusion relationships, and potential connections of knowledge points, helping to form a systematic understanding [2]. For the visual presentation of knowledge graphs, this project adopts a hierarchical structure display method based on force-directed layout, which can more intuitively present the hierarchical relationships and connection density between knowledge points [3].

This project aims to develop a visualization platform that can run in a browser, allowing students to connect knowledge points in a graphical way during review and quickly grasp the overall context. The platform is developed based on the Flask framework—this Python Web programming framework has the advantages of being lightweight and flexible for expansion, and can efficiently support the development of Web-based visualization platforms [4]. At the same time, the platform integrates ECharts components to realize the interactive display of knowledge graph data. ECharts has rich chart types and strong interactivity, which can well meet the needs of visualizing the relationships between knowledge points [5].

In addition, the project adopts embedded Neo4j and built-in JRE technology, enabling the system to run independently without installing additional databases or configuring the environment. This reduces the difficulty of deployment and facilitates its use in computer classrooms, laboratories, and even personal computers [6]. The overall architecture of the platform follows the design idea of distributed Web services based on Flask, which can ensure the stability of the system during access [7].

1. Related Research and Technology Selection

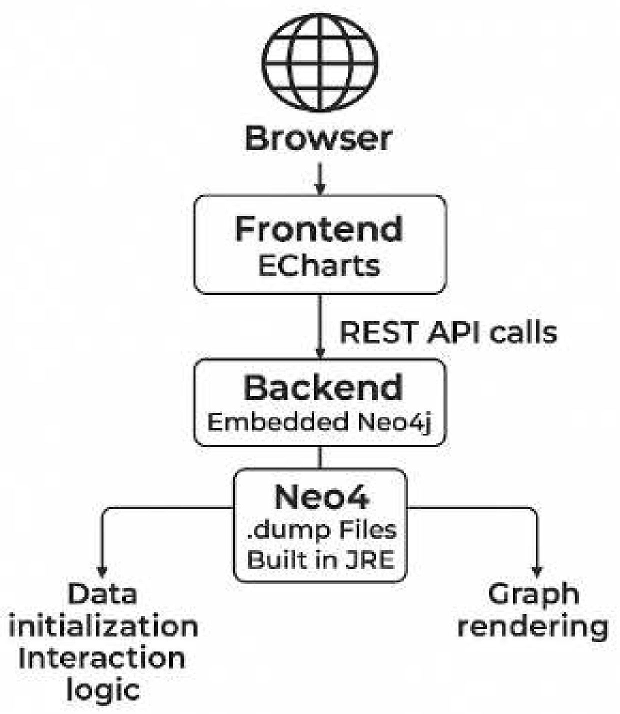


Fig1, System architecture of the knowledge graphvisualization platform

Knowledge graphs originate from research on the Semantic Web and ontology, and use graph structures to represent entities and their relationships. In 2012, Google first proposed the concept of "knowledge graph" and applied it to search engine result optimization, which greatly promoted the development of knowledge graphs. In recent years, knowledge graphs have been widely used in fields such as search, recommendation, intelligent question answering, medical auxiliary diagnosis, and educational resource management.

In educational scenarios, knowledge graphs can help teachers construct course structures, assist students in understanding course logic, improve learning efficiency, and also help narrow the gap in concept understanding and enhance knowledge transfer capabilities [1]. The system architecture of the knowledge graph visualization platform designed in this study is shown in Fig 1. This architecture takes the core needs of "knowledge graph construction - data storage - visual display - user interaction" as the main line, and divides the system into four core layers: the data layer, the service layer, the visualization layer, and the user layer. Among them, the data layer is mainly responsible for storing structured knowledge point data through the embedded Neo4j database; the service layer relies on the Flask framework to provide background services such as data query and relationship calculation; the visualization layer uses ECharts components to realize the graphical display of knowledge graphs; the user layer provides a simple and easy-to-operate browser interface for students and teachers [7]. Meanwhile, this architecture also refers to the design idea of display-type websites based on the Flask framework, emphasizing the simplicity and usability of the user interface while ensuring functional integrity [8].

In terms of graph database selection, this project uses Neo4j. It is a native graph database that uses a node-edge model to store data and provides the Cypher query language, which can efficiently execute complex pattern matching queries. Compared with relational databases, Neo4j has obvious advantages in multi-hop relationship queries, making it very suitable for handling complex concept connections in textbooks [9]. Relevant research on the application of ECharts in data visualization also points out that combining graph databases with visualization tools can maximize the value of structured data, which further confirms the rationality of the "Neo4j + ECharts" technology combination in this project [10].

