Graph-Based Consistency Verification for A Reliable Knowledge-Augmented Question-Answering System

Han Yan
School of Compute Science
Shanghai Jiao Tong University
Shanghai, China
han-yan123@sjtu.edu.cn

Dan Liang
Sichuan Gas Turbine Establishment
Aero Engine Corporation of China
Chengdu, China
1328297346@qq.com

Shenghe Li
School of Compute Science
Shanghai Jiao Tong University
Shanghai, China
lishenghe@sjtu.edu.cn

Hongxin Yan
School of Compute Science
Shanghai Jiao Tong University
Shanghai, China
shironine@sjtu.edu.cn

Han Yu*
School of Compute Science
Shanghai Jiao Tong University
Shanghai, China
han_yu@sjtu.edu.cn

Hongming Cai
School of Compute Science
Shanghai Jiao Tong University
Shanghai, China
hmcai@sjtu.edu.cn

***Abstract*—Knowledge-augmented question answering (KAQA) based on large language models is promising for industrial decision support, yet safety-critical applications impose zero tolerance for factual and logical errors, while large language models remain prone to hallucinations. RAG improves answers by using external context, but it only helps before generation and lacks a way to check the results afterward. This paper proposes a graph-based consistency verification framework for reliable KAQA in industrial environments. The framework introduces a closed-loop process consisting of three main stages. First, the system structures generated answers into knowledge subgraphs. Second, it verifies these answers by checking logic against a domain knowledge graph and finding evidence in technical documents. Third, when errors are found, the system uses structured feedback to trigger a regeneration loop. This process repeats until the answer is reliable, resulting in a final standardized output that includes a reliability status and verified references. Experiments on an aero-engine testing benchmark show that the proposed method achieves higher reliability than representative baseline approaches, including LightRAG and GraphRAG, improving reliability-related metrics by 27% to 66%.**

***Index Terms*—Question Answering Systems, Knowledge Graphs, Consistency Verification.**

I. Introduction

Critical industrial domains such as energy and aerospace rely heavily on accurate technical knowledge to support engineering analysis, operation, and maintenance. In these scenarios, knowledge-augmented question answering (KAQA) systems based on large language models enable engineers to directly query complex technical knowledge and obtain structured guidance, thereby improving information accessibility and decision-making efficiency [1]. By integrating large language models with domain knowledge sources, KAQA systems transform static technical manuals and maintenance records into interactive and queryable resources, facilitating fast access to technical rules, operational procedures, and engineering constraints.

Despite these advantages, deploying large language model-based systems in industrial environments remains challenging due to strict reliability requirements. Large language models are known to produce hallucinations—responses that appear plausible but are factually incorrect [2]. In safety-critical industries, even minor numerical deviations or procedural errors may lead to serious operational risks. Retrieval-Augmented Generation (RAG) alleviates this problem by grounding model outputs in external documents [3]; however, most RAG systems improve answers by using external context, but it only helps before generation and lacks a way to check the results afterward.

Recent research efforts mainly attempt to improve reliability by enhancing retrieval strategies or incorporating additional reference information. Some approaches verify answers through document similarity matching or domain-specific tuning [4]. However, such methods rely primarily on semantic similarity rather than explicit logical constraints, often leading to semantic drift [5], [6], making it difficult to detect subtle errors such as incorrect numerical values or violations of physical and procedural rules. Furthermore, most existing systems lack self-correction capabilities: once an error is generated, it is directly presented to the user without further verification.

To address these limitations, this paper proposes a reliable KAQA framework based on a consistency verification mechanism. The proposed system introduces a closed-

This work was supported by the Shanghai Sailing Program under Grant No.23YF1420600, the AECC Sichuan Gas Turbine Establishment Stable Support Project under Grant No.GJCZ2-0301-04, and the Key Research and Development Program of Shandong Province under Grant No.2023CXGC010112.

loop process that integrates answer generation, verification, and regeneration. Generated answers are first structured into knowledge subgraphs under domain ontology constraints, and then verified through dual-dimensional verification, which jointly checks logical consistency against a domain knowledge graph and traces semantic evidence back to original technical documents. When inconsistencies or unsupported claims are detected, targeted error feedback is used to trigger iterative answer regeneration until predefined reliability criteria are satisfied.

The main contributions of this paper are summarized as follows:

- We propose a closed-loop KAQA framework for safety-critical industrial applications that integrates answer generation, consistency verification, and feedback-driven regeneration to improve reliability.
- We design a dual-dimensional verification mechanism that decomposes generated answers into atomic claims and verifies them through knowledge graph-based logical consistency checking and document-level semantic evidence tracing.
- We evaluate the proposed framework on a specialized industrial benchmark, where it demonstrates consistent improvements over representative baseline methods, including LightRAG [7] and GraphRAG [8], achieving gains of 27% to 66% in key reliability-related metrics.

