Systems & Control Transactions

Original Research Article
Peer Reviewed Conference Proceeding
ESCAPE 36 - European Symposium on Computer Aided Process Engineering
Sheffield, UK. 21-24 June 2026

PSE PRESS

LLM-Based Intelligent Data Extraction System for Industrial Equipment

Zean Chen[a], Kaicheng Song[a], Lingyu Zhu[b], Anjan Kumar Tula[a]* and Xi Chen[a]

[a] State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, Zhejiang, China
[b] College of Chemical Engineering, Zhejiang University of Technology, Hangzhou 310014, Zhejiang, China
* Corresponding Author: anjantula@zju.edu.cn.

ABSTRACT

Data extraction and processing constitute the cornerstone of quantitative management and operational analysis in industrial process plants. However, most manufacturing facilities currently lack efficient data extraction systems, relying instead on engineers to manually write and execute database queries, which is time-consuming, error-prone, and inflexible when handling diverse data formats or large-scale equipment networks. To address these limitations, this work presents a novel Large Language Model (LLM)-based intelligent framework designed for data extraction and basic data of industrial equipment. The system integrates natural language understanding capabilities with process database schemas, enabling users to perform complex data queries and analyses through natural language prompts. Specifically, it can perform data mining, time-dependent analyses, equipment comparisons, and cross-period performance evaluations when process information is provided. By integrating the process context, the system can further interpret the analysis results to suggest potential operational conditions or abnormalities occurring in the system. Developed in collaboration with a process industry partner, the proposed system was evaluated using six months of real-world operational data from 10 heat exchangers. Experimental results demonstrate that the framework achieves an extraction accuracy exceeding 99% in the online mode and ranging from 92% to 96% in the privacy-preserving offline mode, with a reduced retrieval latency of approximately 2.64 seconds for local inference. Unlike generic chatbots, this framework targets process-specific reasoning, acting as a cognitive assistant that empowers users to interact intelligently with industrial systems. The findings confirm that LLM-assisted extraction effectively addresses the rigidity and inefficiency of conventional approaches while maintaining high reliability and scalability for industrial analytics.

Keywords: Large Language Model, exchangers, data extraction, prompt

INTRODUCTION

The pace of digitalization across various industrial sectors has positioned Large Language Models (LLMs) as a transformative technology capable of driving fundamental changes in Process Systems Engineering (PSE) and manufacturing operations. Following the public release of ChatGPT by OpenAI in late 2022, LLMs have attracted widespread attention due to their capabilities in comprehending, reasoning over, and generating sophisticated natural language. When combined with complementary technologies such as external tools, databases, and domain-specific knowledge bases, these models enable novel research directions and practical applications across diverse domains.

Nevertheless, the deployment of LLMs in industrial environments remains at an early stage and encounters considerable challenges. Process industries are characterized by wide variations in unit operations, safety regulations, and exceptionally stringent requirements for reliability and precision; any erroneous prediction or misinterpreted analysis can result in severe economic losses, safety incidents, or environmental damage. One of the most prominent limitations of current LLMs is their ten-

dency to produce "hallucinations, " i.e., plausible-sounding but factually incorrect or fabricated outputs. To counteract this issue, researchers have explored several mitigation strategies, including fine-tuning on industry-specific datasets [1], retrieval-augmented generation (RAG) [2], integration with structured knowledge graphs for fact-checking and long-term contextual grounding, and hybrid neuro-symbolic approaches that combine language models with deterministic reasoning engines.

To date, most LLM-based research in process systems engineering and related fields has concentrated on qualitative and knowledge-intensive tasks, such as professional question-answering systems, fault diagnosis support, process flowsheet generation, reaction mechanism prediction, molecular or catalyst design assistance, and educational tools for thermodynamics or unit operations [3–6]. Notable examples include cognitive assistants for troubleshooting in the textile industry using GPT-3.5 [3], in-context learning to support user queries [4], root-cause analysis in semiconductor fabrication [5] and multimodal reasoning frameworks such as PaLM-E [6].However, before moving to the reasoning and analysis phase, it is essential to have a precise and flexible data extraction system. In most manufacturing facilities, there is a high daily demand for data extraction across various process parameters and timeframes. This necessitates dynamic data retrieval and basic analyses, such as data anomaly detection and trendline identification. For now, contemporary LLMs are not inherently equipped to directly ingest or query large volumes of raw industrial time-series data stored in relational databases. Engineers and operators must still rely on complex database systems that require advanced SQL proficiency or specialized software interfaces. This technical barrier significantly increases the time and effort needed for data extraction, delays insight generation, and ultimately hampers agile, data-informed decision-making on the plant floor.