In this project, each knowledge point is stored as a node, and the dependency, inclusion, or derivation relationships between concepts are stored as edges. This storage method facilitates subsequent knowledge point tracing and path querying. By using embedded Neo4j, the database can be started directly inside the application and can quickly restore data through .dump files. Combined with built-in JRE packaging, the system can run independently without external dependencies, making it convenient for users to deploy quickly [6].

2. Technical Design and Implementation

The overall system adopts a frontend–backend separation architecture, as illustrated in Fig 2.The backend is responsible for embedded database initialization, REST API services, and exception handling; the frontend focuses on data fetching, knowledge graph rendering, user interaction, and logging. The system is packaged with a built-in JRE to ensure cross-platform execution without requiring additional Java installation.

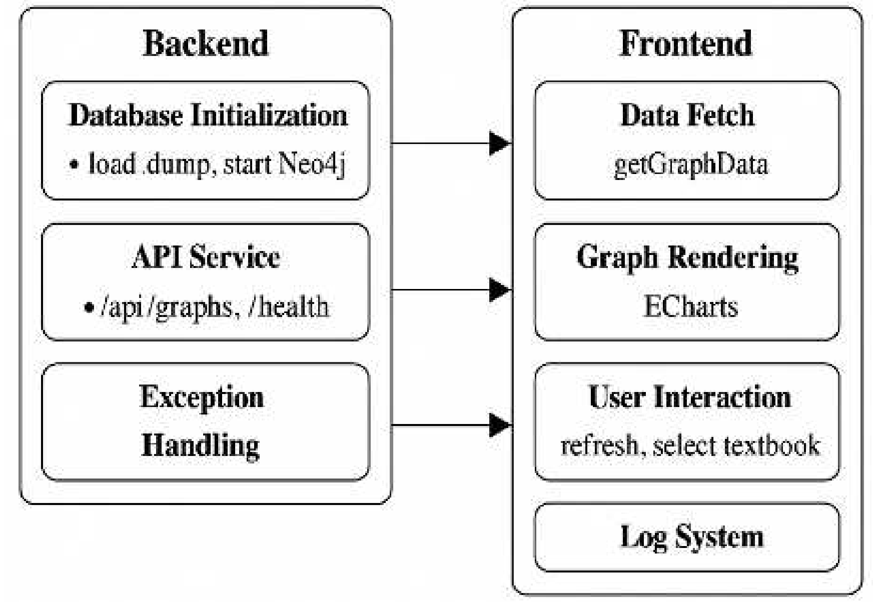


Fig 2：Technical Design and Implementation

2.1. System Architecture
The system adopts a frontend-backend separation model. The backend is responsible for starting the embedded Neo4j database, loading .dump files, and providing REST APIs, while the frontend calls these APIs and renders the results using ECharts . The system is packaged with a built-in JRE, ensuring cross-platform operation and eliminating the need for users to install a separate Java environment. you can see all of them

2.2. Database Initialization
When the backend starts, it first checks whether the .dump file exists. If the file is found, the database is restored using Neo4j's Graph Database Factory and Database Management Service, which support transaction management and query optimization. If the file is missing, the user is prompted to regenerate the data.

2.3. Data Interaction Logic
The backend provides two primary interfaces: /api/graphs/{tag} loads knowledge graph data by textbook and returns node-edge JSON objects, while /health performs system health checks and returns information about database status, number of nodes, and number of edges. Exception handling is applied to all interfaces to ensure that services respond gracefully even in case of incorrect data input .

2.4. Frontend Data Fetching
On the frontend, the asynchronous function get Graph Data(tag) is implemented through the window. Api Client object. It constructs a request URL based on the input tag and fetches data using the Fetch API . If the request fails, it attempts to parse the JSON error message and throws an exception to be caught by the caller. After obtaining data successfully, the function validates that all edges contain source and target fields. Missing type or relationship fields trigger a warning log for debugging purposes.

2.5. Graph Rendering Module
The core rendering function, render Knowledge Graph(data), processes nodes and edges, converts IDs to strings, and normalizes node attributes such as size and type. For edges, the type field is preferred as the display label; if it does not exist, the relationship field is used, and if both are missing, the label defaults to "Association". The ECharts instance is initialized with parameters such as title, tooltip, and legend. A force-directed layout is adopted with carefully tuned node repulsion and edge length to ensure readability. Edge labels are displayed with translucent backgrounds, and a resize event listener ensures responsive rendering.

