

Semantic Data Governance and Collaborative Architecture for LLM-Based Natural Language Analytics in Urban Rail Transit

Shiqi Xiong¹, Guodong Liu²

¹ School of Intelligent Manufacturing, Jiangnan University, Wuhan, Hubei, 430100, China

² Beijing Urban Construction Design & Development Group Co., Limited, Beijing, 100037, China

Abstract: Urban rail transit data are distributed across heterogeneous operational systems, where inconsistent semantics and fragmented metric definitions complicate the controlled use of large language models (LLMs) for natural language analytics. This study proposes a semantic data governance and collaborative analytics architecture for this setting. A lakehouse-based data foundation, a semantic middle layer, and a unified metric store are used to organize physical schemas, business terminology, and statistical definitions. Domain rules, network topology, and a controlled execution workflow are then introduced to constrain query generation and support cross-domain analysis. Two representative scenarios—power-supply fault assistance and passenger-flow-traction-energy analysis—are used to illustrate how data governance, knowledge augmentation, and execution validation work together. The contribution is an engineering architecture and its semantic governance mechanisms rather than a comparative evaluation of general-purpose LLM algorithms. The proposed approach provides a practical reference for integrating LLM-based analytics with existing urban rail transit data environments.

Keywords: Urban Rail Transit; Semantic Data Governance; Semantic Middle Layer; Unified Metric Store; Natural Language Analytics.

1. Introduction

Network-level operation changes the way urban rail data are used. A Network Operations Control Center (NOCC) often needs to examine train-operation, passenger-flow, power-supply, and equipment data together when assessing an operational situation or reviewing an event. In routine work, however, many queries are still performed through fixed reports or separate professional systems. These methods are adequate for predefined tasks, but become cumbersome when an analyst needs to compare information across systems or formulate an ad hoc question [1].

The difficulty is not only that the data are stored in different systems. Automatic Train Supervision (ATS), Supervisory Control and Data Acquisition (SCADA), Automatic Fare Collection (AFC), and Integrated Supervisory Control System (ISCS) were developed for different operational functions, so their tables, field names, statistical definitions, and update cycles do not naturally align. Interpreting an operator's term may therefore require several data objects to be considered together. "Skip-stop" and "service pattern," for example, belong to train-operation semantics, whereas "automatic transfer switching" and "peak-section passenger flow" rely on definitions maintained in different professional domains. Without explicit semantic mapping, the same natural-language request can point to different physical data structures.

LLMs can simplify interaction with such data, but the model itself cannot be treated as the source of operational semantics. Three issues are especially relevant in a rail-transit environment. First, business terms must be linked to governed schema objects and metric definitions. Second, line topology and equipment connectivity are maintained engineering relationships and should be obtained from structured data rather than inferred by the model. Third, any generated SQL

or analytical instruction must remain subject to authorization, audit, and exception handling. Work on intelligent question answering, multi-agent collaboration, knowledge-graph augmentation, and large-model techniques provides methods that can be reused for parts of this process [2–5]. These methods do not, by themselves, resolve the rail-specific requirements for consistent metric definitions, validated topology, cross-disciplinary associations, and controlled execution.

This paper therefore starts from the governed data environment rather than from the language model alone. Physical schemas, business terminology, and metric definitions are connected through a semantic middle layer and a unified metric store. Operating rules and network topology are brought into the analytical process when they are needed as contextual or deterministic constraints. Query generation is placed behind static audit, sandbox trial execution, and bounded error feedback. These elements are combined into a Text-to-Insight workflow intended for engineering use. The paper focuses on how the components are organized and maintained; it does not attempt to demonstrate that one general-purpose LLM is algorithmically superior to another.

2. Network-Wide Data Governance and Semantic Modeling

Reliable natural language analytics depends first on the quality and organization of the data supplied to the model. Urban rail data originate from heterogeneous systems including ATS, SCADA, AFC, and ISCS, and include structured transaction records as well as high-frequency status and event streams. Data volume grows with network expansion and sampling frequency. Conventional data engineering focuses primarily on ingestion, storage, and exchange; for LLM-based use, the platform must additionally represent business semantics, metric definitions, entity

relations, and the provenance needed to trace analytical results.