The present work seeks to close this gap by leveraging the natural-language understanding and generation strengths of LLMs to create intuitive interfaces between process engineers and industrial databases. Rather than requiring users to write structured queries or navigate rigid dashboards, the system allows practitioners to interact with operational data through ordinary conversational prompts. Specifically, We focus on the data extraction process for industrial equipment, selecting the heat exchanger as a primary case study with real industrial data.

The objective of this research is therefore to enable process engineers, shift supervisors, and plant managers to perform complex data retrieval, trend analysis, cross-unit comparisons, and basic performance diagnostics far more efficiently than with conventional methods, ultimately improving real-time supervision, early anomaly detection, and overall production effectiveness. Moreover, the proposed system architecture is intentionally designed to be modular and generalizable, offering a reusable framework that can be adapted to quantitative data challenges involving other unit operations (distillation columns, reactors, compressors, dryers, etc.). The principal contributions of this study are as follows:

1. Dual-Mode Architecture: A flexible system design that supports both an Online Mode (utilizing high-capacity cloud-based LLMs for superior reasoning depth) and an Offline Mode (local deployment of smaller open-source models to guarantee strict data privacy and control in sensitive industrial settings).
2. Robust Prompt Engineering with Self-Correction: Specialized prompting strategies, including a closed-loop self-correction mechanism for SQL generation, that substantially reduce hallucinations and improve reliability even when using resource-constrained local LLMs.
3. Systematic Decomposition of Complex Analytical Queries: A structured methodology that breaks down nested, multi-step, and interdependent engineering questions into simpler, sequentially solvable subtasks, thereby enabling accurate handling of realistic, real-world process data analysis scenarios.

METHODOLOGY AND SYSTEM FRAMEWORK

System Architecture

The overall system design prioritizes flexibility and adaptability through a dual-mode architecture that incorporates distinct Online and Offline modes. This approach effectively addresses the diverse requirements of industrial applications, balancing high computational performance against strict data privacy constraints while supporting different data storage formats. Users can explicitly select the desired mode depending on their operational priorities and security policies.

In the Online Mode, the system leverages cloud-based APIs to access high-performance Large Language Models, such as GPT-4o. This configuration minimizes local hardware requirements and takes full advantage of the model's extensive 128k token context window, enabling it to process highly complex instructions and maintain coherent, long-running conversational histories. By contrast, the Offline Mode is essential for industrial environments that enforce strict data sovereignty and prohibit any transmission of proprietary process data to external networks. In this mode, open-source Large Language Models are deployed and executed entirely on local workstations, ensuring complete isolation of sensitive information. However, local deployment is inherently lim-

ited by available hardware resources. For example, a typical industrial workstation equipped with 16–32 GB of system RAM and a mid-to-high-end consumer GPU (providing 8–24 GB of VRAM) is generally restricted to running quantized 7B-parameter models (commonly 8-bit quantization). While quantization enables feasible inference on such hardware, it inevitably reduces reasoning depth and makes these models less capable of handling complicated, multi-step prompt logic compared to their full-precision, cloud-based counterparts.

To mitigate the performance disparity between the two modes, the system implements two complementary function-calling protocols: a JSON-based format and an SQL-based format. The JSON format supports rich, hierarchical, and expressive function definitions with detailed parameters and intermediate reasoning steps, but it requires strong semantic understanding and precise output formatting from the model. The SQL format, on the other hand, provides a more rigid, structured, and directive interface that aligns closely with predefined database schemas, making it significantly easier for smaller-parameter local models to generate correct and executable statements. Accordingly, the system defaults to the flexible JSON-based protocol in the high-performance Online Mode, while the Offline Mode defaults to the SQL-based protocol to better match the capabilities of constrained local models. A fallback mechanism is integrated to automatically attempt the alternative protocol if the default fails (e.g., due to syntax errors, semantic misalignment, or execution failure). Users also retain the ability to manually override the default and select their preferred processing method at runtime. The overall system architecture is illustrated in Figure 1 and is organized into three tightly integrated layers:

1. Storage Layer: Manages the underlying operational data, supporting both lightweight SQLite databases for structured time-series records and Excel files for legacy or ad-hoc datasets commonly found in industrial settings.
2. Data Processing Layer: Acts as the central orchestration engine of the system (detailed in Figure 2). This layer receives natural-language queries, translates them into structured executable statements according to the active mode and protocol, invokes the appropriate data retrieval and computation functions, processes the returned results, and generates coherent natural-language responses enriched with engineering context and derived insights.
3. Presentation Layer: Serves as the user-facing interface, enabling text query input, audio query input, mode selection and display of conversational history. This layer ensures intuitive interaction and full traceability of the analytical process.

