scientific reports

OPEN

An intelligent guided troubleshooting method for aircraft based on HybirdRAG

Xiaoyue Xie[1], Xilang Tang[1]✉, Siwei Gu[2] & Lijie Cui[1]

To enhance aircraft fault diagnosis efficiency, this paper proposes HybridRAG, an intelligent-guided troubleshooting framework that integrates knowledge graphs and large language models (LLMs). Unlike conventional retrieval-augmented generation (RAG) methods that rely on single-modal retrieval, HybridRAG adopts a multi-dimensional retrieval strategy, combining graph-based reasoning with both vector-based and BM25-based text retrieval techniques. This hybrid approach ensures comprehensive extraction of relevant information from both unstructured text and structured fault graphs, enhancing diagnostic precision, relevance, and robustness. Experimental results demonstrate that HybridRAG achieves an F1 score improvement of at least 4% and reduces hallucination rates by over 7% compared to mainstream RAG baselines. These advancements, combined with its unique integration of multi-modal retrieval, position HybridRAG as a novel framework for addressing complex aircraft maintenance challenges. Additionally, the paper presents an agent-based intelligent troubleshooting assistant that supports more interactive, adaptive, and flexible diagnostic Q&A, providing maintenance personnel with a significant advanced intelligent, context-aware diagnostic tool.

Keywords Knowledge graph, HybridRAG, Multi-Dimensional retrieval, Guided troubleshooting

As aircraft systems grow increasingly complex, fault diagnosis has become a typical knowledge-intensive activity. When facing unknown faults, maintenance personnel often spend considerable time analyzing technical documents pertaining to these anomalies. These documents provide essential details about system component composition, signal parameter variations, and historical fault case records. This time-consuming process leads to extended equipment downtime, significantly affecting aircraft availability and operational efficiency. Consequently, enhancing the efficiency and accuracy of fault diagnosis is crucial for ensuring aircraft stability and reliability.

To address these challenges, various intelligent diagnostic methods have been developed to assist maintenance personnel in more efficient fault identification and resolution. Early approaches primarily relied on rule-based system (RBS) and case-based reasoning (CBR). RBS achieves efficient diagnosis through predefined logic trees, proving particularly effective for routine fault scenarios[1]. However, its rigid knowledge representation imposes limitations when handling novel complex faults. In contrast, CBR leverages historical data similarity matching mechanisms, demonstrating stronger adaptability in unknown fault diagnosis[2], though its diagnostic accuracy remains highly dependent on the quality and comprehensiveness of the case database. To synergize the strengths of both approaches, hybrid expert systems emerged, integrating rule-based reasoning with case-based reasoning[3]. This combination improved diagnostic accuracy and adaptability, but it struggled to cope with the escalating complexity of aircraft systems and the ever-expanding of technical knowledge required.

The emergence of knowledge graph (KG) technology has driven extensive research into domain-specific expert system construction[4–7]. By establishing structured KGs paired with advanced reasoning algorithms, these systems help maintenance personnel reduce time spent on fault information retrieval, accelerate fault localization rapidly localizing faults, and enhance diagnostic efficiency during equipment operations. However, while knowledge graph-based systems streamline information retrieval and partially boost productivity, they ultimately remain constrained as static retrieval tools[8–10], lacking flexible natural multi-turn dialogue mechanisms and unable to provide step-by-step guidance for complex fault isolation. For instance, when diagnosing abnormal brake pressure, traditional KG approaches execute static entity resolution—mapping symptoms to related entities such as brake assemblies, hydraulic sensors, and control valves—followed by template-driven slot filling to retrieve relevant fault cases. This retrieval paradigm, relying on keyword matching or fixed vector similarity metrics, fails

[1]School of Equipment Management and UAV Engineering, Air Force Engineering University, Xi'an 710051, China. [2]Hangzhou Joyful Data Technology Corporation, Hangzhou 310018, China. ✉email: xxyueamss@163.com

to incorporate dynamic contextual cues or adapt reasoning paths through interactive dialogue. These limitations motivate the integration of large language models (LLMs), which offer flexible reasoning capabilities and natural multi-turn conversations for guided troubleshooting. Despite LLMs' well-documented success in general NLP tasks[11–14], their direct deployment for aircraft maintenance diagnostics faces significant barriers. Primary among these are inherent domain knowledge deficiencies — particularly regarding aircraft-specific fault patterns — that may compromise diagnostic reliability[15], coupled with the intrinsic opacity of neural network decision-making processes that undermines traceability in safety-critical scenarios[16]. In comparison, domain-specific KGs (e.g., AFKG[7], AAKG[17]) provide structured knowledge representation via graph databases with transparent reasoning processes. This complementary relationship—combing KG's structured expertise with LLM's adaptive reasoning—creates a promising framework for aircraft fault diagnosis, enabling maintenance personnel to leverage both technologies for substantial improvements in troubleshooting accuracy and operational efficiency.

Current methodologies for enhancing LLMs with KGs fall into two paradigms: knowledge infusion and retrieval-augmented generation (RAG). The knowledge infusion approach embeds KG elements directly into LLM pretraining to enhance domain knowledge retention, but faces critical constraints including the massive parameter sizes of modern LLMs requiring prohibitive computational resources for iterative updates, and the inherent data disparity between compact domain-specific KGs and LLMs' pretraining corpora that mandates meticulous prompt engineering to activate latent knowledge[18]. In contrast, RAG dynamically fetches relevant KG subgraphs during inference, supplementing LLMs with contextual knowledge[19], and demonstrates several advantages on real-time knowledge integration synchronization, cost-effective deployment, and auditable decision trails through explicit retrieval evidence[20,21]. These benefits make RAG particularly suited for aviation maintenance's dynamic environments. Given the cost-effectiveness and need for transparency in fault diagnosis, we prioritize RAG over KG embedding for our approach. Although a hybrid framework combining KG embedding and retrieval-based augmentation, as proposed in[22], appears promising for integrating structured knowledge into large models, it has notable drawbacks. Its effectiveness hinges on high-quality domain-specific KGs, which require continuous updates. Adapting the LLM to evolving knowledge still demands parameter adjustments, incurring significant computational costs. Moreover, unlike aviation assembly, our focus is on real-time fault diagnosis, where users need interactive, step-by-step guidance. This process relies on LLMs' general reasoning and conversational abilities, which can be hindered by subgraph embedding, limiting flexibility and adaptability in troubleshooting. Therefore, we aim to enhance fault diagnosis accuracy and streamline troubleshooting guidance by designing an advanced retrieval-augmented strategy while preserving the reasoning capabilities of the underlying large language model.

This study presents HybridRAG, an intelligent guided troubleshooting framework for aircraft maintenance that achieves an optimal balance between diagnostic accuracy with transparency. The framework establishes a domain-specific fault knowledge graph and implements a novel multi-dimensional retrieval strategy combining vector, BM25 text, and graph retrieval techniques. This hybrid approach comprehensively extracts and synthesizes relevant information from both unstructured text and structured graph data, enabling large language models to deliver more precise and context-aware fault diagnosis. Experimental results demonstrate that our hybrid retrieval strategy, outperforms several mainstream RAG methods in both diagnostic accuracy and interpretability. Furthermore, we develop an agent-based intelligent troubleshooting assistant within this framework, incorporating advanced interactive capabilities to provide adaptive, step-by-step fault localization guidance. Practical evaluations across multiple scenarios, especially in complex fault situations. In summary, this paper primarily contributes in three main areas.

- Addressing efficiency and accuracy issues in aircraft fault diagnosis: The paper addresses challenges encountered by traditional knowledge-based expert systems in complex diagnostic scenarios, such as rigid interaction methods and a lack of multi-round dialogue capabilities. It also considers potential accuracy and authenticity issues that may arise when applying large language models directly.
- Proposing a HybirdRAG-based intelligent guided troubleshooting framework: This framework merges the creation of domain-specific fault knowledge graphs with multi-dimensional retrieval strategies, including vector, BM25 text, and a novel graph exploration retrieval algorithm and highly optimized algorithm implementation.
- Significant advantages demonstrated in comparative experiments: Experimental results demonstrate that HybirdRAG achieves superior performance compared to mainstream RAG baselines in terms of fault diagnosis accuracy and hallucination. Moreover, particularly outstanding effectiveness in complex fault scenarios through our developed agent-based intelligent assistant, which provides adaptive, context-aware troubleshooting guidance.

Related work

RAG techniques have gained widespread adoption across multiple domains where output reliability is critical, including clinical decision support[23], financial risk assessment[24], and personalized education[25]. In aviation maintenance, RAG-based systems have demonstrated potential in accelerating fault diagnosis through the integration of historical case databases with technical documentation. However, their application in safety-critical aviation environments remains challenged by the need for more sophisticated retrieval architectures.

