Scholar Publishing Group
International Journal of Neural Network
https://doi.org/10.38007/NN.2026.050111
ISSN 2790-2692 Vol. 5, Issue 1: 98-107

A Framework for LLM-Based Semantic Configuration Understanding and Automated Change Generation in Microservice Orchestration

Yunshu Zhang

Kit Circle, Austin, TX, 78758, US

Keywords: Large language model; Kubernetes; Microservice orchestration; Semantic configuration understanding; Automatic change generation

Abstract: This paper discusses some problems with inadequate semantic interpretation of configuration information in microservices orchestration environments, including complicated YAML specifications, increased costs from cross-service change coordination, lack of contextual restrictions on automatic corrections, etc. Builds an LLM-based semantic construction of understanding and automated modification generation in Kubernetes orchestration environments. According to the review of research articles in LLM software engineering, Kubernetes configuration security and microservice operation for three years ago, this paper puts forward a closed-loop mechanism of "intent parsing-context retrieval-constraint verification-change generation-canary release veri\-fication", empirically analysizes its popularities and security issues based on public-sourced survey results. In view of the present status and development direction in application at home and abroad, they have two main reasons to pursue them as follows. In combination with a large-scale language model, coupled with rule engines, policy knowledge bases and rollback mechanisms to enhance the consistency, interpretable ability and change security of configuration generation. Theoretical basis and practical support of Intelligent Operation & Management (IO&M) for platforms Construction and autonomous orchestration.

1. Introduction

As cloud-native infrastructure continues to advance in scale, heterogeneity and continuous delivery become increasingly prominent features; therefore, configurations for microservice systems are now composed of multiple layers that include Deployments, Services, Ingresses, ConfigMaps, Secrets, Helm charts, policy lists, etc. No more description text for configuration, but its effects on several dimensions will be directly impacted in the control plane, including elastic scaling, network openness, boundary permissions, and release pace. In recent years, with the continuous improvement of large language models in areas such as code completion, script generation and operation-and-maintenance questions, their development direction is getting closer to being one of the key stages of software engineering [1]. In addition, some scholars are now using

LLMs in the development of Operation & Maintenance (O&M) agents, automated repairs and architecture migration tasks [2].

Three reasons for the difficulties encountered when interpreting the configuration of this scenario: microservice orchestration. Configuration semantics are spread out over the layers of YAML, business context and runtime constraints respectively. The old rule-based matching method has no ability to build an expression map for users' intentions, resource items and runtime results. The second is that there are many defaults, inheritances and Cross-object dependences in the Kubernetes configuration. Excessive privileges, unreliability of images, exposure of port numbers, conflicts with resource usage due to insufficient permissions. Empirical studies show that security misconfigurations of open-source Kubernetes manifests are relatively prevalent [3]. Thirdly, if automatic change generation is disconnected from the system environment, it may produce a piece of syntactically valid but semantically invalid configuration patch; If this occurs during production release and leads to failure, excessive rollback due to human errors; Or accumulate behind-the-scenes threats.

Many types of research currently exist in this area. Some researchers have tested, using LLM technology, how well it can manage Linux servers; Also explored its command understanding capabilities and script generation skills. Some researches have integrated a large-scale model with ChatOps, low-code service orchestration, etc., for driving operation-and-maintenance action based on natural language [5]. Some researches have applied LLM to Intent-Driven Kubernetes Configuration Generation and Microservice Identification Tasks, thus Accelerating the Evolution of Architectural Structure Speedup. There are few studies on semantic configuration understanding and automatic change generation in a closed loop, lacking framework research for the integration of retrieval enhancement, policy constraints, risk scoring and gray-scale verification.

Therefore, this paper, entitled "LAMPS", begins by describing the current situation and deficiencies of it, then put forward an LLM semantic configuration understanding and automatic change generation scheme based on Kubernetes; And at present, there are some issues in its implementation approach. In addition, by using publically available statistical data to check whether the framework is feasible under real-world conditions to some extent.