This modular and layered design guarantees scalability, maintainability, and robustness, allowing the system to deliver efficient, personalized, and context-aware data analysis tailored to the complexities of real-world industrial processes.

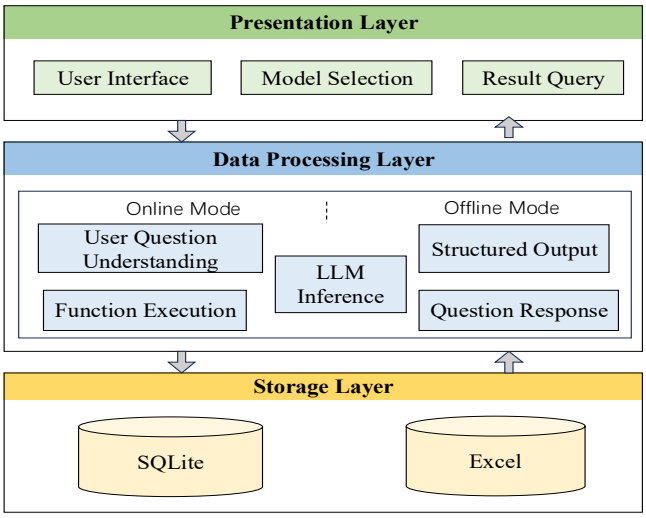

Figure 1. The architecture of the system framework.

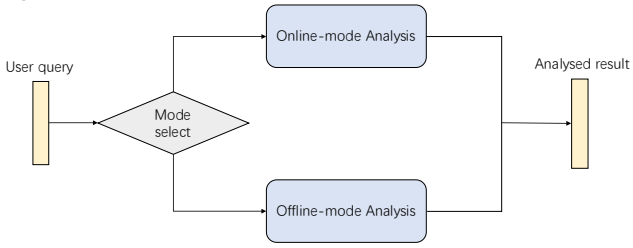

Figure 2. The flowsheet of the Data Processing Layer.

Prompt engineering

In the domain of PSE, effectively bridging the gap between complex industrial queries and precise machine execution requires robust prompt engineering. Standard instructions often fail to capture the strict logical dependencies and engineering-specific constraints inherent in process parameters. To overcome this limitation, the proposed system implements targeted prompting strategies designed to structure and guide the model's reasoning process in a manner that mirrors the systematic workflow of experienced process engineers.

For complex analytical tasks, the system employs Chain-of-Thought (CoT) prompting [7], which compels the model to explicitly illustrate its intermediate reasoning steps before producing a final answer. For example, when diagnosing a potential heat transfer anomaly, the model is guided through a logical sequence such as: 1. Retrieve and verify inlet and outlet temperatures; 2. Calculate the temperature difference; 3. Compare the result against predefined operational limits; 4. Identify any deviations and their possible implications.

This step-by-step reasoning closely replicates the

diagnostic approach of a human process engineer, significantly reducing logical errors in performance analysis, improving result reliability, and ensuring full transparency of the decision-making process for operators and supervisors.

In addition, the system utilizes Decomposed Prompting to effectively handle nested and interdependent engineering queries [8]—for instance, "Find the maximum temperature difference during the period of highest flow rate." A single, monolithic prompt frequently confuses the model in such cases due to the entangled dependencies. To address this, the framework breaks the complex request into independent, sequentially solvable sub-tasks. In the above example, the system first instructs the model to: "Identify the time period during which the flow rate reaches its maximum value." The output of this sub-task (the identified time window) is then passed as a precise constraint to the subsequent prompt: "Within the identified time period, compute the maximum temperature difference across the heat exchanger."

By isolating each engineering sub-task and enforcing sequential dependency, the system achieves high accuracy for individual variables before synthesizing them into a final, comprehensive report. This decomposition strategy proves particularly effective for managing the intricate, interdependent data relationships that are characteristic of chemical process operations.

Dual-Mode Strategy

The framework adopts a dual-mode strategy that tailors the data retrieval and execution mechanism to the specific inferential capabilities and constraints of the underlying language model. In the Online Mode, the system employs a JSON-formatted function-calling protocol optimized for high-performance cloud-based models. This structured format is inherently well-suited to Chain-of-Thought architectures, as it enables reasoning steps and intermediate results to be explicitly encapsulated within distinct key-value pairs. By mapping logical sub-steps to clearly defined JSON fields, the schema enforces systematic decomposition of complex industrial queries into discrete, interpretable units, greatly enhancing both the precision of the analysis and the transparency of the reasoning trace. Moreover, a specialized placeholder mechanism allows the output of one function call to be directly passed as input to a subsequent operation, thereby supporting the multi-hop reasoning required to resolve deeply nested engineering questions. The detailed patten is shown on the Figure 3.