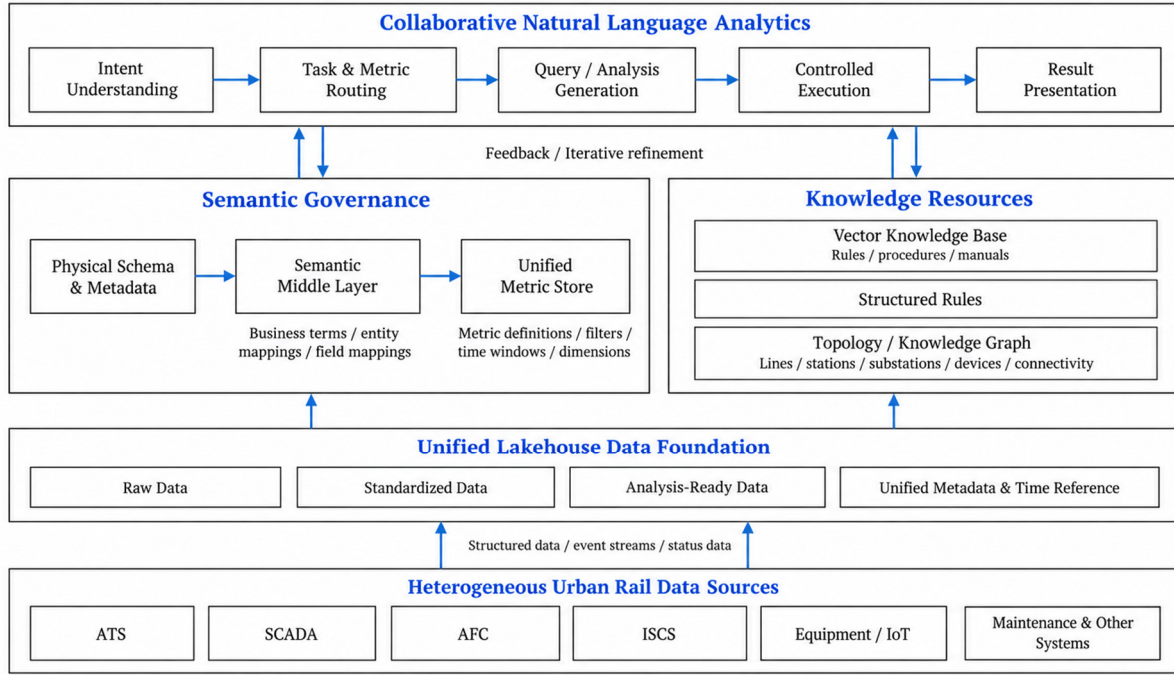


Figure 1. Network-wide data governance and semantic modeling architecture

2.1. Lakehouse Data Foundation and Heterogeneous Data Ingestion

The data foundation follows a lakehouse approach and uses unified metadata management to connect raw data, standardized data, and analysis-oriented data assets. Metadata records the data source, update mechanism, field meaning, and responsible business domain so that datasets can be consistently identified throughout later governance and analytical processes. Ingestion mechanisms are selected according to data characteristics: structured transaction data can be synchronized incrementally through change data capture (CDC), while high-frequency status and event data can enter through message queues or streaming channels. Unified metadata registration, layered storage, and consistent time references provide the basis for cross-domain analysis.

2.2. Semantic Middle Layer and Unified Metric Store

Physical database schemas are designed around system implementation and are rarely organized in the same way that operators describe a task. This is particularly evident in long-tail fields, abbreviations, and equipment identifiers; exposing them directly to an LLM can increase semantic ambiguity and field-selection errors [6]. The semantic middle layer is therefore placed between the source schemas and the analytical workflow. It does not alter the source tables. Instead, it records how tables, fields, entities, and business terms correspond under a defined business context. In practice, the mapping may depend on factors such as professional domain, line scope, or statistical use, so the relation is maintained as governed metadata rather than assumed from field names alone.

Physical metadata and business terminology provide the initial basis for semantic mapping, while manual validation determines which relations are accepted for use. An LLM can assist by proposing candidate labels or associations, but these suggestions are not treated as final semantics. Metric

management is handled in the same governance layer but serves a different purpose. Indicators such as morning-peak passenger flow and traction energy consumption must preserve their aggregation rules, filters, time windows, and dimensional associations. Once these definitions are registered, a natural-language request can reference the governed metric instead of rebuilding the calculation logic for each query.