Conventional RAG implementations, such as VectorRAG and BM25-based RAG, primarily employ dense vector similarity search or keyword matching for knowledge retrieval. These methods excel in terms of efficiency and semantic/keyword alignment capabilities. For instance, VectorRAG can identify semantically similar historical cases (e.g., "engine temperature anomaly") within seconds, while BM25 effectively retrieves precise sections from maintenance manuals based on specific fault codes (e.g., "ECAM code 501"). However, their principal limitation lies in the inability to explicitly represent causal or structural dependencies within knowledge bases[26], often yielding disconnected troubleshooting suggestions. For instance, while VectorRAG might concurrently retrieve fragments related to "fuel pump failure" and "ECAM warning code 501", it cannot generate a causal path like "fuel pump failure → insufficient fuel pressure→ECAM 501 trigger". Similarly, BM25, being heavily reliant on keyword matching, struggles to produce engineering-logical troubleshooting workflows (e.g., "check wear-prone components first, then investigate system faults"). Although recent enhancements incorporate metadata filtering or hybrid search algorithms[27,28], these still underperform with heterogeneous aviation corpora, frequently producing inaccurate or incomplete diagnosis analyses.

GraphRAG has consequently emerged as a promising alternative, leveraging structured knowledge graphs to model fault causality for aviation maintenance. The current state-of-the-art in graph retrieval predominantly adopts a subgraph extraction approach, which entity linking first identifies query-relevant nodes followed by local subgraph expansion[29,30]. Although effective for capturing localized knowledge, this paradigm frequently fails to preserve long-range dependencies[31–33]. This limitation has spurred reasoning-enhanced graph methods like ROG[34], TOG[35] and COE[36], which combine LLMs with KGs to perform multi-hop inference and path planning—capabilities particularly valuable for tracing indirect fault relationships.

Nevertheless, the efficacy of graph-based retrieval remains fundamentally constrained by the knowledge density within the graph structure. Sparse knowledge representations, exemplified by the fault graph in[7], where "test method" entities contain only abstract attributes ("operations" or "judgments") without operational specifics (e.g., detailed workflows or procedural examples), often necessitate recourse to original unstructured text data. This underscores the critical need to augment graph retrieval with document retrieval for comprehensive, step-by-step troubleshooting support. Moreover, certain knowledge sources, such as conversational data like operator-expert troubleshooting dialogues, are not well-suited for direct knowledge graph extraction. The resource-intensive nature of KG construction further compounds these limitations, rendering exhaustive knowledge extraction from all documents impractical. Notably, since knowledge updates predominantly occur in textual formats, direct retrieval from unstructured texts frequently proves more efficient than perpetual graph reconstruction. These considerations collectively demonstrate the complementary advantages of integrating knowledge graph and document retrieval for fault diagnosis applications. Consequently, our work proposes a novel multi-dimensional retrieval paradigm within the RAG framework that synergistically combines vector-based, BM25, and graph-based approaches. This integrated strategy is specifically designed to deliver more robust and efficient troubleshooting support by capitalizing on the respective strengths of each retrieval modality.

Method

As illustrated in Fig. 1., the HybridRAG framework for aircraft guided troubleshooting comprises three core stages: domain knowledge graph construction, multi-dimensional retrieval (graph retrieval, vector retrieval, and BM25 retrieval), and answer generation. The entire process involves that it conducts multi-dimensional retrieval based on the query of user and place the Top-K retrieval results into prompting instructions of the large language model (LLM), which then generates an answer based on this foundation and returns it to the user.

A. Construction of Equipment Fault Graph

1) Graph scheme development.

To construct an effective fault knowledge graph, we systematically collected and organized aircraft technical documentation from manufacturers, maintenance facilities, regulatory agencies, and academic institutions. These materials encompass equipment configurations, operational principles, historical fault cases, and maintenance protocols, forming a comprehensive knowledge base that ensures both practical utility and broad applicability. Following the methodology outlined in[7], we develop an aircraft fault ontology model comprising 5 entity types, 8 relationship categories, and 18 attribute classes across structural, functional, and fault dimensions (Fig. 2). Detailed specifications of entities, relationships, and attributes are provided in Appendix A.

2) Entity-Relation Triple Extraction

Entity-relation triple extraction identifies entity-relation-entity structures (e.g., < Fault, Location, Unit>) from unstructured text, serving as the critical process in knowledge graph construction. While deep neural networks excel at this task through advanced semantic encoding, conventional two-stage pipelines[37–40]—first detecting entity pairs then classifying relationships as discrete labels—demonstrate critical limitations in handling

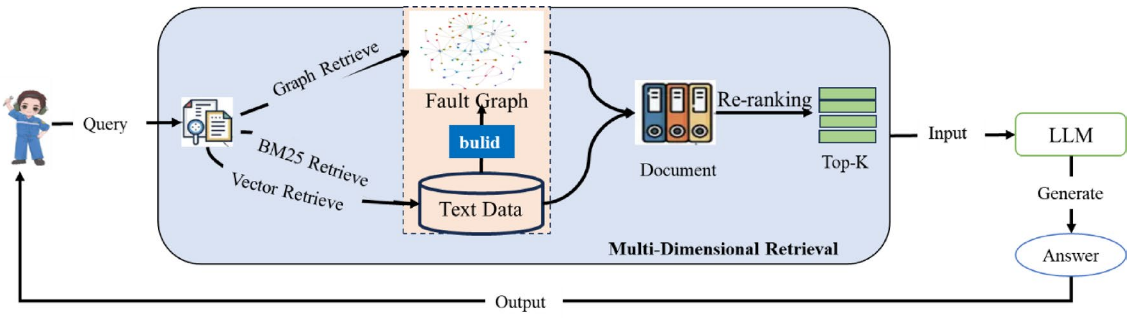


Fig. 1. HybirdRAG-based Troubleshooting Framework for Aircraft.

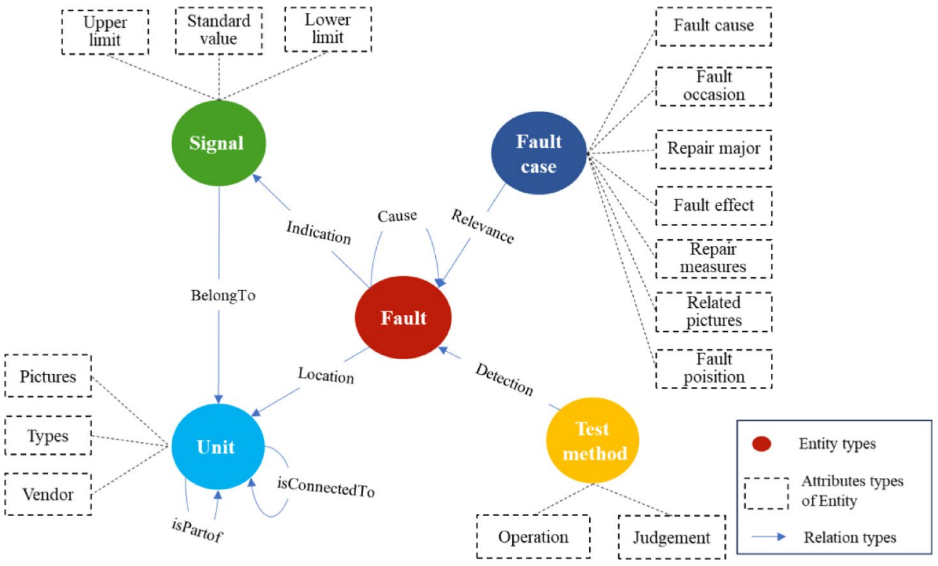


Fig. 2. The schema of fault knowledge graph. This schema consists of three types—entities, relationships, and entity attributes—where blue circles represent entity types, orange squares represent entity attributes, and blue arrows represent the relationship types between entities.

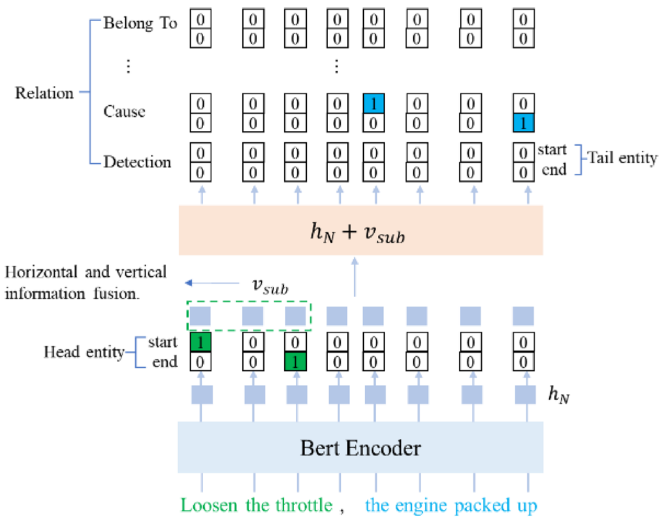


Fig. 3. The Hor-Ver-Casrel model architecture. The diagram illustrates the process of extracting a set of < Fault, Cause, Fault > triple instances, where the head entity is "loose the throttle", the tail entity is "the engine packed up".

overlapping triples prevalent in aviation maintenance data. These limitations become evident in statements like "The brake system consists of a brake solenoid valve, a brake controller, and a brake device", which generate multiple triples sharing identical predicate-subject pairs: < brake solenoid valve, isPartOf, brake system>, < brake controller, isPartOf, brake system>, and <brake device, isPartOf, brake system>. The Cascade Binary Tagging Framework (Casrel)[41,42] partially addresses this through relationship-as-function modeling but suffers from oversimplified head entity representation, using mere the start and end vectors averages that discard crucial semantic information.