Conversely, the Offline Mode relies on a SQL-based approach, which provides a more rigid and prescriptive structural framework better aligned with the syntactic and reasoning limitations of smaller, locally deployed models. Because local models frequently struggle with precise SQL syntax generation, especially under quantization and memory constraints, the system incorporates a robust self-correction mechanism (illustrated in Figure 4). When an initial SQL query fails to execute successfully against the database, the system automatically captures the exact error message returned by the SQLite engine and feeds it back to the model together with the original invalid query and the user's natural-language intent. This closed-loop feedback enables iterative refinement: the model is explicitly instructed to focus solely on correcting syntax and schema errors while preserving the analytical objective. The process repeats until a valid, executable SQL statement is produced or a predefined maximum number of correction attempts is reached. This mechanism substantially increases reliability in resource-constrained environments, ensuring dependable data retrieval even when using 7B-parameter models with limited reasoning depth.

```
{ "Action": <Function Name 1>,
 "Parameters": {
   <arg_1>: <value from user query>,
   <arg_2>: <value from user query>},
 "label": <label1_name>ش
  "equipment_id": <equipment_id>},
{ "Action": <Function Name 2>,
 "Parameters": {
   <arg_1>: <value from user query>,
   <arg_2>: $<label1_name>$},
 "label": <label2_name>,
  "equipment_id": <equipment_id>}
```

Figure 3. An example of how to map sub-steps via function calling. The second function takes the output of the first function and utilizes a placeholder during the return process, which is then executed by the code to replace the placeholder with the actual value.

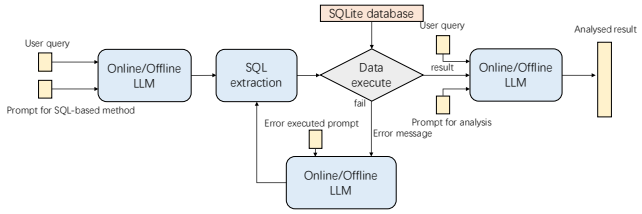

Figure 4. Framework design for SQL-based data analysis.

System Interface

The user interface is carefully engineered to deliver an intuitive, efficient, and accessible experience for industrial operators, engineers, and managers. It is implemented as a desktop application using the Qt framework and Python, providing a robust and cross-platform primary interaction point for the system.

A key design goal is to enable seamless mode

switching: users can toggle between Online and Offline computation modes directly from the main interface, depending on immediate operational needs, such as prioritizing maximum reasoning capability in non-sensitive settings or enforcing strict data sovereignty during critical production periods. To accommodate diverse working environments (e.g., noisy control rooms, hands-free operation, or office-based analysis), the interface supports dual-channel input methods: standard keyboard-based text entry for precise query formulation, and voice input powered by the Whisper automatic speech recognition model, allowing operators to issue natural-language commands hands-free while monitoring equipment or performing field tasks. The resulting interface is illustrated in Figure 5.

The interface maintains a persistent, scrollable chat-style history that displays both user queries and the system's full responses in chronological order. This conversational log preserves context across extended analytical sessions, enables users to revisit earlier results, and supports verifying of the interaction flow—an important requirement for traceability and validation in industrial decision-making processes. To further enhance system robustness, the interface is equipped with error-handling capabilities that automatically detect out-of-scope queries and provide corrective prompts to users.

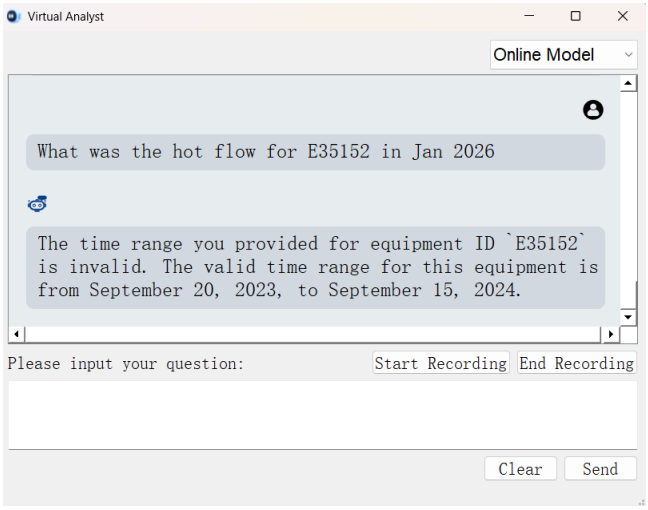

Figure 5. The interface of the data extraction system. This figure shows an example of the system's error-handling capability when encountering an out-of-bounds timeframe request.