2. Current Status Analysis of the Research Topic

2.1 Cloud-native orchestration has become the mainstream infrastructure for microservice governance.

Kubernetes was not a core function originally when creating the platform; However, currently it manages most of business Application deployments. According to a 2023 CNCF survey, as many as 66 per cent of potential or actual cloud-service consumers had already implemented Kubernetes in their systems for use and evaluations at present time; By 2025, over fifty percent of container-users will have adopted Kubesrfts into production-environment. The wide application of this platform has significantly increased the quantity of configuration items and may thus introduce errors in interpretation related to configuration meanings across multiple platforms.

The following Figure 1 has been created using public CNCF surveys' data. Kubernetes's utilisation rate reached as high as 82% in 2025 based on its use-rate of 66% in 2023; Therefore, currently no differences exist among applications' adoption rates of this technology; therefore, configuration generation and change management should take into account their effect on system stability, reduce the occurrence of failure caused by changes during releases, increase the efficiency of checking for compliance issues. With the enlargement of the platform, Operation Teams need to understand configuration meaning in detail and build an automated creation function for Security Change Instructions. As shown in Figure 1.

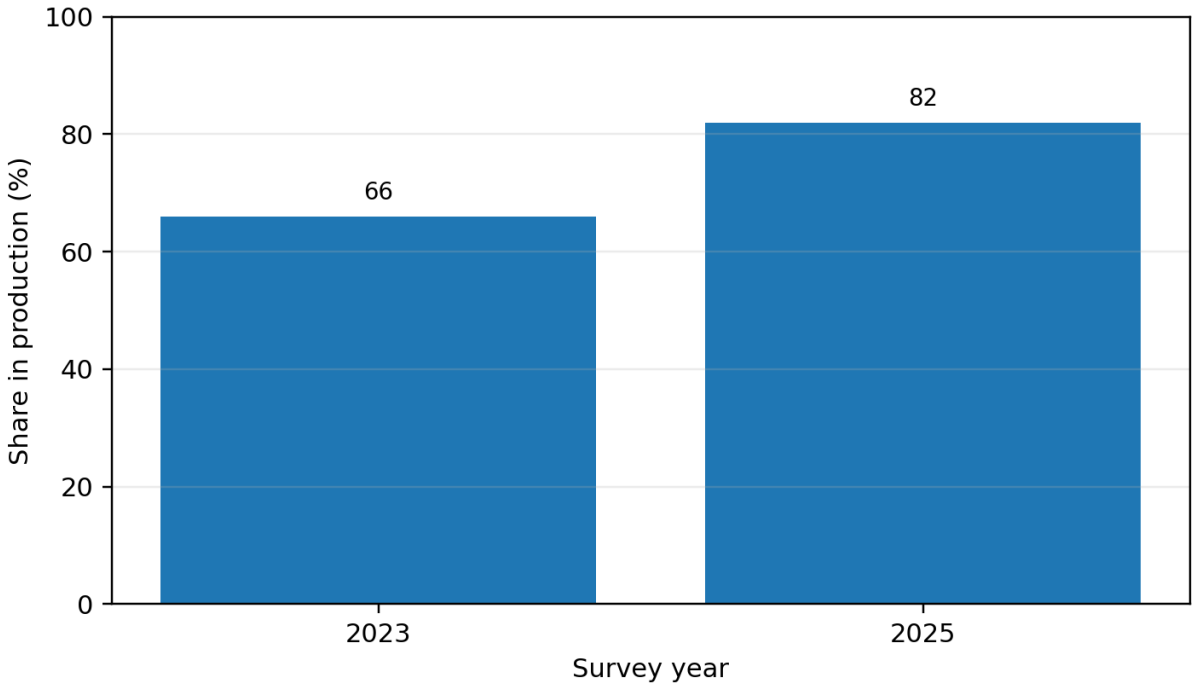

Figure 1, the trend of Kubernetes application in production environments has been reflected in many public surveys to date.

2.2 LLM is expanding from a code assistant to an operations and architecture assistant.

The applications of large language models in software engineering have gradually advanced from simple coding assistance to multiple fields involving requirement analysis, test automation, operation and maintenance implementation, etc., during this process. Microservices and configuration management-related automatic repair experiments with Ansible; Research on intent-based configuration generation for Kubernetes; Architectural decomposers such as microservice identification work; And research has also studied how to evolve configurably in software systems together. Based on this, it can be inferred that there are substantial limitations to the practical use of large language models for textual structuring conversion due to their weaknesses during rules definition, knowledge acquisition and risk analysis during subsequent interactions.