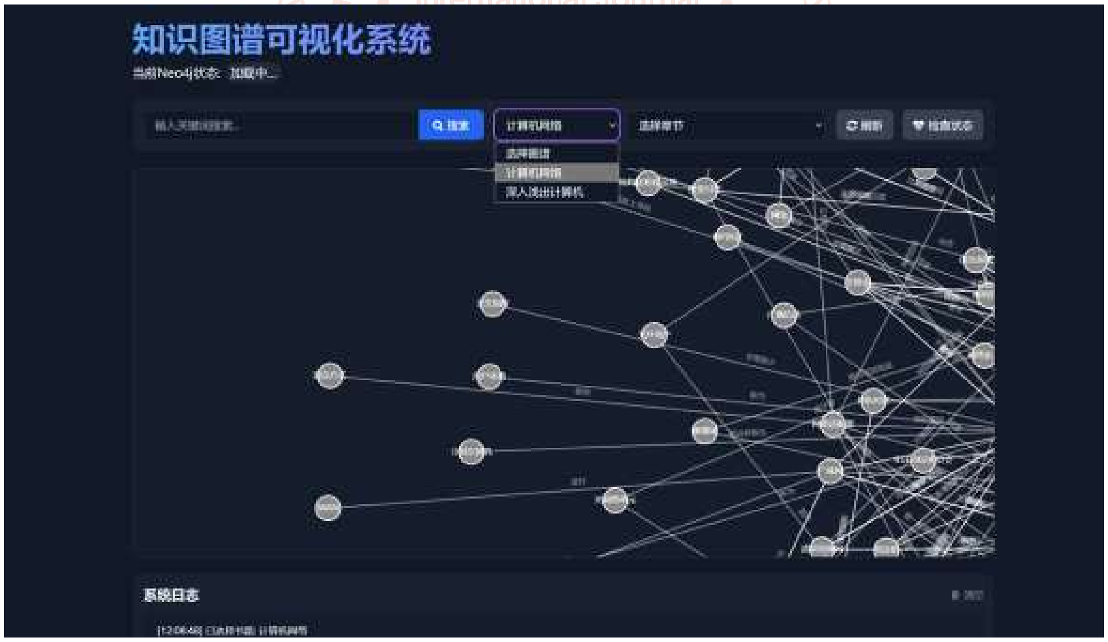
2.6. Logging and User Interface
The frontend layout includes a title bar, a textbook selection control, a graph display area, and a log output area. Users can refresh the graph or call the /health endpoint to check database status. The log system is implemented

through addLog(message), which appends timestamps and supports automatic scrolling for easy debugging. Button interactions provide visual feedback, improving the user experience.

3. Function Verification and Test Results

To fully verify the functional integrity, operational smoothness, and practical teaching adaptability of the textbook knowledge graph visualization system, the testing work adopted the "full user usage process" as its core logic and established a multi-dimensional testing framework covering identity verification, core function operations, and abnormal scenario response. Specifically, this framework includes nine core modules: login and registration, data loading, node interaction, chapter switching, node search, edge label display, refresh function, exception handling, and main page operations. For each module, targeted test cases were designed based on practical usage scenarios such as teachers' lesson preparation and students' knowledge organization. The effects of key operation interfaces are illustrated with diagrams to ensure the test process is reproducible and the test results are traceable and verifiable.

As the core operation entry for users, the main page integrates three core function entrances: "select knowledge graph", "switch chapters", and "entity search" (see Fig 3 for the interface effect). Clicking the "select knowledge graph" button triggers a textbook switching pop-up window, which supports quick switching between the two preset textbooks. After switching, the system automatically clears the current knowledge graph cache and loads the basic knowledge graph data of the corresponding textbook, with no obvious lag during the loading process. After selecting the target textbook, users can further switch between different chapters of the textbook through the "chapter selection" drop-down menu at the top of the page. The test covered all chapters of the two textbooks, and no loss or confusion of the association between knowledge graph nodes and edges occurred after chapter switching. The entity search box on the right side of the page supports two query modes: precise query and fuzzy query. After entering keywords, the system provides real-time feedback on matching results. Among them, the precise query can directly locate the target node and highlight it with a red border. In this test, 20 groups of different types of keywords (including core knowledge points, example question names, and formula numbers) were selected, and the query accuracy rate reached 100%.



[12:06:48] 已选择书籍: 计算机网络

Fig 3：Main Page Display

In the verification of data loading and interaction functions, the complete knowledge graph data of the two textbooks was used as the test sample. After loading, it was confirmed that the matching degree of node quantity and attribute information with the original data both reached 100%. In the node interaction test, hovering the mouse over any node displays complete information in the details panel; when dragging a node to adjust the layout, the edges adapt in real time; the zoom operation with the mouse wheel is smooth without lag; and hovering the mouse over an edge highlights and indicates the two connected nodes (see Fig 4). In addition, in the refresh function test, clicking the refresh button reloads the current knowledge graph data to ensure data synchronization with the background. The test results of all modules meet the preset performance and functional indicators, fully verifying the overall functional effectiveness and operational stability of the system.