These mappings and metric definitions also need to remain usable as the underlying systems change. For this reason, each governed object is linked to its data catalog entry, responsible discipline, version, and effective period. A field change, metric adjustment, or system migration can then be handled by updating metadata and retaining the previous version where necessary. This approach keeps the analytical layer less dependent on low-level database details and provides a traceable basis for explaining why the same request may produce different results under different data or metric versions.

2.3. Unified Knowledge Storage

Operating rules, emergency procedures, equipment relations, and other domain knowledge are organized through a combination of structured facts, a vector knowledge base, and a knowledge graph. Unstructured documents such as regulations, procedures, and manuals are retrieved as contextual evidence, while structured topology captures relations among lines, stations, substations, devices, and other assets. The two forms of knowledge serve different purposes: textual retrieval supplements business context, whereas graph relations and rules provide deterministic checks on entity connections and paths [7].

Knowledge resources also require version and applicability management. Procedures and equipment relationships may change after operating adjustments, line reconstruction, or asset replacement. Retrieved documents should therefore carry metadata such as version, applicable line, and publication date, while topology should remain linked to asset

identifiers and change records. This distinction helps reviewers determine whether an inconsistency originates from model interpretation or from an outdated knowledge version.

3. Collaborative Architecture for Natural Language Analytics

3.1. Overall Architecture

On top of the governed data environment, the Text-to-

Insight architecture separates natural language understanding, metric invocation, knowledge retrieval, query generation, and controlled execution into cooperating processing functions. The LLM is used primarily for intent interpretation, task organization, query planning, and result expression, while metric definitions, operating rules, and network topology remain external, governed, and maintainable. The architecture is organized into three functional modules: semantic alignment, knowledge-augmented reasoning with deterministic constraints, and controlled validation and execution, as shown in Figure 2.

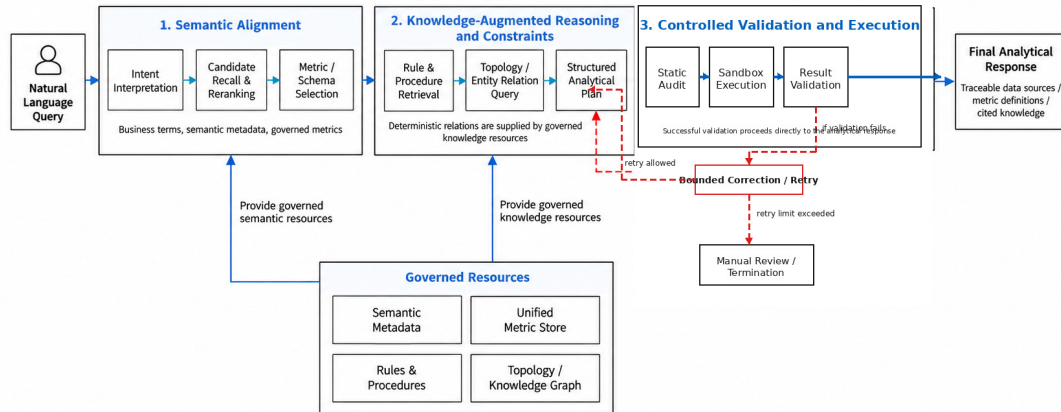


Figure 2. Collaborative architecture for natural language analytics

The semantic alignment module identifies user intent and locates relevant data resources. Standardized requests can directly invoke definitions in the unified metric store; long-tail or ad hoc requests instead construct a limited query context from semantic-layer metadata, business terminology, and candidate tables. The knowledge-augmented reasoning and constraint module retrieves operating rules and procedures while consulting line and equipment topology for deterministic relations. The controlled validation and execution module checks generated SQL or analytical instructions for syntax, authorization, and resource constraints, trials eligible statements in a sandbox, and permits only bounded correction before either returning a result or transferring unresolved cases for manual handling [8–10].

3.2. Semantic Alignment

Semantic alignment uses candidate recall followed by contextual reranking rather than attempting to expose the entire physical schema to the LLM. Table names, field names, metadata descriptions, and manually maintained business-term mappings are indexed to retrieve a candidate set for each user request. The recall stage is intentionally permissive: ambiguous field names, abbreviations, and terms reused across disciplines are distinguished with attributes such as entity type, business domain, line scope, and time granularity. Standard metric definitions and business context are then used to remove irrelevant candidates, reducing the effect of a single similarity score on the final query.