2.3 Security misconfiguration remains a high-frequency risk in orchestration environments.

Security misconfigurations are more prone to occur due to insufficiencies in the semantics of a Kubernetes environment. Based on the empirical study of open-source projects, problems such as exposed credentials, unsafe images, excessive use of resources caused by lack of limit definitions, and security defects arising from loose access control have emerged in Kubernetes manifest files [3]. Later investigation revealed that there exists an insurmountable discrepancy between the recommendation made by the community for enterprise configurations and their actual settings; That is, "being able to configure properly" does not necessarily mean having been correctly implemented. Errors in industry codebing; Leakage of sensitive dataInformation; A shortage of network ksafety Safety risk; The lackofnetworksecurityrisk; The absence fmalware Malignant programs)exist currently. As shown in Figure 2

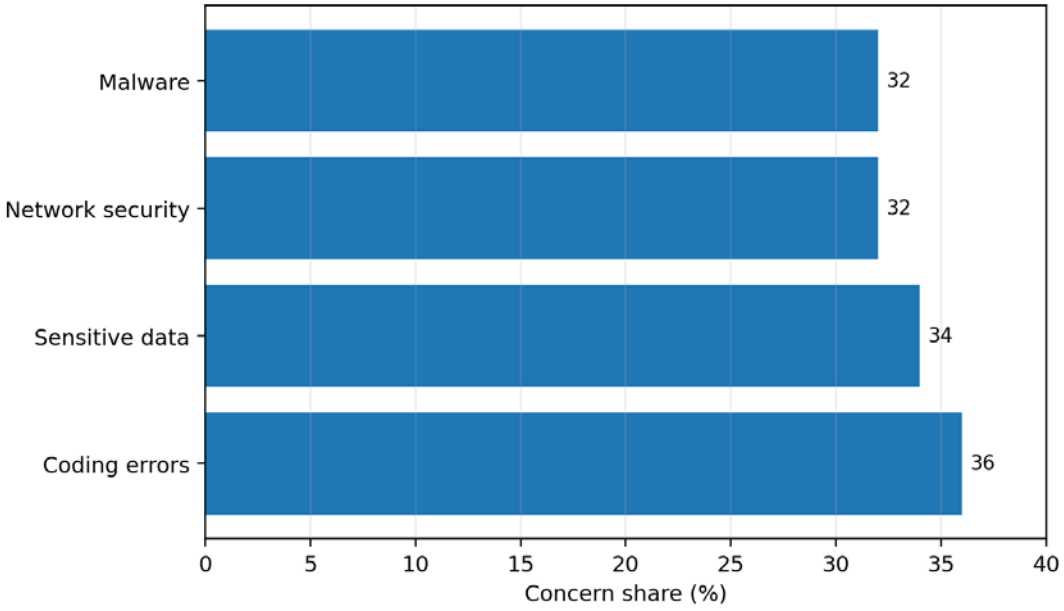

Figure 2, major security issues in container and Kubernetes environments.

Based on RedHat's "State of Kubernetes Security Report 2024", as shown in Figure 2. As can be seen from the above-mentioned research results of organizations' problem response mechanisms for digital trust risk, most businesses are still addressing multiple issues rather than specific problems. Therefore, only static rules cannot address complicated cross-layer semantic risk problems independently. The future intelligent configuration need to build an integrated Management System for natural Language Intent, Structured Policy, Real-Time Feedback and History Change Experience.

2.4 Microservice fault diagnosis and configuration change governance are beginning to merge.

Microservices systems have multiple links, numerous dependencies, asynchronous versioning, rapid fault diffusion, and thus the situation of configuration change being mixed with fault identification is quite common. Some relevant reviews have organised knowledge graphs, topology propagation, etc., of the diagnosis method for faults in microservices [11]. It is necessary to generate the patches automatically, at the same time provide operations-and-maintenance closed-loop functions, including identifying the extent of change impacts, executing a canary release, conducting verification checks, and providing information about whether it needs to be rolled back. If not, no matter how correctly the generated patch is syntactically, an endless series of exceptions will occur because of actual service dependency problems.