II. Related Work

Retrieval-Augmented Generation in Specialized Domains. Retrieval-Augmented Generation (RAG) is now a common method in professional fields. It finds outside documents to help Large Language Models and fixes gaps in their knowledge. Tools using vector search are good at finding similar meanings. However, they often miss exact technical terms used in critical industries. To fix this, recent studies add Knowledge Graphs. For example, QA-GNN uses both graphs and text to find clear relationships [9]. Zhang et al. put structured knowledge directly into model layers for better interaction between text and graph parts [10]. Other studies have used similar RAG methods to improve safety and compliance in industrial automation [11]. Newer tools like GraphRAG and LightRAG look at graph groups to understand the big picture. Even with these updates, most methods only try to find more information. They do not use graph structures to check if the logic is correct.

Hallucination Mitigation and Reliability. Reliability is very important for giving advice in industry. A big problem is that models still create wrong information during use. Most current plans only look for errors after the model finishes writing. Adding citations helps track sources but does not fix errors. Methods like Chain-of-Thought [12] and better prompts [13] make reasoning clearer but do not promise the facts are right. Some recent works look at checking and self-fixing. Self-RAG teaches models to judge their own search quality [14]. CRITIC lets models use outside tools to check their claims [15]. However, these methods usually work on simple text or internet data. They have trouble finding broken physical rules or wrong steps defined in industrial standards.

Logical Verification. A significant gap remains in creating systems that actively correct logical errors. Current strategies largely rely on Self-Correction [16], where the model reflects on its own output, or rigid Rule-Based filtering [17]. However, Self-Correction often suffers from confidence bias due to the lack of external references, while hard rules lack the flexibility to capture complex semantic dependencies. Consequently, there are no standard systems that leverage structured knowledge for iterative refinement. This study suggests a way to check consistency using graphs. It creates a full cycle of creating, checking, and re-writing. This makes sure the answers follow the rules of the industry.

III. Preliminaries and System Overview

A. Problem Formulation

The goal of industrial test guidance is to develop a reliable KAQA system that produces verifiable responses under domain constraints.

Formally, let $\mathcal{K}_{\text{domain}} = \{G_{\text{kb}}, D_{\text{vec}}\}$ denote the Domain Knowledge Base, where $G_{\text{kb}} = (E, R)$ is a structured knowledge graph and D_{vec} is a text vector database. Given an initial response A_{init} generated by a RAG module, the objective is to obtain a final response A_{final} such that all statements are logically consistent with G_{kb} and supported by evidence from D_{vec}. The system further outputs a reliability label $v \in \{\textit{Reliable}, \textit{Pending}\}$.

B. SYSTEM OVERVIEW

We propose a graph-based consistency verification framework for reliable question answering in industrial environments, as shown in Fig. 1. The framework verifies the preliminary response generated by a Large Language Model against a structured Domain Knowledge Base to ensure factual correctness and logical consistency.

The system takes a preliminary response A_{init} from a RAG module and refines it into a reliable response A_{final}. This process relies on a Domain Knowledge Base $\mathcal{K}_{\text{domain}}$, as shown in Fig. 2. The knowledge graph G_{kb} encodes domain-level logical constraints, while the text vector database D_{vec} provides semantic evidence support.

The verification workflow transforms the preliminary answer into a reliable result through three stages:

(1) Answer Subgraph Construction under Knowledge Alignment. The system converts A_{init} into a structured Answer Subgraph G_{ans} aligned with domain knowledge.

(2) Consistency Verification Based on Dual-Dimensional Analysis. The subgraph G_{ans} is evaluated against G_{kb} for logical consistency and against D_{vec} for semantic support.

(3) Reliable Answer Regeneration Based on Iterative Regeneration. If inconsistencies are detected, feedback is generated to guide the refinement of A_{init} until a reliable response A_{final} is obtained.

Overall, the framework establishes a closed-loop mechanism that bridges probabilistic generation and deterministic domain

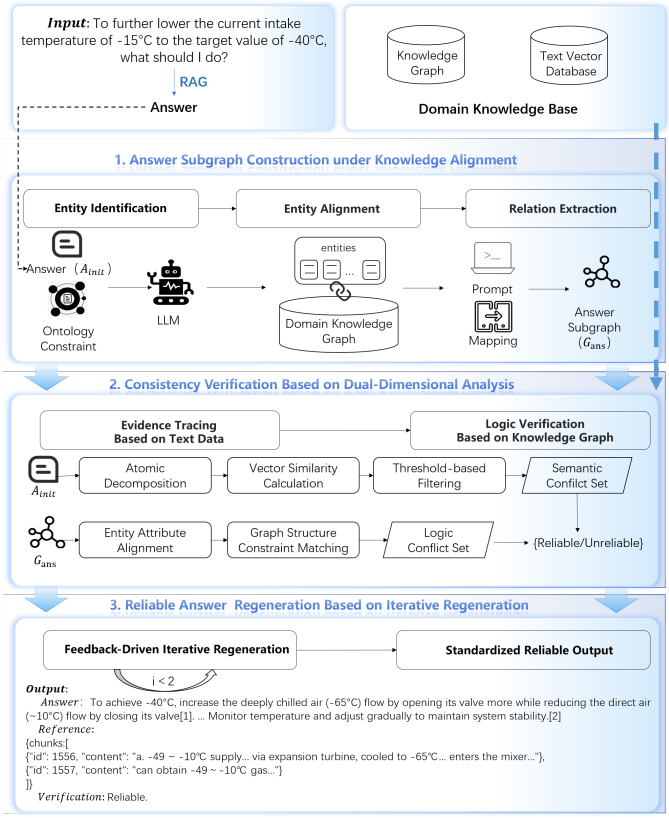

Fig. 1. Overview of the Graph-Based Consistency Verification Framework.