INDUSTRIAL CASE STUDY: HEAT EXCHANGER PROCESS DATA

Dataset

The proposed system was rigorously validated using a comprehensive, real-world operational dataset obtained through close collaboration with a plant in process industries. The dataset spans six consecutive months of continuous production records and covers ten distinct shell-and-tube heat exchanger units operating under typical plant conditions. The dataset integrates key measured thermodynamic and process parameters, including: Inlet and outlet temperatures (hot and cold streams), Mass flow rates of both process fluids, Derived performance indicators, such as the Overall Heat Transfer Coefficient (U) and Logarithmic Mean Temperature Difference (LMTD).

All records are stored in a structured relational database environment (SQLite) designed to faithfully replicate the schema and access patterns of a standard industrial information management system. This setup ensures that evaluation results reflect realistic query execution behavior encountered in actual process plants. To guarantee reproducible and representative performance measurements for the Offline Mode under practical hardware constraints, all local inference experiments were conducted on a high-end workstation equipped with an NVIDIA RTX 5090 GPU (32 GB VRAM) and sufficient system RAM to support quantized 7B-parameter models without excessive swapping. This configuration represents a realistic upper bound for on-premises deployment in many industrial facilities as of 2026.

To systematically assess the system's natural-language understanding, logical reasoning, and data extraction accuracy, a dedicated benchmark dataset was constructed comprising 300 natural-language queries. These queries were carefully stratified into three complexity categories, with 100 queries per category, to probe different aspects of model capability:

1. Simple queries: Direct, single-parameter retrieval tasks (e.g., "What is the average cold-side outlet temperature for exchanger E35112 during January?").
2. Parallel queries: Requests that require simultaneous extraction and reporting of multiple independent parameters without interdependency (e.g., "Report the average flow rate, inlet temperature, and overall heat transfer coefficient for unit E35142 over the past month.").
3. Nested queries: Multi-step reasoning tasks in which the result of one computation serves as a constraint or input for a subsequent operation (e.g., "Count the days in January where the UA for E35112 is greater than the average.").

This stratified benchmark provides a robust and comprehensive framework for evaluating not only overall extraction accuracy and inference latency, but also the effectiveness of the self-correction mechanism and error-handling behavior across varying levels of analytical difficulty.

Conversational Examples

To evaluate the practical utility of the model, we utilized the aforementioned benchmark dataset consisting of 300 test cases, which are categorized into simple, parallel, and nested queries. Our system is designed to support four core operational capabilities that are essential for plant personnel to flexibly extract and analyze data: data retrieval, trend analysis, cross-unit comparisons, and basic performance diagnostics. Focusing on these four practical aspects, we conducted a series of case studies. The representative conversational interactions between the user and the system are illustrated in Figures 6, 7, 8 and 9.

1. Data Retrieval: This foundational capability enables users to extract specific historical or real-time process parameters without requiring knowledge of database structures or query languages. Shift supervisors can quickly obtain current operating values for critical equipment, while process engineers can retrieve statistical summaries such as averages, maxima, or cumulative totals over custom time windows. Beyond simple lookups, the system supports conditional queries (e.g., periods where a parameter exceeded a threshold) and arithmetic combinations of multiple variables, allowing users to perform ad-hoc calculations on the fly. By transforming manual, error-prone reporting into instantaneous, on-demand access, this capability accelerates routine monitoring, daily reporting, and troubleshooting investigations.
2. Trend Analysis: Beyond single-point extraction, the system allows users to examine how process variables evolve over time. Operators can identify gradual performance drifts such as fouling trends in heat exchangers by querying time-series patterns of key indicators like overall heat transfer coefficient or temperature differentials. Engineers can analyze cyclical behaviors (e.g., daily or batch cycles) or compare distinct operational phases (e.g., before and after maintenance, or across different feedstock grades) to assess intervention effectiveness. The system can also compute rates of change, detect monotonic trends, and flag sustained deviations from baseline behavior. This capability supports proactive decision-making by transforming raw time-series data into actionable insights about process stability, equipment degradation, and long-term performance trajectories.
3. Cross-Unit Comparisons: To support system-level oversight, the system enables simultaneous evaluation of performance across multiple equipment units. Process engineers can benchmark the efficiency of parallel units to identify underperforming assets, validate consistency across identical process lines, or assess the impact of differing maintenance schedules. Plant managers may request comparative summaries such as top-performing units by efficiency or units with the highest deviation from setpoints to prioritize maintenance activities and allocate resources effectively. The system can also aggregate data across groups of units (e.g., all exchangers in a specific area) to provide fleet-level statistics. By abstracting the complexity of multi-table joins and conditional aggregations, this capability facilitates holistic monitoring and supports data-driven decisions about equipment utilization, operational parity, and capital planning.
4. Basic Performance Diagnostics: As a practical extension of data extraction, the system provides preliminary diagnostic support through univariate anomaly detection. Operators can swiftly identify when a measured parameter deviates beyond established statistical thresholds (e.g., three-standard-deviation limits), flagging potential malfunctions without requiring advanced analytical training. Engineers can investigate the timing, duration, and magnitude of such deviations to correlate them with external events, operational changes, or emerging fault patterns. The system can also highlight recurring anomalies across multiple units or time periods, helping to prioritize deeper root-cause analysis. While not a substitute for detailed modeling, this capability serves as an effective tool, enabling rapid detection of abnormal conditions and focusing expert attention on areas most likely to require intervention.