Table 1. Configuration Semantic Hierarchy at the Framework Level (text in Table is in English)

Layer	Input artifact	Semantic role	Typical risk
Intent	Ticket / ChatOps / Prompt	Change objective	Ambiguous instruction
Spec	YAML / Helm / Kustomize	Desired state	Field conflict
Policy	OPA / RBAC / NetworkPolicy	Constraint boundary	Privilege escalation
Runtime	Logs / Metrics / Events	Execution evidence	Silent failure

As shown in Table 1, among them are the intent layer, specification Layer, Policy Layer and runtime layer of Semantic Configuration Understanding. The traditional YAML parser only processes the fields of the specification level and cannot judge whether this change is aligned with the work-order intention, organisational policy or operational state. The Value of LLM is not to replace all the rules; instead, it creates a semantically intermediate representation and verifies this with rules via multiple layers of information integration, thus enhancing the fragmented nature of configuration interpretation.

Ensure that the changes in natural language correspond to the corresponding configuration items; therefore, we need to calculate their degree of semantical similarity through an Intent-configuration mapping relationship first. To determine if the object that needs modification is consistent with the user's current real requirements and not in error; Prevent improper patching from affecting normal operation.

$$Sim\_sem(I,C) = (v\_I \cdot v\_C) / (||v\_I|| \cdot ||v\_C||) \quad (1)$$

According to Equation 1, vI represents the change intention vector; vC denotes the candidate configuration fragment vector; and a higher value of Simsem indicates a more suitable work order semiotics and configuration environment. To rank resource candidates, find patches, reduce the search space under a multi-Manifest situation; To compare with other evaluation functions.

3. Raise questions

Based on the research and practice of industry application so far, currently there are four deficiencies in using LLM for micro-service orchestration. The fragmentation of semantic interpretation. Most of the existing methods still consider the generation result of an LLM simply as "text-to-YAML", without provisions for managing versions, dependencies, policy settings and revisions. Secondly, there is no adherence to the rule side. If the generated models only use natural language context for generation without considering organisational constraints like RBAC, image repository construction and resource management restrictions, as well as naming rules; Third, automatic changes lack risk grading. Different patches affect the company in different ways; modifying the number of replicas or NetworkPolicy is not subject to the same standard for risk assessment. Forthwith, the absence of verification or reversal processes exists. Most of the existing research is in the offline generation phase; there are no automated connections among canary releases, runtime failures, and rollback mechanisms yet.

The core issue is an inconsistency among the models' ability, system's capability. LM can handle meaning well and produce text that meets expectations; But there is an unstable mechanism for enforcing constraints in LLMs; Orchestration systems are strictly rules-based and cannot interpret implicitly operating annotations. Without the connection of an intermediate layer, a disconnect arises where model outputs are readable but unusable, and rules are executable but don't understand business objectives.

Thus, this paper believes that the intelligent change management system for microservice orchestration does not need to have a long prompt or large model, but needs to build a framework combining semantic understanding, rule constrains, risk evaluation and continuous verification. In particular, with the growing trend of platform engineering and autonomous operation, if LLm is not under a tight governance Loop in this closed environment to ensure security, comprehensibility and auditability of automatic modifications.

To illustrate the potential hazards brought about by a particular change in the platform's safety and reliability; A risk weighted-scoring method can integrate the probabilities of occurrence, extent of harm, organisational attitudes towards various types of risks within this system for assessment purposes.

$$R = \Sigma\, w\_i\, p\_i\, l\_i \quad (2)$$

According to Equation (2): The weight \(w_i \) of the \(i \)-th type of risk; Its probability \(p_i \); And the impact coefficient for each type of risk \(l_i \). Using this equation, we can judge the way of execution for the patch: direct execution, entry into grey-scale mode, or needing human intervention. Quantifying the risks associated with configuration errors to ensure that all automatic output data has some degree of trustworthiness.

Table 2 Compares the path-generation ability among various settings; that is, it translates English annotations into Chinese.