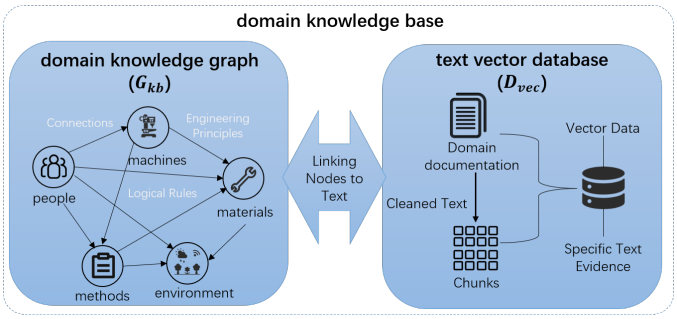

Fig. 2. The Structure of the Domain Knowledge Base.

constraints, ensuring that the final output is logically consistent and factually grounded.

IV. Methodology

This section details the proposed verification mechanism. As a post-processing module, it accepts the initial response A_{init} generated by the RAG system and processes it through three cascaded steps: (A) Answer Subgraph Construction under Knowledge Alignment,(B) Consistency Verification Based on Dual-Dimensional Analysis,(C) Reliable Answer Regeneration Based on Iterative Regeneration.

A. Answer Subgraph Construction under Knowledge Alignment

To perform logical checks on the unstructured first answer A_{init}, the system transforms it into a structured Answer Subgraph G_{ans}. This process uses the domain ontology to find entities and logic chains in the text and maps them to standard knowledge. There are three main steps:

1) Entity Identification: The system first performs entity extraction under Ontology Constraints. An LLM-based tool reads A_{init} to find professional terms. By enforcing the constraints of the five-dimensional ontology framework, the model filters out abstract descriptors, retaining only concrete engineering entities relevant to the verification task.

2) Entity Alignment: To resolve terminological diversity, we employ an Entity Alignment mechanism utilizing a domain-finetuned BERT model. This step computes vector representations for the identified mentions and aligns them with standard nodes in the Domain Knowledge Graph. By maximizing semantic similarity between mention embeddings and standard node embeddings, the system resolves ambiguity and ensures precise entity linking.

3) Relation Extraction: Following alignment, the system performs a two-stage process to recover the semantic dependencies between entities and build the logical backbone of the answer. In the first stage, the LLM identifies initial entity-relation pairs from the generated text, capturing both clear and hidden associations. In the second stage, the system performs ontology constraint verification. It matches these relations against 15 core types defined in the domain ontology, as illustrated in Fig. 3, such as Causal Influence, Control Operation, or Sequential steps. Any relation that violates these domain rules is discarded to ensure the results are standardized. The final set of confirmed entities and relations forms the Answer Subgraph G_{ans}, which serves as a formalized version of the answer for the subsequent consistency verification phase.

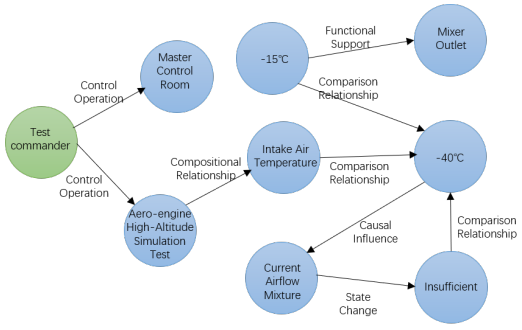

Fig. 3. An illustrative example of an Answer Subgraph (G_{ans}).

B. Consistency Verification Based on Dual-Dimensional Analysis

This phase serves as the core verification module, performing two parallel checks: semantic evidence tracing using D_{vec} and logical consistency verification using G_{kb}.

1) Evidence Tracing Based on Text Data: This step verifies whether generated answers are supported by technical documents. As shown in Algorithm 1, line 2 decomposes the unstructured answer A_{init} into atomic claims C.

Lines 4–9 perform semantic evidence tracing. For each claim c_j, the system searches D_{vec} and identifies the best evidence k_{best} by maximizing the similarity score Score(c_j, v_k).

Lines 10–12 handle continuity detection. Adjacent claims mapped to the same source are merged to avoid fragmented evidence.

Lines 13–16 determine reliability. If the maximum similarity score is below the threshold τ_{ver}, the claim is added to the semantic conflict set $\mathcal{C}_{semantic}$.

Finally, line 18 returns the evidence alignments and the conflict set $\mathcal{C}_{semantic}$.