The reranking stage jointly considers the user request, candidate schema objects, standard metrics, and domain context to form a compact query context. Expressions such as “morning peak,” “skip-stop,” or “service pattern” should first be interpreted through maintained terminology and governed metric definitions. If sufficient high-quality examples become available, few-shot prompting or parameter-efficient fine-

tuning may be introduced to improve domain adaptation [11,12]. These techniques are implementation options rather than prerequisites of the architecture, and the contribution here is the governed two-stage alignment process rather than a specific encoder or fine-tuning framework.

3.3. Knowledge-Augmented Reasoning and Deterministic Constraints

Queries involving dispatching rules, power-supply relations, or network topology require constraints that should not be generated freely by the LLM. For rule- or procedure-related questions, relevant clauses are retrieved from the vector knowledge base and supplied as evidence. For spatial and equipment-connectivity queries, entities and paths are obtained from the structured knowledge graph. In a power-supply scenario, for example, maintained topology can identify the relation among a substation, feeder, and affected traction-supply section. These governed relations are used as deterministic constraints during reasoning and logical validation so that critical entities and paths remain consistent with the maintained network model. The purpose is to reduce path mismatches and omitted relations in long reasoning chains, not to replace professional control systems or operational judgment.

3.4. Robust Execution Loop

Generated queries are executed through a controlled loop consisting of static audit, sandbox trial execution, result validation, and bounded feedback, as illustrated in Figure 3. The workflow is intentionally framework-independent. Static audit checks the SQL structure, operation type, access scope, and resource limits. Write operations, unauthorized access, or queries without required restrictions are blocked or transferred for manual review. Practical implementations should also inspect scan scope, returned row count, execution time, and access to sensitive fields. Rate limits, timeouts,

masking, and read-only database accounts remain conventional database controls; the application-layer pre-execution audit is only a preliminary check and does not

replace database permissions, network security, or business authorization.

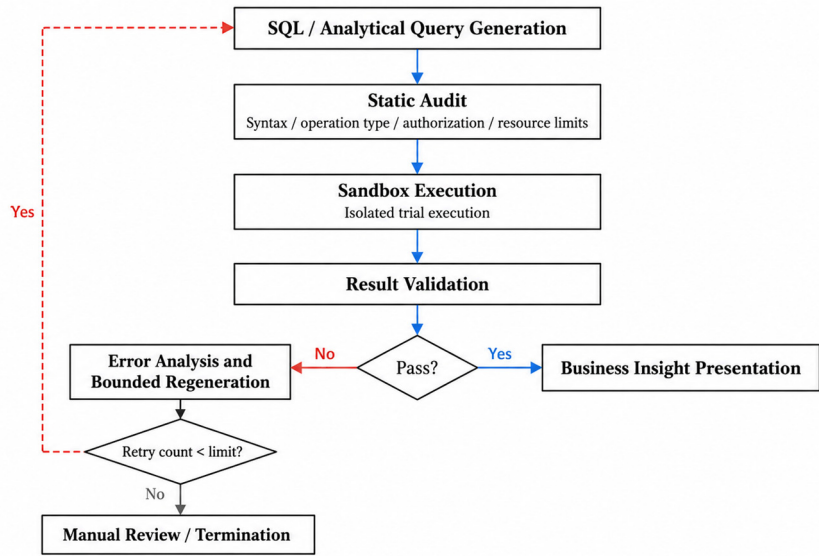


Figure 3. Controlled execution loop with bounded error feedback

Queries that pass the audit can be trialed in an isolated environment. When field ambiguity, syntax errors, or unmet constraints are detected, the system may regenerate or revise the candidate statement using the returned error information, but only within a predefined retry limit. If validation still fails, automatic execution is terminated and the request is transferred for manual handling. The loop is intended to improve executability, auditability, and traceability; it should not be interpreted as an absolute guarantee of business correctness or production safety.

4. Representative Scenarios and Implementation Feasibility

Two scenarios are used to show how the architecture behaves under different types of rail-transit questions. The first concerns a power-supply fault, where topology and operating rules are more important than open-ended language interpretation. The second links passenger flow with traction energy consumption and therefore depends more heavily on cross-domain semantic alignment. These examples are not used as an algorithmic benchmark; their purpose is to check whether the proposed components can be connected into a workable engineering process.