Method	Strength	Weakness	Suitable scope
Rule-only	Deterministic	Poor semantics	Compliance check
LLM-only	Flexible reasoning	Hallucination risk	Draft generation
RAG + Policy	Context grounding	Higher pipeline cost	Production change
Agent loop	Closed-loop correction	Complex governance	Large platform

Table 2: A single technical approach is insufficiently flexible or reliable for both. Pure rule-based methods have good repeatability but cannot well understand complex business intents; pure LLM path generation capabilities are strong but prone to problems such as lack of context and illusions. Combining retrieval enhancement with policy verification is more reasonable; it retains the merits of model-language reasoning while also including platform-level constraints and evaluation information.

4. Problem Solving/Strategies

4.1 Constructing a closed-loop framework for semantic configuration understanding and automatic change generation

Propose a closed-loop system made up of the following five parts. The first part of the processing function is the intent parsing module that extracts the meanings from work order, alarm or chatops command; then obtain target service, resource type, required time range, constraint conditions etc. The second one is the context Retrieval Module that retrieves relevant evidences from Git repositories, Configuration Templates, historical change Records, Policy Knowledge Bases and Runtime Observation Data. The third one is a patch Generation Module that utilises LLM to produce candidate Patches, change descriptions, and predicted effects. The fourth is the rule-validation module that calls, in sequence, Schema, OPA, admission-control policy, organisational specification to perform static filtering. The fifth one is grey-scale test module; it verifies whether these predictions are valid within a small scale of data or under low usage conditions before finalising them.

Given that there is an imbalance among generation quality, rule conformity, and costs of implementation, to express the goal of configuring automation comprehensively within the scope of objectives:

$$J = \alpha \cdot Sim\_sem + \beta \cdot C\_rule - \gamma \cdot C\_cost \quad (3)$$

In Equation (3), simsem is the semantic match degree; crule is the rule consistency coefficient; ccost reflects a resource overload such as regression verification delay or operation and

maintenance expenditure. Parameters α, β, and γ represent the organisational's preference levels. In the case of highly regulated areas like finance and Government Affairs, β's weight needs to be increased; In non-regulated grey test situations, The α factor might be raised a bit more to boost exploratory power.

4.2 Using retrieval enhancement and strategy knowledge base to suppress model illusions

Only models trained in general corpora do not have sufficient knowledge to understand organisational mirror whitelists, naming conventions, service dependencies, release schedules, etc. Therefore, to build a search-based knowledge base of repositories containing GitOps configuration files, Helm charts, security test policies, SLA guidelines, and issue tracking forms within the system. The objective is not to extend the duration of prompt repetition; rather, it is restricted by whether certain Words can be expressed in this process within organisational guidelines. For high-risk components, including secrets, RBAC, NetworkPolicies and Ingresses, use separate templates and validators to ensure their security.

4.3 Driving Change Release Decisions Through Success Probability Prediction

After determining that the candidates are suitable for use, one must decide which ones will actually be utilised in practice. As the result of change varies due to multiple causes, including service status, dependency structure topology, Image version and previous failures; Therefore, this paper suggests that use the Probability of success for evaluating whether a patch can be pushed out.

$$P\_success = 1 / (1 + e^{-z}) \tag{4}$$

Z can take the form of a sum of several features, for example: Semantic similarity, Rule pass rate; Historical successful cases of similar changes in-service health, rollback difficulty etc. If Psuccess is below the set threshold, then this operation will not be carried out automatically; alternatively, it can choose between manual verification or a more prudent grey-scale option. It can change "can be produced" to "can be handled safely".

4.4 Embedding Canary Validation and Rollback into the Continuous Delivery Chain

Finally, the purpose of auto-repair is to ensure that the system can be made accurate, unchanged and documented in its expected functions. Therefore, this system adds sandbox Deployment, Canary Release, Key metric monitoring and automatic Rollback to the delivery Pipeline. After applying the configuration patches, track errors rate, latency, resources consumption, pod failures, and policies denied by time intervals. When the threshold is exceeded, a rollback patch will be generated automatically, along with the failed reason. This will improve the knowledge base, provide a higher-quality negative sample training environment for subsequent applications of LLM.