To overcome these limitations, we propose the Hor-Ver-CasRel model—an enhanced Casrel framework integrating horizontal and vertical contextual information for enriched head entity representation. Horizontal context captures intra-sentence entity relationships through syntactic dependencies, while vertical context models inter-sentence semantic patterns across documents. The synergistic fusion of these dimensions elevates head entity representation quality, as visualized in Fig. 3. The algorithm is outlined as follows.

a. Input the text sentences into a pre-trained BERT model, which employs the Transformer architectures as its core encoding mechanism. The model adjusts the weight coefficient matrices based on the inter-character association degree within a sentence to extract word features, that is

$$Attention\left(Q,K,V\right)=Softmax\left(\frac{QK^{T}}{\sqrt{dk}}\right)V, \tag{1}$$

whereQ,K,Vrepresents the character vector matrix, dk represents the embedding dimension, $Attention$ denotes the self-attention mechanism function,$Softmax$indicates the activation function, and T represents the transpose symbol of the matrix.

b. Based on the vector representation of each character, predict the starting and ending positions of the head entity that

$$p_{i}^{start\_s}=\sigma\left(W_{start}F_{i}+b_{start}\right) \tag{2}$$

$$p_{i}^{end\_s}=\sigma\left(W_{end}F_{i}+b_{end}\right) \tag{3}$$

where F_{i} represents the vector of the i -th character, $p_{i}^{start\_s}$ and $p_{i}^{end\_s}$ denote the probability of the i-th character being the start position of the head entity, respectively. The weight matrices W_{start} and W_{end}, along with their corresponding biases b_{start} and b_{end}, are employed to predict the head entity's start and end positions, while σ represents the fully connected layer responsible for integrating these predictions into the final output.

c. Based on the head entity predicted in the previous step, let F_{a}, F_{b} be the word vector at the start and end position of the head entity. Through information fusion in both horizontal and vertical directions, obtain the vector representations for horizontal v_{sub}^{hor} and vertical v_{sub}^{ver} directions, respectively. That is

$$v_{sub}^{hor}=average\left(F_{a},F_{b}\right) \tag{4}$$

$$v_{sub}^{ver}=\sigma(W_{sub}[sum\left(F_{a}\right),sum\left(F_{b}\right)]+b_{sub}) \tag{5}$$

Further, concatenate these two vector to obtain the final information representation of the head entity, v_{sub}. That is

$$v_{sub}=v_{sub}^{hor}+v_{sub}^{ver} \tag{6}$$

d. Construct a mapping function for each relation using the vector representation of each character and the predicted head entity for predicting the tail entity, that is

$$p_{i}^{start\_o}=\sigma\left(W_{start}^{r}\left(F_{i}+v_{sub}\right)+b_{start}^{r}\right) \tag{7}$$

$$p_{i}^{end\_o}=\sigma\left(W_{end}^{r}\left(F_{i}+v_{sub}\right)+b_{end}^{r}\right) \tag{8}$$

where $p_{i}^{start\_o}$ and $p_{i}^{end\_o}$ denote the start and end position labels of the predicted tail entity for the i-th character, The weight matrices W_{start}^{r} and W_{end}^{r}, paired with biases $b_{start}^{r}$and b_{end}^{r}, predict these positions under the relation mapping.

The Hor-Ver-Casrel model's encoding process is completed via maximum likelihood estimation. The likelihood function combines two components that.

$$L\left(\theta\ ,\phi\ r\right)=logP_{\theta}\ \left(s|X\right)+logP_{\phi\ r}\left(o|s,X\right) \tag{9}$$

where $P_{\theta}\ \left(s|X\right)$ denotes the likelihood function of head entity prediction, $P_{\phi\ r}\left(o|s,X\right)$ denotes the likelihood function of relation and tail entity prediction. That is

$$P_{\theta}\ \left(s|X\right)=\prod\nolimits_{t\in\left\{start\_s,end\_s\right\}}\sum\nolimits_{i=1}^{L}\left(p_{i}^{t}\right)^{I\left\{y_{i}^{t}=1\right\}}\left(1-p_{i}^{t}\right)^{I\left\{y_{i}^{t}=0\right\}}$$

$$P_{\phi\ r}\left(o|s,X\right)=\prod\nolimits_{t\in\left\{start\_o,end\_o\right\}}\sum\nolimits_{i=1}^{L}\left(p_{i}^{t}\right)^{I\left\{y_{i}^{t}=1\right\}}\left(1-p_{i}^{t}\right)^{I\left\{y_{i}^{t}=0\right\}}$$

$$\theta=\left\{W_{start},b_{start},W_{end},b_{end}\right\},\varphi r=\left\{W_{start}^{r},b_{start}^{r},W_{end}^{r},b_{end}^{r},W_{sub},b_{sub}\right\}$$

Here, $\{y_{i}^{t}=1\}$ indicates that the i-th character is the start/end position of the head/tail entity, *L* is the sentence length, *I* is the indicator function.

Using fault knowledge extraction technology, we construct a fault knowledge graph for a certain type of aircraft. Figure 4. shows the local fault knowledge graph of a certain type of UAV system, which captures critical information such as system architecture, signal parameters, associated faults, and fault detection methods. The constructed graph provides a foundation for subsequent research on LLM-based reasoning over knowledge graphs.

B. HybirdRAG method based on Multi-Dimensional retrieval

The retrieval stage represents a critical component in RAG-based intelligent troubleshooting systems, where its primary objective is to deliver accurate responses to user queries by effectively extracting relevant information from a well-constructed knowledge base. As retrieval performance directly determines the quality

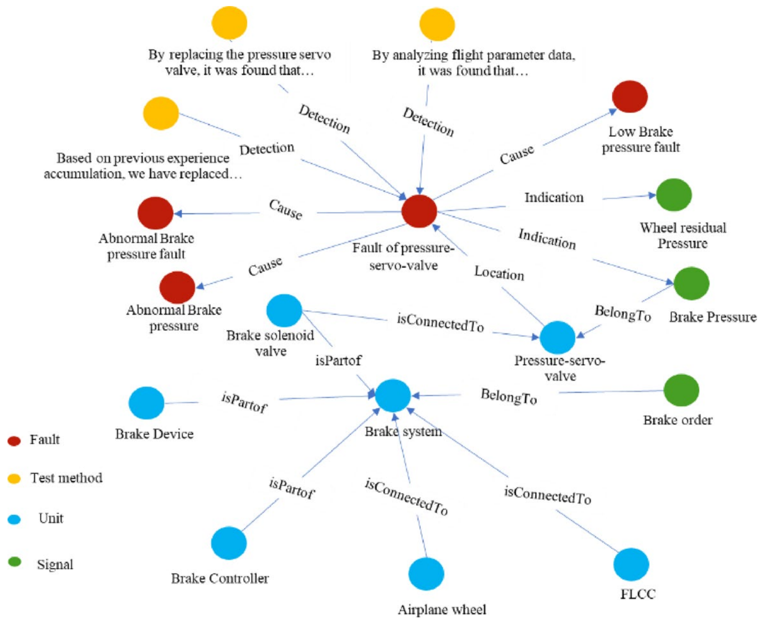


Fig. 4. The local fault knowledge graph of a certain type of UAV system, Circles of different colors represent different types of entities, and edges represent different types of relationships, where orange circles denote Test method, red circles denote Fault, blue circles denote Unit, and green circles denote Signal.

```
Require:
Query $q$ : A natural language query or knowledge retrieval task.
Graph $G = (V, E)$ : Knowledge graph with nodes $V$ and edges $E$ .
Tools T ={explore neighbors, explore paths, explore patterns}: Set of atomic graph exploration tools.
MAX ITERATIONS: Maximum number of exploration iterations
Ensure:
1: Response $R$ : A generated response or retrieved knowledge for $q$ .
2: iteration count $\leftarrow$ 0
3: while query q is not solved and iteration count < MAX ITERATIONS do
4:     iteration count $\leftarrow$ iteration count +1
5:     Plan: The planner-agent evaluates current state and either.
           •Designs next batch of exploration tasks B (to leverage tools from T ), or
           •Decides to generate final response if sufficient information exists
6:     if planner decides to explore then
7:         Execute tasks in the batch of B :
               $R_B \leftarrow \{t(p_1, p_2, \ldots) | t \in B\}$
           where $t(p_1, p_2, \ldots)$ stands for planned task, $R_B$ stands for results of current batch of exploration task.
8:         Self-Reflection: Evaluate exploration results and either:
           •Continue with next planning cycle, or
           •Add additional exploration tasks to current cycle if needed
9:         if additional exploration needed and iteration count <MAX ITERATIONS then
10:            Design additional tasks $B'$
12:        end if
13:    else
14:        Generate the response $R$ based on accumulated knowledge
16:        return $R$
17:    end if
18: end while
19: Generate the best possible response $R$ based on current knowledge
20: return Final response $R$ .
```