Together, these four capabilities exemplified in the representative conversational interactions shown in Figures 6 (data retrieval), 7 (trend analysis), 8 (cross-unit comparisons), and 9 (basic performance diagnostics) demonstrate how the system bridges the gap between complex industrial data stores and the diverse informational needs of plant personnel, enabling more agile, informed, and collaborative operational decision-making.

RESULTS AND DISCUSSION

Performance Evaluation

We conducted tests in both online and offline modes to verify the system's robustness across different operational scenarios. For this evaluation, two primary metrics were established: Execution Accuracy and Inference Latency. Accuracy is emphasized as the paramount indicator, given that in safety-intensive process industries, any hallucination or incorrect data retrieval can lead to flawed decision-making and potential operational hazards. Meanwhile, Inference Latency is critical to assessing the system's viability for real-time interaction, particularly when deployed on resource-constrained local hardware for immediate anomaly detection.

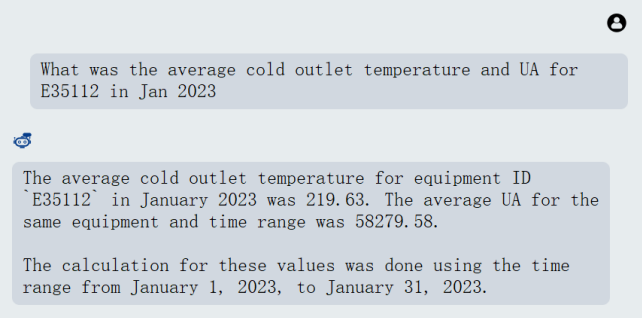

Figure 6. Dialog example of Data Retrieval.

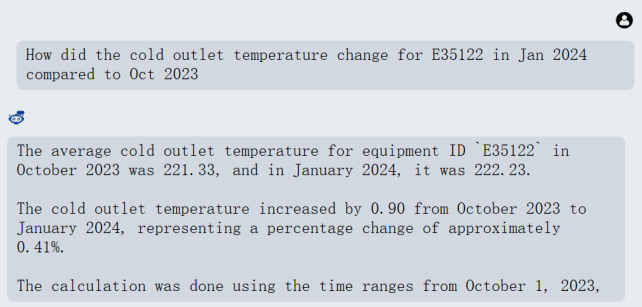

Figure 7. Dialog example of Trend Analysis.

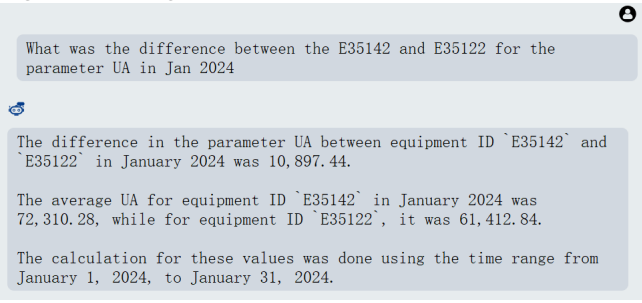

Figure 8. Dialog example of Cross-Unit Comparisons.

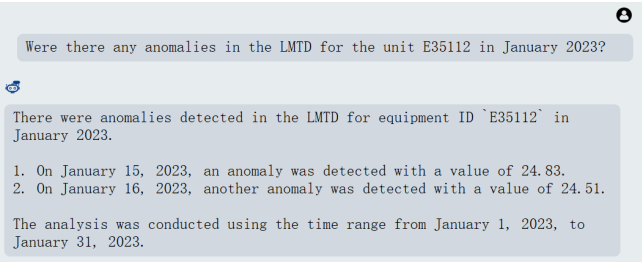

Figure 9. Dialog example of Basic Performance Diagnostics.