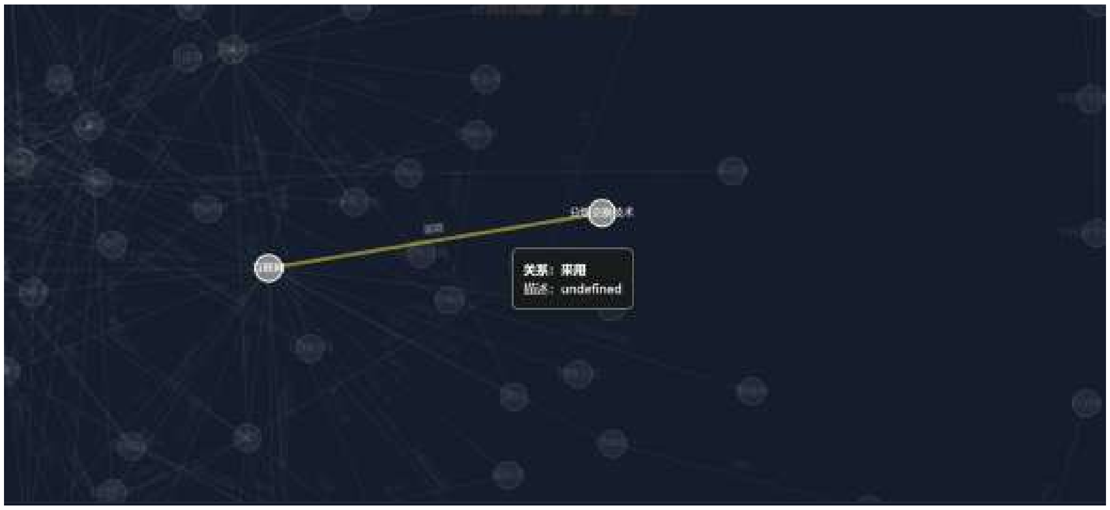


Fig 4：Relationship Highlight Display

4. Conclusion

This paper implements a textbook knowledge graph visualization system based on embedded Neo4j, covering the entire process of data collection, database construction, backend API development, frontend rendering, and interaction. The system can transform abstract knowledge points into a visualized network graph, helping learners form an overall understanding. The packaging solution of embedded Neo4j + .dump files + built-in JRE significantly reduces the deployment threshold; users can run the system without additional configuration, improving its applicability in teaching and demonstrations.

References

[1] LIU M Y, LIU X S, YANG W C, et al. Design and Research of IT Technology Learning Knowledge Graph System Based on Neo4j[J]. Modern Information Technology, 2025, 9(16): 113-119. DOI: 10.19850/j.cnki.2096-4706.2025.16.020.

[2] Wang, Z. Y., & Zhang, C. H. (2016). Design and implementation of data visualization analysis component based on ECharts. Microcomputer & Its Applications, 35(14), 46-48+51. https://doi.org/10.19358/j.issn.1674-7720.2016.14.015.

[3] Li, Z. G., Chen, Y., Zhang, X. Y., et al. (2014). A hierarchical structure visualization method based on force-directed layout. Computer Simulation, 31(03), 283-288.

[4] Ye, F. (2015). Research on Flask, the latest Python web programming framework. Computer Programming Skills & Maintenance, (15), 27-28. https://doi.org/10.16184/j.cnki.comprg.2015.15.010.

[5] Fan, L. Q., Gao, J., & Duan, B. X. (2023). Data visualization system for domestic popular tourist attractions based on Python+Flask+ECharts. Modern Electronics Technique, 46(09), 126-130. https://doi.org/10.16652/j.issn.1004-373x.2023.09.024.

[6] LIU W, AN X, YAN P F, et al. Research on the Construction of Knowledge Graph for China's Information Innovation Industry Based on Neo4j[J]. Science and Technology Management Research, 2025, 45(14): 151-159.

[7] Xu, J. (2020). Research and application of distributed web service architecture based on Flask. Industrial Control Computer, 33(10), 97-98+101.

[8] Ma, X., & Wang, S. L. (2018). Design and implementation of a display-type website based on the Flask framework. Digital Technology and Application, 36(11), 137-138. https://doi.org/10.19695/j.cnki.cn12-1369.2018.11.73.

[9] WANG Y L. Comparative Study between Graph Database Neo4j and Relational Database[J]. Modern Electronics Technique, 2012, 35(20): 77-79. DOI: 10.16652/j.issn.1004-373x.2012.20.045.

[10] Cui, P. (2019). Application of ECharts in data visualization. Software Engineering, 22(06), 42-46. https://doi.org/10.19644/j.cnki.issn2096-1472.2019.06.011.