Algorithm 1 Chain of Exploration with Planner-Agent

```
Require:
    Node $v$: Target node for neighbor exploration.
    Depth $d$ : Maximum depth for subgraph retrieval.
    Graph $G = (V, E)$ : Knowledge graph.
Ensure:
    Subgraph $G' = (V', E')$ : Semantically ranked subgraph centered at $v$.
1: Initialize $V' = \{v\}$ and $E' = \varnothing$.
2: for $i = 1$ to d do
3:     Retrieve neighbors $N(v)$ for all nodes in $V'$.
4:     Perform vector search over $N(v)$ using cosine similarity: $V' \leftarrow V' \cup \{u | \cos\_\mathrm{sim}(\mathbf{e}_v, \mathbf{e}_u) \geq \tau\}$
5:     Add corresponding edges $E'$ from $G$.
6: end for
7: return Subgraph $G' = (V', E')$.
```

Algorithm 2 Explore neighbors(node, depth)

```
Require:
    Start node $v_s$, End node $v_e$ : Nodes to explore paths between.
    Graph $G = (V, E)$ : Knowledge graph.
Ensure:
    P :List of ranked paths.
1: Extract embeddings $\mathbf{e}_{v_s}$ and $\mathbf{e}_{v_e}$ for $v_s$ and $v_e$.
2: Perform vector-guided path search:
       $\mathrm{P} \leftarrow \{\mathrm{path} | \mathrm{path\ connects\ } v_s \mathrm{\ to\ } v_e \mathrm{\ and\ } \cos\_\mathrm{sim}(\mathbf{e}_{v_s}, \mathbf{e}_{v_e}) \geq \tau\}$.
3: return Ranked path list P .
```

Algorithm 3 Explore paths(start, end)

```
Require:
    Pattern $P$ : Graph pattern to match.
    Condition $C$ : Conditions on pattern matching.
    Semantic query $Q$ : Additional semantic constraints.
    Graph $G = (V, E)$ : Knowledge graph.
Ensure:
    M : List of matched subgraphs.
1: Translate $P$ and $Q$ into vector embeddings.
2: Execute pattern match under $C$ :
    $\mathrm{M} \leftarrow \{S | S \mathrm{\ satisfies\ } P, C \mathrm{\ and\ is\ semantically\ similar\ to\ } Q\}$.
3: Rank matches M by semantic relevance.
4: return Ranked subgraph list M .
```

Algorithm 4 Explore patterns(pattern, condition, query)

of RAG-generated outputs, our methodology employs a novel multi-dimensional approach that synergistically combines three complementary retrieval techniques. Our framework establishes knowledge graph retrieval as the foundational element, leveraging its rich semantic relationships to capture deep fault characteristics. This core retrieval layer is augmented by vector retrieval and BM25 algorithms, which enable efficient extraction of supplementary information from unstructured data sources. To further optimize retrieval quality, we implement a sophisticated re-ranking mechanism that evaluates and refines initially retrieved contexts, ensuring optimal relevance and precision in the final output. This integrated multi-retrieval strategy can screen out the most decisive information from large-scale knowledge repositories, significantly enhancing the efficiency and reliability of fault troubleshooting.

1) Primary retrieval based on KG exploration chain

KG retrieval exploits graph structures to identify the most relevant paths for query resolution. Formally, given a query q and domain-specific KG G, the retrieval task aims to find the optimal path w_z that maximizes the probability of answering q:

$$\mathrm{IR} = \arg\max_{w_z} P(w_z|q, G) \tag{10}$$

Inspired by the graph-based LLM reasoning work[34–36], we propose a Chain of Exploration with Planner-Agent for fault knowledge graph. Our strategy leverages a planner-agent framework to orchestrate exploration tasks, dynamically adjusting its exploration task based on intermediate results and ensuring context-aware, efficient traversal of the graph, as detailed in Algorithm 1. The algorithm is structured into three primary phases: planning, execution, and self-reflection. In the planning phase—the core component of the algorithm—two key tasks are performed. First, the system evaluates the current state of the query and designs a sequence of exploration tasks $\mathcal{B}$ that are tailored to the structure and content of the knowledge graph. These tasks are organized by a planner-agent utilizing the graph exploration tools $\mathcal{T}$ (namely, explore neighbors, explore paths, and explore patterns as detailed in Algorithms 2–4). Second, the planner evaluates the current retrieval state to decide whether to refine the exploration strategy, expand the search scope, or finalize the response via the LLM. Once the planner decides to explore further, the algorithm transitions to the execution phase, where the planned tasks are carried out. Following execution, the system enters the self-reflection phase in which the LLM assesses alignment between retrieved information and the initial query . If the current traversal sufficiently addresses the query requirements, the system proceeds with the next planning cycle; otherwise, it dynamically adjusts the exploration strategy—either by refining existing tasks or by generating additional tasks to address identified knowledge gaps. This iterative process continues until a conclusive answer is reached or the maximum iteration limit is met.

In contrast to conventional GraphRAG approaches that typically operate on schemeless triple graphs[33], our Chain of Exploration (CoE) framework leverages a structured schema and rich relational semantics within knowledge graph , offering superior performance for complex aircraft fault diagnosis. The innovation of CoE centers on the Zero-Shot Planning paradigm, which utilizes a domain-specific language (DSL) for sophisticated graph reasoning (see Appendix B for DSL details). In this paradigm, we define succinct commands—such as exploring neighbors, paths, or patterns—that can be seamlessly chained together by a large language model. These commands are interpreted by an execution mechanism that translates the DSL into precise graph exploration operations with minimal overhead. This approach not only differentiates CoE from standard GraphRAG solutions but also optimizes its performance for advanced reasoning tasks, making it ideal for handling intricate, real-world challenges. By integrating the Zero-Shot Planning paradigm with DSL, our method eliminates the need for task-specific fine-tuning, enabling rapid adaptation to diverse diagnostic problems while maintaining high efficiency and interpretability.

2) Complementary retrieval based on vectors and BM25

To fully leverage the original fault descriptions in aircraft technical documentation, we propose a complementary retrieval method based on vector similarity and BM25 algorithms to augment the knowledge obtained through graph retrieval.

a. Vector retrieval

Vector retrieval employs semantic similarity matching, with its workflow depicted in Fig. 5. This method initially processes fault text data through segmentation and chunking, followed by vector representation using advanced embedding techniques[43]. The system then constructs an efficient search index utilizing Hierarchical Navigable Small World (HNSW) graphs[44], enabling high-performance approximate nearest neighbor search. When processing user queries, the same embedding model generates vector representations that maintain semantic alignment with the indexed content. The HNSW algorithm subsequently retrieves the Top-K most semantically similar text passages, combining the nuanced semantic understanding of embeddings with the computational

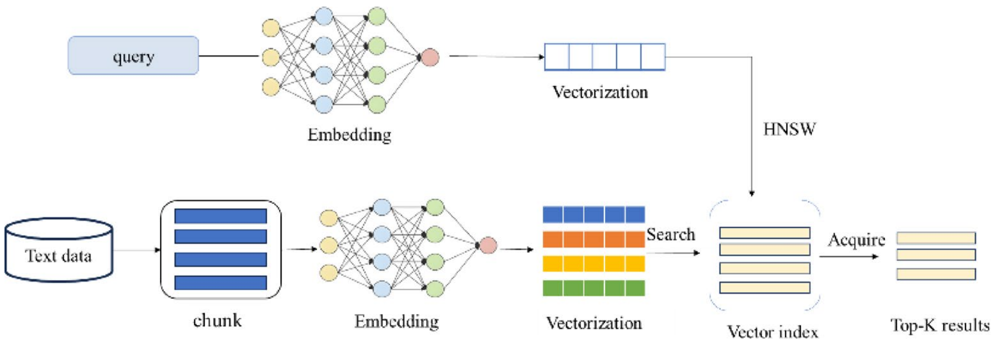


Fig. 5. Vector Retrieval Framework Diagram. The framework converts fault text data into vector representations via embedding techniques to construct an index, then retrieves the Top-K most similar results from the vector index for user queries using the HNSW algorithm.

efficiency of modern approximate nearest neighbor search techniques. This dual advantage makes the approach particularly effective for handling large-scale fault text corpora while ensuring both retrieval speed and result relevance.

b. BM25 retrieval
The BM25 algorithm establishes itself as a probabilistic retrieval framework that substantially improves upon traditional TF-IDF methods[45]. At its core, this algorithm evaluates document relevance through a sophisticated interplay of three critical factors including the frequency of query terms within documents, their inverse document frequency across the entire collection, and the length of the document. Within RAG systems, BM25 has proven particularly valuable for enhancing recall performance when employed alongside vector retrieval techniques[46]. For a query q composed of terms $q_1, q_2, \ldots, q_N$ and a document d, the BM25 ranking function is as follows:

$$Score(d, q) = \sum_{i=1}^{N} \mathrm{IDF}(q_i) \cdot \frac{f(q_i, d) \cdot (k_1 + 1)}{f(q_i, d) + k_1 1(1 - b + b\frac{|d|}{avgdl})} \tag{11}$$

where $Score(d, q)$ denotes the relevance score between the document d and the query q, $f(q_i, d)$ is the frequency of the term q_i in the document d, $|d|$ is the document length, and $avgdl$ represents the average document length across the corpus. The inverse document frequency $\mathrm{IDF}(q_i)$ is computed as $\log((N - n(q_i) + 0.5)/(n(q_i) + 0.5) + 1)$, where N is the total number of documents in the collection and $n(q_i)$ is the number of documents containing q_i. The free parameters k_1(typically set between 1.2 and 2.0) and b (usually 0.75) control term frequency saturation and document length normalization, respectively.