To have a more quantitative result, A benchmarking was conducted across Simple, Parallel, and Nested query categories. As detailed in Tables 1 and 2, the Online Mode achieved near-perfect accuracy, exceeding 99% across all query types. This superior generalization is attributed to the high-parameter count of the cloud model, which effortlessly managed complex nested logic. However, this high fidelity involved a latency trade-off, with average response times ranging from 3.90s to 5.09s due to network overhead. Conversely, the Offline Mode delivered commendable performance, with accuracy rates ranging from 92% for simple queries to 96% for parallel queries. Notably, the local deployment significantly outperformed the online configuration in processing speed, averaging an inference time of 2.64s for simple tasks. This characteristic makes the Offline Mode highly suitable for time-sensitive edge applications where data privacy and rapid response are paramount. The primary sources of error in the offline setting were identified as minor formatting inconsistencies and occasional parameter hallucinations (e.g., incorrect column name generation), which the self-correction mechanism largely mitigated but did not entirely eliminate.

Table 1: Evaluation results across various query categories in Online Mode.

Dataset	Accuracy	Average time cost
Simple question	100%	3.90s
Parallel question	99%	5.09s
Nested question	99%	4.96s

Table 2: Evaluation results across various query categories in Offline Mode.

Dataset	Accuracy	Average time cost
Simple question	92%	2.64s
Parallel question	96%	2.95s
Nested question	93%	5.19s

Furthermore, to scientifically validate the contribution of specific prompt engineering components, an ablation study was performed using the Online Mode as a baseline. The results, presented in Table 3, demonstrate that the full prompt configuration yields the optimal balance of precision and efficiency.

1. Chain-of-Thought Guidance: Removing CoT resulted in a moderate increase in inference time (approx. 60%) and a marginal dip in accuracy. This suggests that explicit reasoning steps are crucial for streamlining the model's internal logical processing.
2. Few-Shot Examples: The removal of few-shot demonstrations led to catastrophic performance loss, especially for Nested queries where accuracy hit 0%. This confirms that few-shot examples are essential for teaching the complex JSON schemas required for multi-hop reasoning.
3. Rules and Constraints: While removing explicit constraints did not severely impact accuracy, it significantly increased latency. Without predefined bound-

Table 3: Ablation study results quantifying the individual contributions of Chain-of-Thought (CoT), Few-Shot prompting, and Constraints to the system's overall performance in Online Mode.

Prompt config	Question Type	Accuracy	Format Acc.	Function Acc.	Time Acc.	Parameter Acc.	Average time cost
Full prompt	Simple	100%	100%	100%	100%	100%	3.90s
	Parallel	99%	100%	100%	99%	100%	5.09s
	Nested	99%	100%	100%	99%	100%	4.96s
w/o CoT	Simple	93%	100%	100%	93%	100%	4.62s
	Parallel	94%	100%	100%	94%	100%	8.29s
	Nested	96%	100%	100%	96%	100%	9.15s
w/o Few-Shot	Simple	53%	55%	100%	98%	100%	6.51s
	Parallel	60%	63%	100%	97%	100%	7.07s
	Nested	0%	0%	100%	98%	75%	8.22s
w/o Rules	Simple	99%	100%	100%	99%	100%	5.60s
	Parallel	99%	100%	100%	99%	100%	11.54s
	Nested	98%	100%	100%	98%	100%	9.17s

aries, the model required more iterative "thought" cycles to arrive at the correct structured output.

Comparison with Generic Frameworks

To further validate the necessity of the proposed specialized architecture, a benchmarking study was conducted against LangChain [9], a widely adopted framework for building LLM applications. To ensure a fair comparison, the generic framework was evaluated under identical hardware constraints using the same Mistral-7B model deployed in the Offline Mode. The comparative analysis reveals that the proposed direct-calling framework significantly outperforms the generic LangChain implementation in both execution accuracy and inference efficiency.

Initial experiments utilizing LangChain's default SQL chain resulted in a critical failure to generate valid executable queries, yielding a 0% accuracy rate across all test categories (Table 4). This failure was primarily driven by dialect mismatches, where the generic prompts defaulted to standard SQL syntax incompatible with the strict requirements of the local SQLite environment. Additionally, the generic approach induced frequent schema hallucinations, where the model attempted to query non-existent columns or functions prohibited by the local schema definition.

To isolate the impact of prompt quality, the robust, optimized prompts developed for our system were subsequently injected into the LangChain pipeline. Despite this enhancement, performance remained significantly suboptimal, with accuracy rates failing to exceed 50% even for simple queries (Table 5). This persistent performance gap suggests that LangChain's internal abstraction layers—which often encapsulate user instructions within generic system roles or append significant token overhead—tend to dilute the strict instruction adherence required by smaller, parameter-constrained local models. In contrast, our proposed framework's direct "Generation-Correction" loop provides the fine-grained control necessary to enforce syntax constraints on 7B-parameter models.

Table 4: Evaluation results based on the LangChain framework using its default SQL prompt templates.