Algorithm 1 Semantic Decomposition and Mapping

Require: A_{init}, D_{vec}
Ensure: $alignments$, $\mathcal{C}_{semantic}$
1: $alignments, \mathcal{C}_{semantic} \leftarrow \emptyset$
2: $C \leftarrow$ semantic_segmentation(A_{init})
3: **for all** $c_j \in C$ **do**
4: $\quad best\_score \leftarrow 0,\ k_{best} \leftarrow \emptyset$
5: $\quad$ **for all** $v_k \in D_{vec}$ **do**
6: $\quad\quad$ **if** Score$(c_j, v_k) > best_score$ **then**
7: $\quad\quad\quad best\_score \leftarrow$ Score$(c_j, v_k),\ k_{best} \leftarrow v_k$
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: $\quad$ **if** $j > 1$ **and** continuity_detect(c_j, c_{j-1}) **and** $k_{best} = alignments[j-1].k_{best}$ **then**
11: $\quad\quad$ merge_claims$(alignments[j-1], c_j)$; **continue**
12: $\quad$ **end if**
13: $\quad$ **if** $best\_score < \tau_{ver}$ **then**
14: $\quad\quad \mathcal{C}_{semantic} \leftarrow \mathcal{C}_{semantic} \cup \{c_j\}$
15: $\quad$ **end if**
16: $\quad alignments \leftarrow alignments \cup \{(c_j, k_{best}, best\_score)\}$
17: **end for**
18: **return** $alignments, \mathcal{C}_{semantic}$

2) Logic Verification Based on Knowledge Graph: At the same time, the system verifies the logical correctness of the answer by comparing the Answer Subgraph (G_{ans}) with the standard Domain Knowledge Graph (G_{kb}). As detailed in Algorithm 2, this process involves strict conflict detection at both the entity and relation levels, resulting in a logical conflict set $\mathcal{C}_{logic}$.

Next, lines 2 to 9 perform the entity-level check. For an entity e_{gen} in the answer, the system finds the matching standard node e_{kg} in the knowledge graph. It then calculates the similarity between their attribute vectors, $Sim_{attr}(e_{gen}, e_{kg})$. If this similarity is lower than the attribute threshold δ_{attr}, the system reports an Attribute Mismatch:

$$Sim_{attr}(e_{gen}, e_{kg}) = \sum_{i=1}^{d} w_i \cdot \mathrm{sim}\left(a_i^{gen}, a_i^{kg}\right), \qquad (1)$$

$$Sim_{attr}(e_{gen}, e_{kg}) < \delta_{attr} \ \Rightarrow\ Conflict_{attr} \in \mathcal{C}_{logic}, \quad (2)$$

where a_i^{gen} and a_i^{kg} denote the i-th attribute of the generated entity and the corresponding knowledge graph entity, respectively, and w_i represents the importance weight of each attribute defined by domain ontology constraints.

After checking entities, lines 10 to 14 check the relationships. The system also checks if the relationships $r_{gen}(e_1, e_2)$ exists in the standard Domain Knowledge Graph G_{kb}. If a relationship is not present in the Domain Knowledge Graph G_{kb}, the system adds a Relation Conflict to the list.

Finally, lines 15 to 19 decide the result. The system checks both the logical conflicts $\mathcal{C}_{logic}$ and the semantic conflicts $\mathcal{C}_{semantic}$ from the previous step. If both sets are empty, the result is marked as *Reliable*. Otherwise, it is marked as *Unreliable*.

Algorithm 2 Knowledge Graph-Based Reliability Analysis

Require: G_{ans}, G_{kb}, $\mathcal{C}_{semantic}$
Ensure: $reliability\_result$, $\mathcal{C}_{logic}$
1: $\mathcal{C}_{logic} \leftarrow \emptyset$
2: **for all** $e_{gen} \in G_{ans}.entities$ **do**
3: $\quad e_{kg} \leftarrow$ find_best_match(e_{gen}, G_{kb})
4: $\quad$ **if** $e_{kg} \neq \emptyset$ **and** Sim$_{attr}(e_{gen}, e_{kg}) < \delta_{attr}$ **then**
5: $\quad\quad \mathcal{C}_{logic} \leftarrow \mathcal{C}_{logic} \cup \{Conflict_{attr}\}$
6: $\quad$ **else if** $e_{kg} = \emptyset$ **and** detect_novel_entity_risk(e_{gen}) **then**
7: $\quad\quad \mathcal{C}_{logic} \leftarrow \mathcal{C}_{logic} \cup \{Conflict_{entity}\}$
8: $\quad$ **end if**
9: **end for**
10: **for all** $r_{gen} \in G_{ans}.relations$ **do**
11: $\quad$ **if** $r_{gen} \notin G_{kb}$ **then**
12: $\quad\quad \mathcal{C}_{logic} \leftarrow \mathcal{C}_{logic} \cup \{Conflict_{relation}\}$
13: $\quad$ **end if**
14: **end for**
15: **if** $\mathcal{C}_{logic} = \emptyset$ **and** $\mathcal{C}_{semantic} = \emptyset$ **then**
16: $\quad reliability\_result \leftarrow$ Reliable
17: **else**
18: $\quad reliability\_result \leftarrow$ Unreliable
19: **end if**
20: **return** $reliability\_result, \mathcal{C}_{logic}$

C. Reliable Answer Regeneration Based on Iterative Regeneration

This final phase acts as the corrective control loop of the system. Unlike standard RAG systems that provide the first answer immediately, our framework evaluates the verification results from the previous phase.