3) Cross-Encoding Re-Ranking

To enhance the relevance and accuracy of retrieved results, we introduce a cross-encoding re-ranking[47] mechanism that operates on the initial retrieval outputs from heterogeneous sources including knowledge graphs, vector embeddings, and BM25 algorithms. This module serves as a critical refinement stage in our Retrieval-Augmented Generation (RAG) pipeline, effectively filtering irrelevant content and optimizing the ranking of retrieved passages based on their semantic relevance to the input query[48,49].

The architecture of the cross-encoder, depicted in Fig. 6, implements a sophisticated relevance scoring system through sequential processing stages. Formally, given a query q and document chunk d, the model constructs an input sequence following the standard Transformer format:

$$Input = [CLS]q[SEP]d[SEP] \tag{12}$$

where [CLS] denotes the classification token and [SEP] represents sequence separators. This composite sequence undergoes deep contextual encoding through multiple Transformer layers:

$$H = Transformer\,(Input) \tag{13}$$

yielding a sequence of hidden states H. The model subsequently extracts the hidden state corresponding to the [CLS] token:

$$h_{\mathrm{CLS}} = H[\mathrm{CLS}] \tag{14}$$

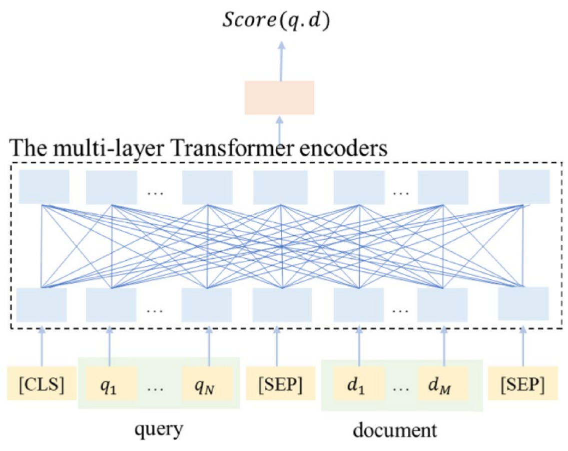


Fig. 6. Cross-encoding Re-ranking Model Architecture. The framework processes concatenated query-document sequences (yellow boxes) through multi-layer Transformer encoders (dashed box). The encoded sequence representations (blue box) are used to extract the [CLS] token's hidden state (orange box), which serves as the joint query-document representation for relevance scoring.

which encapsulates the joint query-document representation. Finally, a feedforward neural network (FFN) processes this representation to generate the relevance score:

$$Score\,(q, d) = FFN(h_{CLS}) \tag{15}$$

Based on these computed scores, the system returns the top-K most relevant contents for subsequent answer generation.

C. Answer generation

The LLM synthesizes coherent and contextually relevant responses by processing the top-K re-ranked retrieval results from previous stages. This input concentration mechanism ensures the model receives the most pertinent information for each query. Through carefully designed prompt engineering, the retrieved content is transformed into structured input parameters that enhance the LLM's contextual comprehension. The model then leverages its language generation capabilities to produce informative, query-specific responses. This generation process is formally expressed as

$$a = \mathrm{LM}(q, Q) \tag{16}$$

where Q represents the retrieved information. To optimize response quality, the LLM maximizes the probability of generating the most appropriate answer a given the query q and the retrieved context Q, that is

$$\arg\max_{a \in A} P(a|q, Q) = \prod_{j=1}^{m} P(a_j|a_{<j}, q, Q) \tag{17}$$

where A denotes the set of all possible answers, m is the answer length, and $P(a_j|a_{<j}, q, Q)$ represents the conditional probability of each word a_j in output sequence given the preceding context $a_{<j}$, query q, and retrieved information Q.

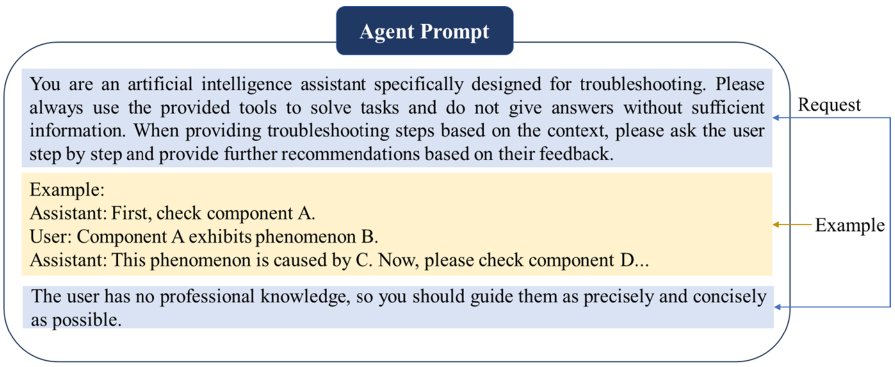
To operationalize interactive fault diagnosis, we developed an intelligent troubleshooting assistant within our guided framework. As illustrated in Fig. 7, the assistant implements structured system instruction prompts consisting of explicit request specifications for AI inference and formatted demonstration examples. These prompts enforce a strict step-by-step diagnostic protocol where the assistant must first request user verification of a specified component, then provide subsequent guidance based on the observed phenomenon, and finally proceed to the next logical check point. The design specifically accommodates non-expert users through constrained interaction patterns that prevent unsupervised diagnostic actions.

Experiment studies

This section evaluates the troubleshooting guidance efficiency of the proposed method in aircraft fault diagnosis scenarios, with experimental dataset derived from the UAV fault knowledge graphs constructed in Method section. The implementation details and results analysis are presented in the following subsections.

A. Experimental settings

The experiments were conducted on a local machine equipped with four NVIDIA GeForce RTX 3090 GPUs running Ubuntu 22.04 with a 32-core and 256GiB RAM. The large language model (LLM) employed was qwen2-72b[50]. We utilized the constructed UAV fault knowledge graph corpus—consisting of 1,000 fault handling records/articles and 20 MB of fault-related technical text manuals—as the RAG experiment dataset. This study investigates three RAG frameworks for assisted diagnosis: HybridRAG, VectorRAG, and GraphRAG_


Agent Prompt

You are an artificial intelligence assistant specifically designed for troubleshooting. Please always use the provided tools to solve tasks and do not give answers without sufficient information. When providing troubleshooting steps based on the context, please ask the user step by step and provide further recommendations based on their feedback.

Example:
Assistant: First, check component A.
User: Component A exhibits phenomenon B.
Assistant: This phenomenon is caused by C. Now, please check component D...

The user has no professional knowledge, so you should guide them as precisely and concisely as possible.

Fig. 7. Structured Instruction Prompts for Agent-Guided Troubleshooting. The blue highlighted block represents the request, which specifies the assistant's mandatory inference rules including step-by-step verification requirements. The yellow highlighted block provides the example, demonstrating the expected interaction sequence. This two-component design ensures context-aware guidance while maintaining strict diagnostic protocols.

sub (subgraph retrieval-based). For each query, HybridRAG was configured with a maximum of 5 iterations, returning the most relevant results through top-10 ranking[51] while maintaining a default subgraph depth of 3. The subgraph depth of 3 was determined through a comprehensive analysis of domain-specific schema design and instance data characteristics, ensuring sufficient knowledge coverage for fault diagnosis queries. When the default retrieval results are insufficient, the planner-agent autonomously adjusts the subgraph retrieval depth to meet task-specific requirements.

In our experiments, we evaluated the methods using four key metrics—F1-score, hallucination rate, end-to-end latency, and RAM utilization—to assess both accuracy and practical deployment feasibility. Specifically, the F1-score quantifies diagnostic accuracy, while the hallucination rate measures the frequency of irrelevant or incorrect outputs generated during inference. End-to-end latency and RAM utilization were used to assess computational efficiency and resource consumption. Notably, traditional statistical significance tests (e.g., confidence intervals, p-values) were not applied to evaluate the robustness of these metrics due to the non-independent and identically distributed (non-IID) nature of textual data, which violates fundamental statistical assumptions. Our metric selection follows established practices in LLM, knowledge graph, and RAG research communities, where absolute performance improvements typically outweigh statistical validation[22,36].