4.1. Emergency Support for Power-Supply Faults

Take a trip alarm associated with a switch in a main substation. After the SCADA alarm is received, an operator may start an analytical request with the alarm record and the relevant equipment identifier. The alarm on its own is not enough to describe the possible impact: switch position, protection status, automatic transfer switching status, current supply topology, and the applicable operating procedure may all be needed. The workflow therefore uses the alarm as an entry point and gathers these items from the governed data and knowledge resources. Its output is intended for review and assistance; control commands remain outside the scope of the proposed system.

Impact analysis begins with the equipment state rather than

with free-form reasoning. The switch position, protection status, and automatic transfer switching status are retrieved from the unified data model and associated with the corresponding equipment entities through consistent identifiers. These values define the operating state against which topology is checked. The corresponding nodes and connections are then retrieved from the maintained knowledge graph to identify traction substations or supply sections that may be affected. The LLM can organize the retrieved information and explain the result, but it does not independently infer physical connectivity.

Once the impact scope is established, relevant clauses are retrieved from the procedure and emergency-plan knowledge base and combined with the topology result into a structured recommendation. “Recommendation” in this context means information integration and explanatory support, not automatic actuation. Before presentation, the output is checked against rules and access permissions, sensitive data are masked where necessary, and the affected scope, cited clauses, and data sources are retained so that NOCC personnel can review the basis of the analysis.

4.2. Cross-Domain Interactive Analysis of Operational Data

As a second example, consider the request: “Analyze the relationship between passenger-flow peak periods and traction energy consumption on Line 1 during working days last week, and produce a brief report.” Several parts of this question cannot be translated into database conditions without prior business interpretation. The operational calendar determines which dates are working days, and the metric store defines what counts as a peak period. Passenger-flow records and traction-power metering or SCADA data may also come from different source structures, so they must be aligned to the same line and time dimensions before the relationship can be examined.

The system first narrows the data scope rather than generating a query against the full schema. Metadata retrieval provides candidate tables and fields related to the two business domains. The request, governed metric definitions,

and domain attributes are then used to remove candidates that do not match the required line, time range, or statistical meaning. The remaining objects form the schema context supplied to query generation. Because the business definitions are maintained separately from the generated query, later changes to a metric can be handled in the governance layer while the applicable historical version remains available for earlier analyses.

Using this restricted context, the query module constructs the joins and calculations needed for the analysis. The statement is not used immediately: it passes through static audit and sandbox trial execution as described in Section 3.4. Syntax problems, ambiguous fields, or failed constraints can be corrected only within the bounded retry process, after which unresolved requests are handed over for manual processing. A successful run produces the statistics and visual material used in the brief, together with the source tables, time range, metric version, and execution record required for later review.

For cross-domain analysis, interpretability depends not only on the wording of the generated explanation but also on whether the analytical result can be traced to its tables, filters, metric versions, and join relations. The result object should therefore retain query conditions, metric identifiers, the data time range, and execution logs. This makes it possible to locate an abnormal conclusion in the semantic mapping, metric definition, or source data rather than judging credibility from generated text alone. The scenario illustrates cooperation among the semantic layer, metric store, and controlled execution mechanism; it does not constitute quantitative evidence of algorithmic improvement. Actual accuracy still depends on source-data quality, metric definitions, and the completeness of business mappings.

4.3. Engineering Evaluation Dimensions and Validation Boundaries

Engineering evaluation should not rely on a single question-answering accuracy score. Data, semantics, knowledge, and execution should be evaluated separately so that source-data defects, model-interpretation errors, and execution-environment problems are not all attributed to the LLM. At the data and semantic layers, measurable indicators include business-term-field mapping accuracy, entity-normalization accuracy, and unified-metric hit rate. Knowledge-layer evaluation can examine the relevance of retrieved rules and the consistency of topology relations, while the execution layer can record query success rate, the proportion of manual intervention, blocked hazardous operations, and end-to-end response time.

These indicators require actual line data, desensitized engineering data, or an independent validation set. A power-supply fault test should retain equipment entities, topology paths, and cited rules; a passenger-flow-energy test should retain the time window, line scope, metric definition, and cross-domain join relations for error localization. Until measured data are available, these indicators should be treated as an engineering validation and acceptance framework rather than as claimed performance results. No unmeasured accuracy, millisecond-level latency, throughput, or safety conclusion is reported here, and engineering validation data should remain independent from algorithm-comparison data used in a separate methods study to avoid reusing the same evidence for different research claims.