1) Feedback-Driven Iterative Regeneration: When the verification result is marked as *Unreliable*, the system initiates a feedback-driven iterative regeneration process. Rather than naively rerunning the language model, the proposed framework explicitly converts verification outcomes into structured feedback signals to guide targeted error correction.

The feedback information is formalized as a constraint set:

$$\mathcal{F} = \{\mathcal{C}_{logic}, \mathcal{C}_{semantic}\}, \tag{3}$$

Conditioned on $\mathcal{F}$, the system dynamically constructs a constrained regeneration input $A_{new} = \text{LLM}(q, A_{curr}, \mathcal{F})$ based on the structured template shown in Fig. 4.

The process terminates when the answer is labeled as *Reliable* or the maximum iteration limit N is reached. If conflicts remain unresolved after N iterations, a fail-safe mechanism is activated, and the answer state is set to *Pending*. In this case, the remaining conflict points are explicitly exposed to the user to prevent unsafe or misleading decision-making.

Instruction and Role Definition:
You act as a domain expert responsible for refining a preliminary answer based on structured verification feedback.
The objective is to correct detected inconsistencies while ensuring accuracy, traceability, and compliance with domain standards.

Context and Input:
- User Query: {q}
- Preliminary Answer: {A_{curr}}

Structured Verification Feedback(F):
- Knowledge Graph Conflicts (C_{logic}): Detected violations of knowledge graph constraints, including incorrect entities, attributes, or relations.
- Document Gaps($C_{semantic}$): Statements lacking sufficient supporting evidence from source documents.

Regeneration Constraints:
- Knowledge Grounding: All revised content must be derived strictly from the provided reference knowledge and verified evidence fragments.
- Traceability: Each corrected factual statement must be explicitly linked to its source identifier.
- Minimal Revision: Preserve all verified components of A_{curr} and modify only those elements associated with F

Target Output Structure:
[Automatically Generated Title Based on User Intent]
{Revised Answer Content}
Evidence Sources
1.Standards / Specifications: [List of IDs]
2.Technical Documents: [List of Document IDs]

Fig. 4. Structured Feedback Prompt Template

2) Standardized Reliable Output: When the process ends, the system creates a structured output to ensure the results are clear and easy to track. The final output is a tuple:

$$Verified\_Output = (A_{final}, Reference, Status)\,, \tag{4}$$

where the components are defined as follows:

- A_{final} is the final natural language response, which corresponds to A_{new} if the system successfully resolved conflicts, or A_{init} if the initial answer was verified as *Reliable*.
- $Reference$ is the set of verified claim–evidence alignments $\mathcal{R} = \{(c_j, k_{best})\}_{j=1}^{m}$ obtained through semantic tracing, proving the grounding of each claim in the text database D_{vec}.
- $Status \in \{\text{Reliable}, \text{Pending}\}$ represents the final reliability grade. An answer is marked as *Reliable* only if the aggregate conflict set $\mathcal{C}_{conflict} = \mathcal{C}_{logic} \cup \mathcal{C}_{semantic} = \emptyset$. If conflicts remain after N iterations, it is labeled as *Pending*.

This structured format focuses on exposing potential risks rather than generating superficially fluent but factually incorrect answers. This approach satisfies the rigorous standards and safety demands of the industrial domain.

V. Experiments and Evaluation

A. Benchmark Set Construction

To test the system, we selected the specific field of aero-engine testing as our experimental domain. This domain includes 17 entity types and 15 relationship types, as illustrated in Fig. 5, to provide clear rules for checking logic. We systematically evaluated the proposed method using a specialized Q&A benchmark derived from a 1.5-million-character corpus of textbooks and technical standards.

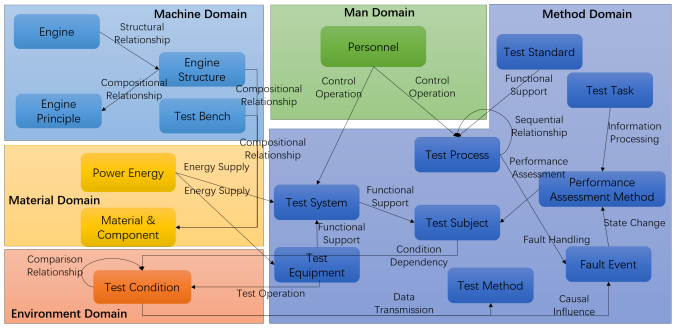

Fig. 5. The Ontology Construction Diagram of the Aero-Engine Testing Domain.

The dataset comprises 150 expert-verified pairs, distributed evenly with 30 per category across five cognitive levels:

- Factual: Precision in terminology, composition, and parameters.
- Explanatory: Elucidation of complex mechanisms and causal relationships.
- Procedural: Articulation of test workflows and operational specifications.
- Reasoning: Logical deduction and root cause analysis for fault scenarios.
- Open-ended: Macro-analysis of industry trends and technical challenges.