1) F1-score
The F1-score evaluates answer prediction quality by combining precision and recall metrics. For each diagnostic query, this is formally computed as

$$\mathrm{F1}(\%\ ) = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \times 100\% \tag{18}$$

$$\mathrm{Precision} = \frac{count\_pred}{len\,(\hat{a})} \tag{19}$$

$$\mathrm{Recall} = \frac{count\_pred}{len\,(a)} \tag{20}$$

where $\hat{a}$denotes the predicted answer, a represents the ground truth answer, $len\,(\hat{a})$ and $len\,(a)$ indicate the length of the predicted and true answers respectively, and$count_pred$corresponds to the number of matching tokens between $\hat{a}$ and a.

2) Hallucination Rate
The hallucination rate (HR) quantifies the frequency of generating content that does not contain real information. Formally, HR is computed as

$$\mathrm{HR}(\%\ ) = \frac{\sum_{i=1}^{N} h(\hat{a}_i, a_i)}{N} \times 100\%, \quad i = 1, 2, \cdots, N \tag{21}$$

where$\hat{a}_i$ and a_i denotes respectively predicted and ground truth answers for the i-th question, N denotes the total number of questions, $h(\hat{a}_i, a_i)$denotes the hallucination score for the i-th question, which is ranged from 0 to 1 and implemented via "LLM as a judge" between ground truth and generated answers by the system.

B. Experimental Results
1) Overall performance
The performance of the proposed HybridRAG method and baseline approaches (VectorRAG and GraphRAG_sub) was evaluated using 100 domain-specific fault diagnosis queries derived from real-world aviation scenarios. Experimental results, summarized in Table 1, demonstrate that HybridRAG significantly outperforms its counterparts in both diagnostic accuracy and reliability. Notably, HybridRAG achieves an F1-score of 84.83%, representing a 4% relative improvement over the next-best GraphRAG_sub (80.12%), while reducing the hallucination rate (HR) to 16.67% —a 7% reduction compared to VectorRAG's 24.05% HR. These results underscore the efficacy of HybridRAG's hybrid retrieval strategy in minimizing diagnostic errors and improving decision-making precision. Regarding resource consumption, HybridRAG incurs higher computational costs due to the Graph Index system overhead—specifically, a 41.19-second inference latency (versus 32.81–33.51 s for baselines) and 26.12 GiB RAM usage (versus 16.91–18.11 GiB). However, these costs are justified by the substantial accuracy gains. Furthermore, in practical troubleshooting scenarios, the 41.19-second latency per query is acceptable when contrasted with the total troubleshooting duration, which typically requires one hour

method	F1-score	HR	Latency*	RAM
VectorRAG	76.59%	35.43%	32.81 s	16.91 GiB
GraphRAG_sub	80.12%	24.05%	33.51 s	18.11 GiB
HybirdRAG(ours)	84.83%	16.67%	41.19 s	26.12 GiB

Table 1. The performance results of some methods.

method	F1-score	HR
VectorRAG	69.54%	46.05%
GraphRAG_sub	75.69%	23.76%
HybirdRAG(ours)	81.50%	15.01%

Table 2. The performance results of some methods for new fault.

Category	Example
Statistics	How many cases have occurred where the left brake pressure is too low?
Fact	What are the components of the brake system?
Cause	What causes the left normal brake pressure to be too low?
Troubleshooting	How to troubleshoot the left brake pressure being too low?
Preventive	How to maintain normal brake pressure on the left side?

Table 3. The question categories and examples.

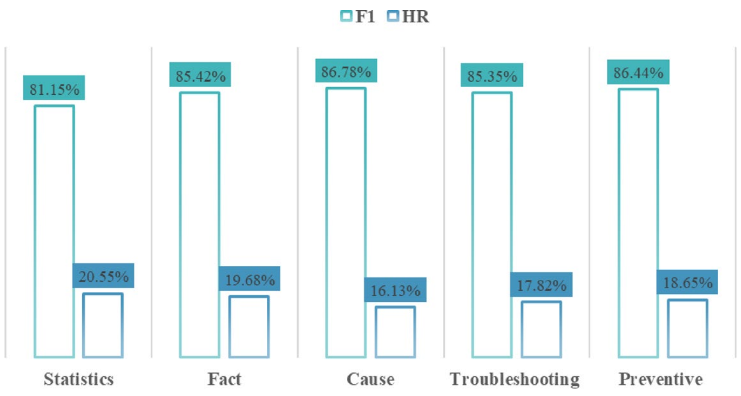


Fig. 8. The performances of five categories of fault-related questions on HybirdRAG method.

or more of technician time. Critically, in this domain, diagnostic precision outweighs response speed, as even minor inaccuracies, such as hallucinations that mislead engineers, can derail diagnostic efforts and substantially extend troubleshooting timelines.

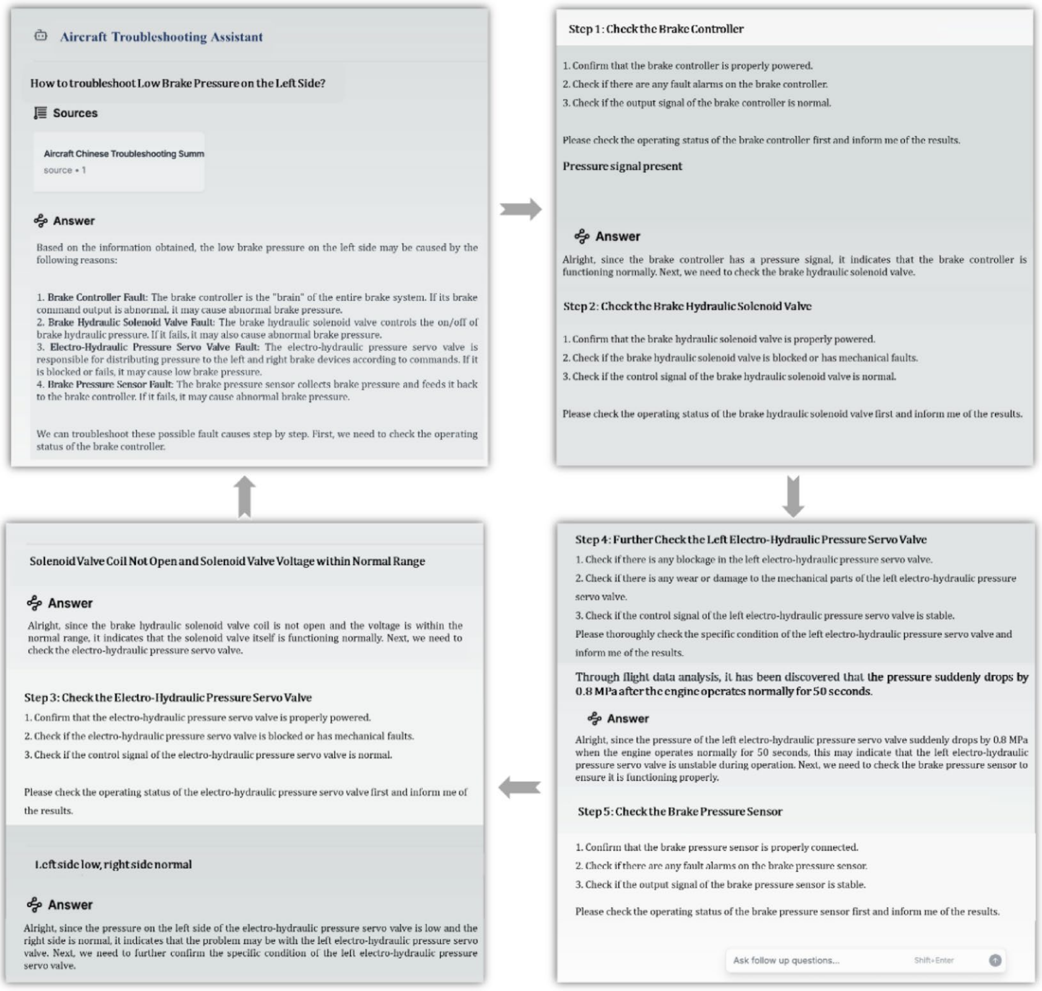
*Latency denotes the average time taken to process 100 questions, measured from the start of retrieval to the completion of answer generation.

In addition, we discovered through BM25 retrieval analysis that 82 out of 100 collected fault queries were entirely novel—their corresponding fault cases, root causes, or solutions were absent in the training corpus, but their underlying operating principles and structural configurations were fully covered in the dataset. We thus designated these 82 query pairs as a dedicated benchmark for assessing generalization, with results presented in Table 2. The findings demonstrate that graph-based reasoning outperforms baseline methods for unseen queries. Specifically, HybridRAG exhibits a smaller performance decline compared to historical data in terms of F1-score (84.83% → 81.50%) and HR (16.67%→ 15.01%), suggesting a more effective extraction of the contextual information required for robust knowledge reasoning in unseen fault scenarios.