Dataset	Accuracy	Average time cost
Simple question	0%	4.46s
Parallel question	0%	5.35s
Nested question	0%	5.54s

Table 5: Evaluation results based on the LangChain framework using self-designed prompts.

Dataset	Accuracy	Average time cost
Simple question	47%	5.60s
Parallel question	9%	4.34s
Nested question	22%	5.05s

Furthermore, regarding computational efficiency, the LangChain implementation demonstrated an average inference latency approximately 50% higher than our custom framework. These results underscore that while generic frameworks aid rapid prototyping, specialized industrial edge applications necessitate the bespoke optimization presented here to ensure reliability and real-time performance.

CONCLUSIONS

This study addresses the critical bottleneck of manual data extraction in process industries by introducing a novel, LLM-based Intelligent Analysis System specifically tailored for heat exchanger. Through a flexible Dual-Mode Architecture, the proposed framework effectively resolves the trade-off between complex reasoning capabilities and strict data privacy requirements.

The Online Mode, leveraging JSON-based Chain-of-Thought prompting, demonstrated exceptional fidelity in handling nested engineering queries, achieving over 99% accuracy. Conversely, the Offline Mode ensured complete data sovereignty while delivering reliable performance (92–96% accuracy) and superior real-time response (~2.64s). This was achieved through a specialized SQL "Generation-Correction" mechanism designed to mitigate the hallucinations common in smaller local models. Furthermore, comparative benchmarking against LangChain confirmed the necessity of this specialized architecture, as the proposed system significantly outperformed the generic framework in both syntax validity and inference efficiency under identical hardware constraints.

Beyond immediate technical performance, this work has broader implications for the democratization of industrial data analytics within the context of Industry 5.0. By abstracting complex database interactions into intuitive natural language, the system serves as a domain-specific cognitive assistant that significantly lowers the technical barrier for process engineers and operators. This capability not only facilitates efficient knowledge transfer and workforce empowerment but also bridges the critical gap between raw operational data and actionable decision-making. Ultimately, this framework provides a scalable foundation for human-AI collaboration, paving the way for future integration with digital twins and autonomous anomaly detection systems across the process industry.

ACKNOWLEDGEMENTS

The financial support of the "Pioneer" and "Leading Goose" R&D Program of Zhejiang (Grant No. 2024C01028) is gratefully acknowledged.

AUTHOR IDENTIFIERS

Author ORCIDs:
Zean Chen: 0009-0003-6972-8155
Kaicheng Song: 0009-0002-3812-9582
Xi Chen: 0000-0002-0571-6177
Lingyu Zhu: 0000-0002-3412-3805

REFERENCES

1. Naveed H, Khan AU, Qiu S, et al. A comprehensive overview of large language models. arXiv preprint arXiv:2307.06435 (2023) https://doi.org/10.48550/arXiv.2307.06435
2. Lewis P, Perez E, Piktus A, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. Adv Neural Inf Process Syst 33:9459-9474 (2020) https://doi.org/10.48550/arXiv.2005.11401
3. Kernan Freire S, Foosherian M, Wang C, Niforatos E. Harnessing large language models for cognitive assistants in factories. Proceedings of the 5th International Conference on Conversational User Interfaces :1-6 (2023). https://doi.org/10.1145/3571884.3604313
4. Kernan Freire S, Wang C, Foosherian M, et al. Knowledge sharing in manufacturing using large language models: User evaluation and model benchmarking. arXiv preprint arXiv:2401.05200 (2024) https://doi.org/10.48550/arXiv.2401.05200
5. Ezukwoke K, Hoayek A, Batton-Hubert M, Boucher X, Gounet P, Adrian J. Big GCVAE: decision-making with adaptive transformer model for failure root cause analysis in semiconductor industry. J Intell Manuf 36:2423-2438 (2024). https://doi.org/10.1007/s10845-024-02346-x
6. Driess D, Xia F, Sajjadi MSM, et al. PaLM-E: An embodied multimodal language model. Proc Mach Learn Res 202:8469-8488 (2023) https://doi.org/10.48550/arXiv.2303.03378
7. Wei J, Wang X, Schuurmans D, et al. Chain-of-thought prompting elicits reasoning in large language models. Adv Neural Inf Process Syst 35:24824-24837 (2022) https://doi.org/10.48550/arXiv.2201.11903
8. Khot T, Trivedi H, Finlayson M, et al. Decomposed prompting: A modular approach for solving complex tasks. Proc Int Conf Learn Represent (2023) https://doi.org/10.48550/arXiv.2210.02406
9. LangChain AI. LangChain: Build context-aware reasoning applications. https://github.com/langchain-ai/langchain (2025)