This benchmark encapsulates the domain's high specialization and strict safety demands, offering a robust standard for comprehensive system evaluation.

B. Experimental Setup

To comprehensively evaluate the proposed method, we designed controlled experiments for both accuracy and efficiency. Accuracy tests compare answer quality across five knowledge configurations:

- Base LLM (No Context): Uses Qwen2.5-14B-Instruct relying solely on internal parametric knowledge [18].
- Entity Augmentation: Incorporates document fragments via multi-path retrieval as context references.
- Vector Augmentation: Utilizes paraphrase-multilingual-MiniLM-L12-v2 for semantic matching and re-ranking [19].

- GraphRAG: Extends retrieval scope through knowledge graph paths for structural enhancement.
- LightRAG: Adopts a lightweight architecture focusing on efficient semantic integration.

All models are deployed locally on the ModelScope platform with Qwen2.5-14B-Instruct. Hyperparameters are fixed at a temperature of 0.1 and a maximum generation length of 512 tokens to ensure deterministic outputs. The semantic verification threshold τ_{ver} and attribute similarity threshold δ_{attr} are set to 0.85 and 0.90, respectively, based on empirical tuning. The maximum iteration number N is limited to 2 to control end-to-end latency. All experiments are conducted under a unified hardware environment, and efficiency is measured by end-to-end processing time.

C. Evaluation Metrics

We employ a dual-dimensional framework to assess system performance. Efficiency is measured by the end-to-end Average Generation Time. Accuracy is evaluated using a multi-level approach:

- Traditional Metrics: We utilize TF-IDF Similarity [20] and Semantic Similarity for vector-based alignment, alongside BLEU [21] and ROUGE-L [22] to assess n-gram overlap and lexical recall.
- LLM-Score: To capture semantic quality, we employ Qwen to approximate expert judgment. It evaluates the response against the ground truth based on accuracy, completeness, and professionalism, assigning a comprehensive score of 0–10.

D. Experimental Analysis

1) Accuracy Experiment Analysis: To comprehensively evaluate the system's generation performance, we first conducted a multi-dimensional automated evaluation of answer quality under six knowledge configuration conditions and incorporated a Large Language Model for automated scoring. A detailed comparison of the specific metrics is presented in Table I.

The proposed method demonstrates superior comprehensive performance, achieving the highest values across all automated metrics, including TF-IDF, Semantic Similarity, BLEU, and ROUGE scores. By leveraging knowledge graph relational constraints, it shows distinct advantages in terminological precision, outperforming the LightRAG baseline and the Base LLM with significant improvements of 27% and 66% in TF-IDF similarity, respectively. Additionally, it surpasses the Entity Augmentation baseline with a 79% increase in BLEU score. Furthermore, the method secured the highest LLM automated score of 8.41, confirming its exceptional capability in ensuring factual accuracy and reliability for aero-engine testing scenarios.

2) Accuracy Analysis Based on Fine-grained Question Types: To investigate the enhancement effects of knowledge strategies on questions of varying cognitive levels, we conducted a detailed comparison of the performance across four question categories under six knowledge configuration conditions, based on the constructed benchmark set. Key data are presented in Fig. 6.

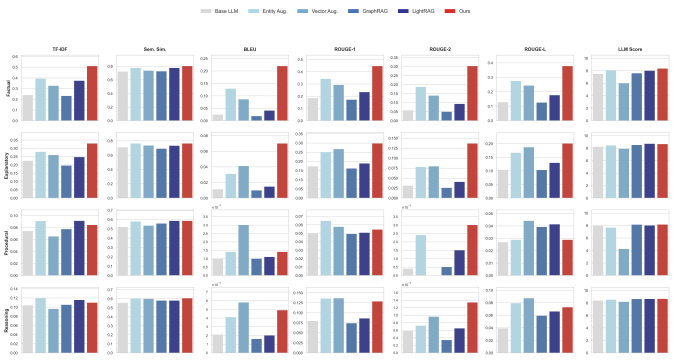

Fig. 6. Performance Comparison of Fine-grained Question Types (excluding open-ended question)

Knowledge enhancement strategies yielded positive gains across all categories, with the most significant optimization in Factual questions, where the proposed method improved TF-IDF by 113.5% and BLEU by over 800%, demonstrating its exceptional terminological precision. For Explanatory questions, it achieved the highest scores across TF-IDF, BLEU, and ROUGE metrics, demonstrating its effectiveness in handling complex questions.

In challenging Procedural and Reasoning tasks, while surface-level metrics remained competitive, our method demonstrated superior high-level semantic alignment. Specifically, it secured the highest ROUGE-2 and LLM Score for procedural steps, and the top Semantic Similarity and LLM Score for logical deduction, indicating a robust grasp of core logic over mere keyword matching. Additionally, for Open-ended questions, the method achieved a leading expert score of 8.44.

Overall, the results show that the proposed method is particularly effective for tasks requiring strict factual knowledge. It dominates in accuracy while also maintaining a clear lead in semantic consistency for complex logical reasoning.