2) Robustness

To validate the robustness of the HybirdRAG method, we collected five categories of aircraft fault-related questions as detailed in Table 3, with 100 distinct test queries per category. Figure 8 presents the evaluation results, showing that the proposed method consistently achieves F1-scores above 80% while keeping hallucination rates below 30% across all question categories, which confirms its stable diagnostic performance. However, HybridRAG exhibits performance variations across different question types due to varying inference complexities. Statistical questions, in particular, are more challenging due to their complex reasoning requirements. For instance, when answering *"What percentage of a specific root cause's frequency accounts for all possible causes of a given fault phenomenon?*", the model must first infer all plausible root causes and then calculate their relative frequencies. This two-step process is prone to hallucinations, such as inventing non-existent cause distributions or miscounting statistical patterns.

Further analysis reveals that HybridRAG exhibits performance limitations in specific fault scenarios, particularly for root cause identification and judgment of abnormal signal parameter characteristics. These challenges originate from both knowledge graph schema ambiguities and corpus data quality issues. The schema


Aircraft Troubleshooting Assistant

How to troubleshoot Low Brake Pressure on the Left Side?

Sources
Aircraft Chinese Troubleshooting Summ
source • 1
Answer

Based on the information obtained, the low brake pressure on the left side may be caused by the following reasons:

1. **Brake Controller Fault**: The brake controller is the "brain" of the entire brake system. If its brake command output is abnormal, it may cause abnormal brake pressure.
2. **Brake Hydraulic Solenoid Valve Fault**: The brake hydraulic solenoid valve controls the on/off of brake hydraulic pressure. If it fails, it may also cause abnormal brake pressure.
3. **Electro-Hydraulic Pressure Servo Valve Fault**: The electro-hydraulic pressure servo valve is responsible for distributing pressure to the left and right brake devices according to commands. If it is blocked or fails, it may cause low brake pressure.
4. **Brake Pressure Sensor Fault**: The brake pressure sensor collects brake pressure and feeds it back to the brake controller. If it fails, it may cause abnormal brake pressure.

We can troubleshoot these possible fault causes step by step. First, we need to check the operating status of the brake controller.

Step 1: Check the Brake Controller

1. Confirm that the brake controller is properly powered.
2. Check if there are any fault alarms on the brake controller.
3. Check if the output signal of the brake controller is normal.

Please check the operating status of the brake controller first and inform me of the results.

Pressure signal present

Alright, since the brake controller has a pressure signal, it indicates that the brake controller is functioning normally. Next, we need to check the brake hydraulic solenoid valve.

Step 2: Check the Brake Hydraulic Solenoid Valve

1. Confirm that the brake hydraulic solenoid valve is properly powered.
2. Check if the brake hydraulic solenoid valve is blocked or has mechanical faults.
3. Check if the control signal of the brake hydraulic solenoid valve is normal.

Please check the operating status of the brake hydraulic solenoid valve first and inform me of the results.

Solenoid Valve Coil Not Open and Solenoid Valve Voltage within Normal Range

Alright, since the brake hydraulic solenoid valve coil is not open and the voltage is within the normal range, it indicates that the solenoid valve itself is functioning normally. Next, we need to check the electro-hydraulic pressure servo valve.

Step 3: Check the Electro-Hydraulic Pressure Servo Valve

1. Confirm that the electro-hydraulic pressure servo valve is properly powered.
2. Check if the electro-hydraulic pressure servo valve is blocked or has mechanical faults.
3. Check if the control signal of the electro-hydraulic pressure servo valve is normal.

Please check the operating status of the electro-hydraulic pressure servo valve first and inform me of the results.

Left side low, right side normal

Alright, since the pressure on the left side of the electro-hydraulic pressure servo valve is low and the right side is normal, it indicates that the problem may be with the left electro-hydraulic pressure servo valve. Next, we need to further confirm the specific condition of the left electro-hydraulic pressure servo valve.

Step 4: Further Check the Left Electro-Hydraulic Pressure Servo Valve

1. Check if there is any blockage in the left electro-hydraulic pressure servo valve.
2. Check if there is any wear or damage to the mechanical parts of the left electro-hydraulic pressure servo valve.
3. Check if the control signal of the left electro-hydraulic pressure servo valve is stable.

Please thoroughly check the specific condition of the left electro-hydraulic pressure servo valve and inform me of the results.

Through flight data analysis, it has been discovered that **the pressure suddenly drops by 0.8 MPa after the engine operates normally for 50 seconds.**

Alright, since the pressure of the left electro-hydraulic pressure servo valve suddenly drops by 0.8 MPa when the engine operates normally for 50 seconds, this may indicate that the left electro-hydraulic pressure servo valve is unstable during operation. Next, we need to check the brake pressure sensor to ensure it is functioning properly.

Step 5: Check the Brake Pressure Sensor

1. Confirm that the brake pressure sensor is properly connected.
2. Check if there are any fault alarms on the brake pressure sensor.
3. Check if the output signal of the brake pressure sensor is stable.

Please check the operating status of the brake pressure sensor first and inform me of the results.

Fig. 9. Interactive fault diagnosis process guided by the HybridRAG agent.

design introduces reasoning inconsistencies when edge types are misinterpreted—the BelongTo relationship between Signal and Unit entities is occasionally applied incorrectly to Unit-Unit connections, while the explicitly defined Cause relationship between Fault entities is sometimes misconstrued as linking Fault to Fault Case, despite prompt-based constraints. Parallel to these schema challenges, the original corpus contains unstructured maintenance records featuring vague descriptions, inconsistent terminology, and missing technical specifications. These data quality limitations introduce substantial noise during information retrieval, particularly affecting experience-based articles written by maintenance personnel.

The experimental results conclusively demonstrate that HybridRAG's innovative integration of multi-dimensional retrieval strategies yields significant improvements in fault diagnosis efficiency. This enhanced performance, particularly evident in complex diagnostic scenarios, underscores the methodological superiority of the hybrid approach and its critical value for practical aviation maintenance applications. Although schema ambiguities and data quality limitations present identifiable challenges, they represent addressable constraints rather than fundamental flaws—offering clear pathways for future methodological refinement. Collectively, these empirical results establish HybridRAG as a practically viable solution, providing a robust foundation for its ongoing implementation in aircraft fault diagnosis systems.

C. Agent-Guided troubleshooting

Based on the proposed HybirdRAG method and a designed system prompt framework, we developed an interactive troubleshooting guidance agent. As demonstrated in Fig. 9, the agent guides users through a structured fault diagnosis process. For example, when a user inquiries about "how to troubleshoot low pressure in the left normal brake system," the agent provides all possible causes of the fault along with detailed troubleshooting steps. The user follows these steps to perform tests and reports the results back to the agent, which then determines and suggests the next steps until the specific fault cause and component are identified. This guided fault diagnosis agent emulates an expert's thought process. Compared to traditional question-and-answer systems, it more effectively guides users through complex fault diagnosis tasks. This interactive model both troubleshooting efficiency and user experience, demonstrating the practical value of this technology in aircraft fault diagnosis applications.

Conclusion

In this paper, we propose a guided fault diagnosis method for aircraft based on the HybridRAG framework, which integrates knowledge graphs and large language models. By combining multiple retrieval strategies, including VectorRAG, BM25, and graph-based reasoning, our method enhances inference capabilities for complex fault diagnosis through the synergistic use of structured knowledge and unstructured data. We further develop an intelligent diagnostic agent that implements a multi-turn conversational functionality to guide maintenance personnel through troubleshooting process, significantly improving interactivity, and operational efficiency in real-world aviation maintenance scenarios.

However, despite its effectiveness, our experiments reveal that the proposed method can still fail due to issues such as schema ambiguities and low data quality. To address these, we plan to refine the domain-specific schema by leveraging expert knowledge as a foundation and using large language models (LLMs) to identify inconsistencies and gaps in the current design, thereby eliminating ambiguities and ensuring completeness—a one-time upfront cost since the finalized schema remains unchanged during actual deployment. Furthermore, we will incorporate reinforcement learning (RL) techniques to train the model to critically evaluate the reliability of input data, adopting an offline-training/online-inference framework that ensures only marginal inference costs during deployment post-training, enabling it to prioritize credible data sources during the reasoning process and reduce the impact of noisy inputs. These improvements will enhance HybridRAG's robustness without compromising deployability.

Beyond addressing the current limitations, we recognize that the scope of diagnostic data needs extension. While this study focuses on unstructured text data, real-world aircraft maintenance relies on diverse data sources including fault images, diagnostic videos, and sensor readings. Integrating these modalities will enable more comprehensive fault analysis—particularly for complex cases where visual inspection or sensor trends provide critical supplementary evidence. Moving forward, we will explore unified multimodal reasoning methods to jointly interpret textual, visual, and time-series data. This expansion is expected to improve robustness in handling ambiguous or rare faults while broadening the system's applicability across different maintenance scenarios.