5. Conclusion and Future Work

This study addresses the connection between heterogeneous urban rail data environments and LLM-based natural language analytics through an architecture that combines a lakehouse data foundation, a semantic middle layer, a unified metric store, hybrid knowledge resources, and controlled execution. The representative scenarios show how business language, physical data structures, knowledge evidence, and execution validation can be organized within one engineering workflow. The semantic and metric layers provide a governed entry point from operational expressions to data resources, rules and topology add traceable evidence, and static audit with sandbox trial execution constrains the use of generated queries. The conclusions concern architectural completeness and an implementation path; the scenario analysis is not presented as quantitative proof of algorithmic performance.

Future work will establish an independent engineering validation set using real-line or desensitized data and will measure term-mapping accuracy, metric invocation, cross-domain query completion, manual correction rate, and response time. Knowledge, metric, and data version management will also be strengthened, and different model and deployment choices can then be compared under the same governed workflow. This will provide reproducible evidence for practical deployment in urban rail transit rather than relying on architecture-level reasoning alone.

References

- [1] Chen, L., Yang, Y., & Gong, B. (2025). Security protection system for intelligent rail transit data: Challenges, technologies, and future evolution. *Urban Rapid Rail Transit*. (in Chinese)
- [2] Yao, Q., Yang, T., Yang, J., Wu, J., & Huang, M. (2025, November 24). Design and implementation of a multi-agent collaborative intelligent question-answering system for radio management. *Radio Communications Technology*. (in Chinese)
- [3] Liu, Y., Huang, W., & Zhou, S. (2025). A collaborative multi-agent architecture for risk intelligence perception and early warning in emergencies. *Library and Information Service*. (in Chinese)
- [4] Jia, Z., Zhang, J., He, H., Feng, S., & Liu, Y. (2026). Survey on mutually augmenting technologies and applications of large models and knowledge graphs. *Computer Science*, 53(8), 219–228. <https://doi.org/10.11896/jsjx.250700129>. (in Chinese)
- [5] Zhu, X., Yao, H., Liu, J., & Xiong, X. (2024). Review of the evolution of large language model algorithms. *ZTE Technology Journal*, 30(2), 9–20. (in Chinese)
- [6] Zhou, Y., Cai, P., & Dong, Z. (2024). Large language model knowledge management system. *ZTE Technology Journal*, 30(2), 63–71. (in Chinese)
- [7] Jiang, X., Zang, Y., Xie, Z., Ju, Y., Liu, Y., & Sheng, G. (2025). Application model of DeepSeek-like paradigm in fault diagnosis of power equipment. *High Voltage Engineering*, 51(8), 4006–4021. (in Chinese)
- [8] Kou, G., Tang, X., Chen, X., Wang, X., & Chang, J. (2025). A new paradigm of large-model collaborative digital-intelligent decision-making: Key mechanisms and future prospects. *Bulletin of National Natural Science Foundation of China*, 39(5), 728–735. <https://doi.org/10.3724/BNSFC-2025.07.14.0001>. (in Chinese)
- [9] Cui, L., Wang, Z., Bao, Y., & Zhao, H. (2026). A survey on question answering systems and evaluation in the era of large models. *Computer Engineering and Applications*, 62(5), 1–17.

- <https://doi.org/10.3778/j.issn.1002-8331.2503-0004>. (in Chinese)
- [10] Han, Y., & Wei, G. (2026). Overview of task optimization and domain application development of knowledge graph and recommendation algorithm. *Computer Engineering and Applications*, 62(10), 26–53. <https://doi.org/10.3778/j.issn.1002-8331.2505-0279>. (in Chinese)
- [11] Han, B., & Liu, T. (2024). Key technologies and applications of large models. *ZTE Technology Journal*, 30(2), 76–88. <https://doi.org/10.12142/ZTETJ.202402012>. (in Chinese)
- [12] Tian, H., Zhang, M., Chang, R., & Tong, X. (2024). A survey on large model training technologies. *ZTE Technology Journal*, 30(2), 21–28. <https://doi.org/10.12142/ZTETJ.202402004>. (in Chinese)