3) Generation Efficiency Analysis: We evaluate generation efficiency by measuring average processing time across six configurations, as shown in Table I. The integration of external knowledge improves output quality but introduces additional latency. This overhead mainly arises from expanded input context, multi-stage retrieval and re-ranking, and graph-based reasoning.

When the initial response A_{init} is incorrect, the system triggers an iterative feedback loop, leading to increased processing time. The verification stage scans G_{ans} against G_{kb} and D_{vec} to identify inconsistencies and generate an error report. The regeneration stage then performs an additional inference pass to refine the response. Although this process increases computational cost, it ensures that errors are corrected through explicit validation.

Overall, the added latency reflects a trade-off for improved reliability. While the Base LLM achieves the highest ef-

TABLE I
OVERALL PERFORMANCE AND GENERATION EFFICIENCY COMPARISON UNDER DIFFERENT KNOWLEDGE CONFIGURATION CONDITIONS.

Configuration	TF-IDF	Sem. Sim.	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	LLM Score	Avg. Time (s)
Base LLM	0.1705	0.6385	0.0108	0.1292	0.0265	0.0805	8.06	7.18
Entity Aug.	0.2376	0.6927	0.0474	0.2119	0.0785	0.1494	8.22	27.17
Vector Aug.	0.2020	0.6613	0.0385	0.2007	0.0648	0.1510	6.82	8.10
GraphRAG	0.1611	0.6471	0.0086	0.1211	0.0223	0.0861	8.24	35.32
LightRAG	0.2222	0.6802	0.0166	0.1497	0.0400	0.1104	8.39	30.78
Ours	**0.2826**	**0.7020**	**0.0850**	**0.2519**	**0.1300**	**0.1878**	**8.41**	31.22

ficiency, the proposed method provides verified responses required for safety-critical tasks such as aero-engine testing and scheme evaluation. For such non-real-time applications, the increased latency is acceptable to ensure factual correctness and eliminate hallucinations.

4) Ablation Study: To rigorously quantify the contribution of the proposed post-processing module, we conducted a comparative ablation study. We evaluated our Graph-Based Verification method against two primary baselines. The first is the w/o Verification setting, which outputs the raw RAG generation A_{init} to represent the baseline capability. The second is a Self-Correction mechanism, where the LLM is prompted to review and refine A_{init} relying solely on its internal parametric knowledge.

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT VERIFICATION STRATEGIES.

Configuration	TF-IDF	Sem. Sim.	BLEU	ROUGE-L	LLM Score
w/o Verification	0.2510	0.6850	0.0521	0.1494	8.25
Self-Correction	0.2498	0.6845	0.0518	0.1488	8.24
Ours	**0.2826**	**0.7020**	**0.0850**	**0.1878**	**8.41**

As shown in Table II, the Self-Correction method did not improve performance. The LLM Score stayed at 8.24. Because the model relied only on its internal memory, it repeated its errors instead of fixing them. This proves that the model cannot verify complex facts using only its own knowledge.

In contrast, our Graph-Based method raised the LLM Score to 8.41. The rise in TF-IDF and BLEU scores highlights a key benefit: the use of precise terms. While baseline models often use vague words, our method uses graph rules to force the model to use the exact technical terms needed in the industry. This effectively prevents confusion and ensures the output maintains professional accuracy. Also, the better Semantic Similarity confirms that our structure makes the text more logical and detailed, qualities that are often lost in standard text generation. By catching factual errors that self-checks miss, our method connects uncertain model outputs with strict engineering rules, ensuring the answer is safe and reliable, meeting the high standards required by the industry.

E. case study

To demonstrate practical effectiveness, we conducted a qualitative analysis of a typical engineering query, as shown in Fig. 7.

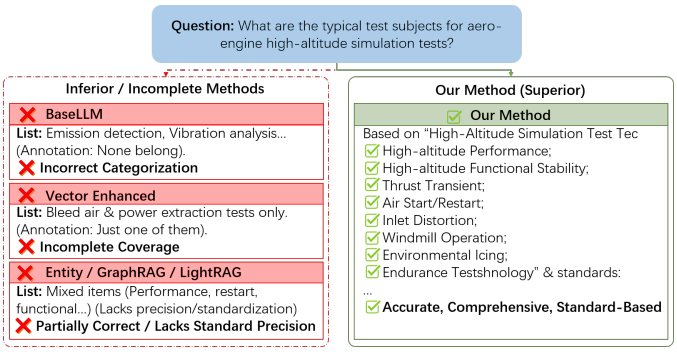

Fig. 7. Comparison of Response Quality Across Different Methods

Comparative results reveal distinct performance gaps. Baseline models, lacking context, frequently hallucinated generic items irrelevant to altitude simulation. Traditional RAG methods, constrained by retrieval granularity, often produced fragmented, unstructured answers.

In contrast, the proposed method, anchored by the domain Knowledge Graph, successfully generated a complete set of core test subjects—including "altitude performance," "thrust transients," and "inlet distortion"—strictly adhering to National Military Standards. By avoiding colloquialisms and ensuring structural completeness, the system transcends mere textual relevance to deliver reliable, standardized guidance suitable for actual test planning.