Data availability

The datasets analysed during the current study are not publicly available due to confidentiality and sensitive information but some de-identified data can be available partly from the corresponding author on reasonable request.

Received: 23 December 2024; Accepted: 14 May 2025
Published online: 22 May 2025

References

1. Aamodt, A., Plaza, E., Case-Based & Reasoning Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, vol. 7, no. 1, pp. 39–59, (1994).
2. Jackson, P. *Introduction To Expert systems[M]* (Addison-Wesle, Jan, 1998).
3. Yurin, A. et al. Towards an intelligent decision support system for aircraft troubleshooting. *Proceedings of 10th International Conference on Recent Advances in Civil Aviation*, (2022).
4. Qiu, L. et al. Method for Building and Applying Knowledge Graphs for Unmanned Aerial Vehicle Fault Diagnosis. *Computer Engineering and Applications*, vol. 59, no. 9, pp.9–14, (2023).
5. Xu, J. X. et al. Construction and Application of Knowledge Graphs in Engine Fault Diagnosis. *Computer Systems & Applications*, vol. 7, no. 31, pp.66–76, (2022).
6. Wu, C., Zhang, L., Tang, X. L., Cui, L. J. & Xie, X. Y. Construction and application of fault knowledge graphs for aircraft engine lubrication systems. *J. Beijing Univ. Aeronaut. Astronaut.* **4** (50), 1336–1346 (2024).
7. Tang, X. L. et al. Exploring research on the construction and application of knowledge graphs for aircraft fault diagnosis. *Sensors* **11** (23), 5295–5313 (2023).
8. Peng, C. Y., Xia, F., Mehdia, N. & Francesco, O. Knowledge Graphs: Opportunities and Challenges, arXiv:2303.13948v1. (2023).
9. Meng, X. et al. Fault knowledge graph construction and platform development for aircraft PHM. *Sensors* **24**, 231–252 (2024).
10. Mishra, R. & Shridevi, S. Knowledge graph driven medicine recommendation system using graph neural networks on longitudinal medical records. *Sci Rep*, **14**, 25449, (2024).
11. Zhao, W. X. et al. A survey of large language models, arXiv.2303.18223. (2023).
12. Dale, R. Start-up activity in the LLM ecosystem. *Nat. Lang. Eng.* **30** (3), 650–659 (2024).
13. Li, Y. F., Wang, H. & Sun, M. ChatGPT-Like Large-Scale Foundation Models for Prognostics and Health Management: A Survey and Roadmaps, arXiv:2305.06472. (2023).
14. Mittelstädt, J. M. et al. Large Language models can outperform humans in social situational judgments. *Sci Rep*, **14**, (2024).
15. Ji, Z. W. et al. Survey of hallucination in natural Language generation. *Comput. Surv.* **55** (12), 1–38 (2023).
16. Hao, G. et al. Quantifying the uncertainty of LLM hallucination spreading in complex adaptive social networks. *Sci Rep*, **14**, 16375, (2024).
17. Liu, P., Qian, L., Zhao, X. & Tao, B. The construction of knowledge graphs in the aviation assembly domain based on a joint knowledge extraction model. *IEEE Access.* **11**, 26483–26495 (2023).
18. Sun, Y. et al. JointLK: Joint reasoning with language models and knowledge graphs for commonsense question answering. In Proc. 2022 Annual Conference of the North American Chapter of the Association for Computational Linguistics-Human Language Technologies, pp. 5049–5060, (2022).
19. Guu, K. et al. REALM: Retrieval-augmented language model pre-training, arXiv:2002.08909. (2020).
20. Gao, Y. F. et al. Retrieval-augmented generation for large language models: A survey, arXiv:2312.10997.8. (2023).
21. Church, K. W. et al. Emerging trends: a gentle introduction to RAG. Natural Language Engineering. **30**, 4, pp. 870–881, (2024).
22. Liu, P. F. et al. Joint knowledge graph and large Language model for fault diagnosis and its application in aviation assembly. *IEEE Trans. Industr. Inf.* **20** (6), 8160–8169 (2024).

23. Liu, L. et al. A Survey on Medical Large Language Models: Technology, Application, Trustworthiness, and Future Directions, arXiv.2406.03712. (2024).
24. Nie, Y. Q. et al. A Survey of Large Language Models for Financial Applications: Progress, Prospects and Challenges, arXiv.2406.11903. (2024).
25. Wang, Y. et al. Knowledge graph prompting for multi-document question answering. *In Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp.19206–19214, (2024).
26. Gao, Y. F. et al. Retrieval-augmented generation for large language models: A survey, arXiv:2312.10997. (2023).
27. Luo, M. et al. A Study on the Efficiency and Generalization of Light Hybrid Retrievers, arXiv:2210.01371. (2022).
28. Juvekar, K. & Purwar, A. Cos-mix: cosine similarity and distance fusion for improved information retrieval, (2024). arXiv:2406.00638.
29. Zhang, J. et al. Subgraph retrieval enhanced model for Multi-hop knowledge base question answering. *Association Comput. Linguistics*. **1**, 5773–5784 (2022).
30. He, G. L. et al. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. *In Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 553–561, (2021).
31. Li, X. X. et al. Chain-of-Knowledge: Grounding Large Language Models via Dynamic Knowledge Adapting over Heterogeneous Sources, arXiv.2305.13269. (2024).
32. He, X. X. et al. G-Retriever: Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering, arXiv.2402.07630. (2024).
33. Edge, D. et al. From local to global: A graph RAG approach to Query-Focused summarization. *Comput. Lang.* arXiv.2404.16130 (2024).
34. Luo, L. H. et al. Reasoning on graphs: faithful and interpretable large Language model reasoning. *ICLR*, (2024). arXiv:2310.01061v2.
35. Maciej, B. et al. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. *In Proceedings of the AAAI Conference on Artificial Intelligence*, arXiv:2308.09687. (2024).
36. Diego, S. et al. KG-RAG: Bridging the Gap Between Knowledge and Creativity, arXiv:2405.12035. (2024).
37. Church, K. W. et al. Emerging trends: a gentle introduction to RAG. *Nat. Lang. Eng.* **30** (4), 870–881 (2024).
38. Geng, Z., Zhang, Y. & Han, Y. Joint entity and relation extraction model based on rich semantics. *Neurocomputing*, **429**, 1, pp. 132–140, (2021).
39. Yuan, Y. et al. A relation-Specific attention network for joint entity and relation extraction. *In IJCAI*, pp. 4054–4060. (2020).
40. Shan, Y., Lu, H. & Lou, W. A hybrid attention and dilated convolution framework for entity and relation extraction and mining. *Sci Rep*, vol. 13, (2023).
41. Nayak, T. & Ng, H. T. Effective modeling of encoder-decoder architecture for joint entity and relation extraction. *In Proceedings of the Thirty-Fourth AAAI conference on artificial intelligence*, Hilton New York Midtown, New York, USA, pp. 8528–8535, (2020).
42. Drury, B., Oliveira, G. & de Andrade Lopes, H. A survey of the extraction and applications of causal relations. *Nat. Lang. Eng.* **28** (3), 361–400 (2022).
43. Reimers, N. & Gurevych, I. Sentence-BERT: sentence embeddings using Siamese BERT-Networks. *EMNLP* arXiv:1908.10084 (2019).
44. Malkov, Y. A. & Yashunin, D. A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* **42** (4), 824–836 (2020).
45. Robertson, S. & Zaragoza, H. The probabilistic relevance framework: BM25 and beyond. *Foundations & Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, (2009).
46. Joaquín, P. I. et al. Integrating the probabilistic models BM25/BM25F into Lucene. (2009). arXiv.0911.5046.
47. Gao, L. Y., Dai, Z. Y. & Callan, J. Rethink training of BERT rerankers in Multi-Stage retrieval pipeline. *ECIR* https://doi.org/10.48550/arXiv.2101.08751 (2021).
48. Pradeep, R., Sahel, S., Jimmy, R. & RankZephyr Effective and Robust Zero-Shot Listwise Reranking is a Breeze, arXiv.2312.02724. (2023).
49. Glass, M. et al. Re2G: retrieve, rerank, generate. *NAACL* arXiv.2207.06300 (2022).
50. An, Y. & Song, B. &Hui, B. Y.Qwen2: Technical Report, arXiv.2407.10671. (2024).
51. Han, H. et al. RAG vs. GraphRAG: A Systematic Evaluation and Key Insights, arXiv:2502.11371. (2025).

Author contributions
X.Y.X. wrote the main manuscript text. X.L.T and L.J.C. dealt with data collection and preparation. S.W.G. wrote experiment codes in the work. All authors reviewed the manuscript.

Funding
This work was supported by the National Natural Science Foundation of China (No. 72201276).

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-025-02643-2.

Correspondence and requests for materials should be addressed to X.T.