The validation phase is used to determine how much the predicted results differ from the actual observations in terms of key performance indicators related to patches; The root-mean-square-error (RMSE) metric should be employed here.

$$RMSE = \sqrt{(\Sigma(y\_t - \hat{y}\_t)^2 / n)} \tag{5}$$

As shown in Figure 5-1, the relationship between the sample values yt,ŷt, which are obtained through a certain prediction method from the input data; The lower the value of RMSE indicates that the forecasted result most closely approximates the true condition. This metric can be used to assess the Quality of change under different prompter Strategies, Different Verification Links, And Different Knowledge Retrieval Granularity, and provide Quantitative references for continuous

Improvement Of The Framework.

Table 3 Key control points of the closed-loop framework (Text in the table is written in English)

Control point	Input	Output	Governance action
Intent parsing	Prompt / Ticket	Structured intent	Scope narrowing
Patch generation	Config context	Candidate patch	Syntax check
Policy validation	Patch + policy	Pass / Reject	Risk tagging
Canary verification	Live metrics	Observed effect	Rollback or release

From Table 3 can be observed that there are mainly automatic change objects to be governed among these types; They constitute multiple-connected models instead of single model nodes. Without intent resolution, the range of patching is too extensive; Without policy confirmation, output results fail to meet regulatory standards; And lack of canary deployment test leads wrong configuration to enter live systems directly. Set up audit records and individuals accountable for ensuring employees are informed of their responsibilities under the management of risks.

5. Conclusion

Firstly, this paper will discuss an initial understanding of configuration semantics and automatically generate security change records through LLM for microservice orchestration situations. Kubernetes becoming the standard of control planes, configuration governance has moved from a local issue of scripts to that on the platform level. A review of related literature, publicised surveys, and industry developments in recent years shows that although large language models have some semantic reasoning abilities for text, scripts, etc., operations; they cannot yet be used directly in production without further improvement in query accuracy, rule restrictions, and closed-loop verification.

Therefore, the proposed paper presents an LLM-based semantic configuration understanding and automatic change-generation framework for Kubernetes orchestration environment, outlining path implementations in terms of semantic alignment, risk scores, success probabilities analysis and canary rollbacks. According to these results, it can be seen that the best way for the future is to consider the model as a person under control. Through integration of natural language intent, configuration knowledge, organisational strategy and run-time feedback in one cycle to guarantee the safety and efficiency of an automated change generation process.

Further research needs to be carried out in multiple directions: building high-quality configuration semantic data sets and evaluation baselines; Exploring the interpretable mechanism of multi-agent collaborative behavior under complex change environments; And using formal verification together with LLm-generation to enhance the correctness and traceability of automatic changes.

References

[1] *Gao C, Hu X, Gao S, Xia X, Jin Z. The Current Challenges of Software Engineering in the Era of Large Language Models[J]. ACM Transactions on Software Engineering and Methodology, 2025, 34(5): 127.*

[2] *Sarda K, Ibrahim S, Varshney D, et al. Leveraging Large Language Models for the Auto-Remediation of Microservice Applications: An Experimental Study[C]//Proceedings of the 2024 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion. 2024.*

[3] *Liang, Q. (2026). Reconstruction of Commercial Building Space Reuse Mode Driven by Composite Business Types. International Journal of Engineering Advances, 3(1), 107-113.*

[4] *Gao, Y. (2026). The Role and Practice of Delaware Law in Global Cross-border M&A Transactions. International Journal of Law, Policy & Society, 2(1), 38-46.*

[5] *Gao, Y. (2026). Application of Delaware Corporate Law in Cross-Border M&A: Business Structures, Contractual Risk, and Case-Based Lessons. Economics and Management Innovation, 3(2), 128-136.*

[6] *Wang, N. (2026). Research on Evaluation and Optimization Models of Enterprise Resource Allocation Efficiency from a Data-driven Perspective. Advances in Computer and Communication, 7(1).*

[7] *Su, J. (2026). Research on Android Real-time Communication System Architecture and High-reliability Assurance Pathways Integrating AI-based Anomaly Detection Mechanisms. Engineering Advances, 6(2).*

[8] *Wu, Y. (2025). Software Engineering Practice of Microservice Architecture in Full Stack Development: From Architecture Design to Performance Optimization. Machine Learning Theory and Practice, 5(1), 64-75.*