F. Discussion

The results indicate that enforcing deterministic constraints via knowledge graphs effectively reduces hallucinations in terminology-sensitive tasks, while variability in reasoning performance suggests the need for tighter integration of retrieval and logical synthesis. The observed latency reflects a trade-off inherent to safety-critical applications, where accuracy outweighs response speed. Although the current evaluation is based on 150 expert-verified Q&A pairs, future work will extend the benchmark to larger multi-scenario datasets and incorporate more direct reliability metrics, such as reductions in logical violations, alongside human-in-the-loop evaluation. Further efforts will focus on improving efficiency through parallelization and exploring adaptive thresholds to balance safety and performance.

VI. CONCLUSIONS

This paper presents a graph-based consistency verification framework for reliable question answering in safety-critical industrial environments. A closed-loop mechanism integrating generation, verification, and regeneration is established to mitigate hallucinations and enforce domain constraints. By structuring model outputs into knowledge subgraphs, the framework enables logical consistency checking against a knowledge graph and evidence validation from textual sources. Experimental results show that the proposed method outperforms representative baselines, including LightRAG and GraphRAG, in reliability-related metrics, while ablation studies further demonstrate improvements in technical accuracy and logical coherence. These findings highlight the effectiveness of integrating graph-based reasoning with generative models for industrial applications.

ACKNOWLEDGMENT

The authors would like to thank Shanghai Dingge Industrial Software Co., Ltd. for supporting this research.

REFERENCES

[1] T. B. Brown, B. Mann, N. Ryder, et al., "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[2] L. Huang, W. Yu, W. Ma, et al., "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions," *ACM Trans. Inf. Syst.*, vol. 43, no. 2, pp. 1–55, Jan. 2025.

[3] P. Lewis, E. Perez, A. Piktus, et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," *arXiv preprint arXiv:2005.11401*, 2021.

[4] T. Pang, K. Tan, Y. Yao, et al., "REMED: Retrieval-augmented medical document query responding with embedding fine-tuning," in *Proc. Int. Jt. Conf. Neural Netw. (IJCNN)*, Yokohama, Japan, 2024, pp. 1–8.

[5] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Hong Kong, China, 2019, pp. 3982–3992.

[6] M. Yuan, Q. Lu, X. Zeng, et al., "Alleviating semantic drift in multi-hop question answering on knowledge graphs with bidirectional semantics," *2024 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2024.

[7] Z. Guo, L. Xia, Y. Yu, T. Ao, and C. Huang, "LightRAG: Simple and fast retrieval-augmented generation," *arXiv preprint arXiv:2410.05779*, 2025.

[8] D. Edge, H. Trinh, N. Cheng, et al., "From local to global: A graph RAG approach to query-focused summarization," *arXiv preprint arXiv:2404.16130*, 2025.

[9] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, and J. Leskovec, "QA-GNN: Reasoning with language models and knowledge graphs for question answering," *arXiv preprint arXiv:2104.06378*, 2022.

[10] X. Zhang, A. Bosselut, M. Yasunaga, et al., "GreaseLM: Graph reasoning enhanced language models for question answering," *arXiv preprint arXiv:2201.08860*, 2022.

[11] M. L. Bernardi, M. Cimitile, and R. Pecori, "Automatic job safety report generation using RAG-based LLMs," in *Proc. Int. Jt. Conf. Neural Netw. (IJCNN)*, Yokohama, Japan, 2024, pp. 1–8.

[12] J. Wei, X. Wang, D. Schuurmans, et al., "Chain-of-thought prompting elicits reasoning in large language models," *arXiv preprint arXiv:2201.11903*, 2023.

[13] P. Liu, W. Yuan, J. Fu, et al., "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Comput. Surv.*, vol. 55, no. 1, pp. 1–35, 2023.

[14] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," *arXiv preprint arXiv:2310.11511*, 2023.

[15] Z. Gou, Z. Shao, Y. Gong, et al., "CRITIC: Large language models can self-correct with tool-interactive critiquing," *arXiv preprint arXiv:2305.11738*, 2024.

[16] A. Madaan, N. Tandon, P. Gupta, et al., "Self-Refine: Iterative refinement with self-feedback," *arXiv preprint arXiv:2303.17651*, 2023.

[17] N. Kassner, O. Tafjord, H. Schütze, and P. Clark, "BeliefBank: Adding memory to a pre-trained language model for a systematic notion of belief," *arXiv preprint arXiv:2109.14723*, 2021.

[18] A. Yang, B. Yang, B. Zhang, et al., "Qwen2.5 technical report," *arXiv preprint arXiv:2412.15115*, 2025.

[19] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manag.*, vol. 24, pp. 513–523, 1988.

[20] N. Reimers and I. Gurevych, "Making monolingual sentence embeddings multilingual using knowledge distillation," in *Proc. 2020 Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Online, 2020, pp. 4512–4525.

[21] K. Papineni, S. Roukos, T. Ward, and W. J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, Philadelphia, PA, USA, 2002, pp. 311–318.

[22] C. Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Proc. ACL-04 Workshop (Text Summarization Branches Out)*, Barcelona, Spain, 2004, pp. 74–81.