[9] *Chen, J. (2026). Construction of a Cloud-Native High-Performance Service Engineering System for Real-Time Decision-Making Platforms.*

[10] *Wang, Z. (2026). Relationship between Supply–Demand Structures of Base Metals and the Evolution of Corporate Value.*

[11] *Wu, Y. (2025). Software Engineering Practice of Microservice Architecture in Full Stack Development: From Architecture Design to Performance Optimization. Machine Learning Theory and Practice, 5(1), 64-75.*

[12] *Wu, Y. (2025). Optimization of Generative AI Intelligent Interaction System Based on Adversarial Attack Defense and Content Controllable Generation.*

[13] *Qian, X. (2026). Supply Chain Collaboration Mechanisms Driven by Demand Orchestration in Omni-Channel Retail Environments.*

[14] *Chen, M. (2025). Research on Automated Risk Detection Methods in Machine Learning Integrating Privacy Computing.*

[15] *Sun, J. (2025). Quantile Regression Study on the Impact of Investor Sentiment on Financial Credit from the Perspective of Behavioral Finance.*

[16] *Wang, Y. (2025). Application of Data Completion and Full Lifecycle Cost Optimization Integrating Artificial Intelligence in Supply Chain.*

[17] *Wang, Z. (2026). Mineral Commodity Time-Series Cycle Modeling and Feature Learning Algorithm Based On Improved Transformer. Procedia Computer Science, 281, 1347-1356.*

[18] *Yu, X. (2026). Application of Time Series Data Analysis Algorithm Combined with Attention Mechanism in Growth Marketing User Lifetime Value Prediction. Procedia Computer Science, 281, 57-64.*

[19] *Wang, Y. (2026). Construction of Supply Chain Demand Forecasting Algorithm Based On Time Series Data Analysis and Improvement of Its Management Efficiency. Procedia Computer Science, 281, 484-493.*

[20] *Yin, J. (2025). Research on Financial Time Series Prediction Model Based on Multi Attention Mechanism and Emotional Feature Fusion. Socio-Economic Statistics Research (2025), 6(2), 161-169.*

[21] Chen, M. (2026). Real-Time Compliance Monitoring Engine Based on eBPF: Privacy-Enabled Data Tracking for Edge Computing. Procedia Computer Science, 281, 216-225.

[22] Chang, C. W. (2026). Privacy Infrastructure Vulnerability Mining and Automated Framework Based on Multimodal AI. Procedia Computer Science, 279, 702-711.

[23] Gao, Y. (2026). Research on the Design of Governance Structure for Private Equity Funds and the balance of GP and LP Rights.

[24] Liang, Q. (2026). Research on the Renovation Design Path for Enhancing the Efficiency of Mixed-Use Office and Retail Spaces. European Journal of AI, Computing & Informatics, 2(2), 163-170.

[25] Chen, J. (2026). Elastic Scaling and Stability Assurance Mechanisms for Distributed Systems under High-Throughput Scenarios.

[26] Zhu, P. (2025, December). Construction of Multi-Scale Biostatistical Analysis Framework and its Application in Biomedical Signal Feature Recognition and Classification. In 2025 5th International Conference on Mobile Networks and Wireless Communications (ICMNWC) (pp. 1-7). IEEE.

[27] Wang, Z. (2026). Exploration of the Application of Resource Circular Economy in the Agricultural Industry Chain.

[28] Huang, J. (2026). AI-Assisted Decision Support in Housing Policy Implementation: Distinguishing Deterministic Rules, Manual Review, and Heuristic Suggestion. European Journal of AI, Computing & Informatics, 2(2), 153-162.

[29] Yin, J. (2026, March). An Efficient Iterative Algorithm for Calibrating Agency MBS Estimation Parameters Based on Bayesian Optimization, Implemented in Python. In 2026 IEEE Madhya Pradesh Section Conference (MPCON) (pp. 1462-1467). IEEE.

[30] Zheng, H. (2026, April). Research on Cloud-Edge Collaborative Elastic Computing and Cost Optimization for High-Concurrency Scenarios. In 2026 IEEE 15th International Conference on Communication Systems and Network Technologies (CSNT) (pp. 1489-1496). IEEE.