Aviation Parser: A Knowledge-Guided Self-Evolving Optimization Framework with LLMs for NOTAM Understanding

Anonymous ACL submission

Abstract

Accurate parsing of Notices to Airmen (NOTAMs) constitutes a critical requirement for aviation safety, yet existing methods suffer from template rigidity that impedes effective handling of non-standard syntax, regional expression ambiguities, and the semantic-practice gap. We propose a knowledge-guided self-evolving optimization framework that integrates Large Language Models (LLMs) with an Aviation Knowledge Graph (AviationKG) to achieve efficient structured NOTAM parsing. The framework comprises three innovative modules: 1) Knowledge-Enhanced Retrieval (KG-TableRAG), which resolves semantic ambiguities through binding of knowledge graph relations with infrastructure tables to constrain search spaces; 2) Self-Evolving Optimization (SEVO), employing dynamic preference alignment and error-driven curriculum learning to iteratively enhance complex instruction compliance; 3) Consensus Inference Engine (CIE), improving edge-case robustness via terminology-preserved input diversification and majority voting decoding. Experimental results demonstrate that our framework achieves a *30.4%* accuracy improvement over the base model within 3-5 iterations on a labeled dataset of 10,000 global NOTAMs, with ablation studies confirming the collaborative efficacy of modular components. This research establishes the first knowledge-driven, continuously optimized LLM solution for aviation text parsing, whose methodology demonstrates extensibility to other high-precision-demanding professional domains.

1 Introduction

Accurate interpretation of NOTAMs (Notice to Airmen) constitutes a critical yet challenging component of modern flight operations. These specialized bulletins contain time-sensitive information regarding temporary airspace restrictions and navigational hazards, characterized by linguistic features distinct from conventional technical documentation. With over one million active NOTAMs published annually worldwide (Morarasu and Roman, 2024), the aviation industry urgently requires robust automated analysis systems to reduce manual workload and mitigate human processing errors. Existing systems predominantly rely on regular expression-based template matching, leading research efforts to focus primarily on automated rule discovery or basic NOTAM classification (Dieter et al., 2024; Mi et al., 2022a).

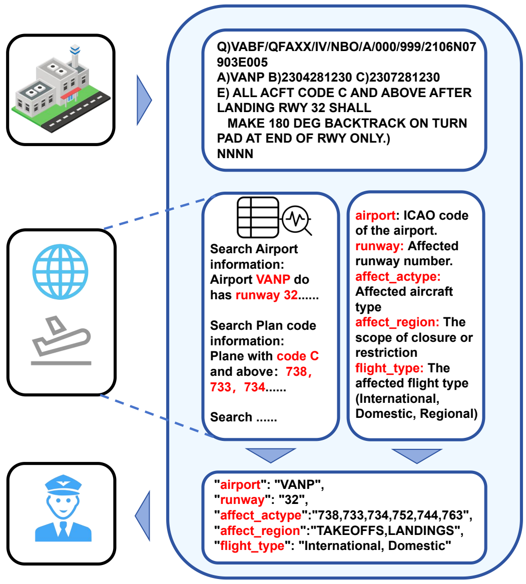

Figure 1: An illustration of NOTAM analysis task

However, NOTAMs present unique parsing challenges due to their heavy dependence on 300+ standardized abbreviations and non-standard syntactic structures (e.g., "RWY 09L/27R CLSD DUE BIRD ACT"), which frequently violate conventional parsing rules. Furthermore, practical scenarios introduce additional complexity through regional expression variations (e.g., "EGBA" encompassing both EGBA1A and EGBA1B) and typographical/grammatical errors. Beyond syntactic challenges, NOTAM processing faces a fundamental **semantic-practice gap**: the disconnect between textual descriptions and operational impacts requires implicit correlation with aviation infrastructure status. For instance, interpreting "APCH LGT U/S" necessitates knowledge of specific runway configurations, yet NOTAMs may reference non-existent runways or omit critical identifiers (Patel et al., 2023). These operational constraints rely on factual data from regularly updated official sources, as illustrated in Figure 1.

The emergence of large language models (LLMs) with advanced natural language understanding capabilities opens new frontiers for NOTAM analysis. While no prior studies specifically address LLM applications in this domain, recent breakthroughs in complex instruction following and generic information extraction (Morarasu and Roman, 2024) establish critical technical foundations. Building on these advances, we present the first LLM-adapted framework for NOTAM analysis featuring three pioneering contributions:

- **Knowledge-Driven Architecture**: Innovating the inaugural application of LLMs to NOTAM parsing, our framework integrates an aviation knowledge graph with TableRAG retrieval to overcome domain-specific challenges through constraint-aware information extraction.
- **Self-Optimizing Pipeline**: Through synergistic integration of dynamic preference alignment, error-driven curriculum learning, and consensus inference mechanisms, we establish an end-to-end optimizable system capable of self-evolution without manual intervention.
- **Empirical Performance Leap**: Experimental validation demonstrates our optimized model achieves a 30.4% accuracy improvement over base LLMs, with multi-perspective analysis and majority voting decoding.

2 Related Work

2.1 NOTAM Analysis

Natural Language Processing (NLP) has emerged as a cornerstone technology in reducing manual operations in the aviation industry, particularly in NOTAM (Notice to Airmen) analysis(Mogillo-Dettwiler, Year (if available; Mi et al., 2022b). Researchers from Lucerne University of Applied Sciences and Skyguide demonstrated the potential of transformer-based models by training on 100,000 unlabeled NOTAMs to implement an "Intelligent NOTAM" service, showing significant promise in automatically filtering irrelevant information and rectifying inconsistencies in raw NOTAMs (Bravin et al., 2020). Similarly, Clarke et al. (2021) explored NLP workflows using a comprehensive dataset of 3.73 million NOTAMs. Their methodology, which combined TF-IDF, topic modeling, and Named Entity Recognition (NER), provided valuable insights into automated segmentation and tagging of structured content within NOTAMs. Further advancing this field, Airbus AI's 2022 study expanded the application of NLP in NOTAM parsing by utilizing pre-trained BERT models on 1.2 million NOTAMs for aviation knowledge extraction (Arnold et al., 2022). While these pioneering studies have made significant contributions, they collectively highlight several unresolved challenges in large-scale NOTAM processing (Morarasu and Roman, 2024). These include handling ambiguous abbreviations, semantic-practical mismatches, and adaptation to diverse input sources with regional variations in expression patterns. Our work builds upon these insights, presenting a more sophisticated and practical approach to NOTAM analysis tasks, with a particular focus on enhancing the adaptability and efficiency of Large Language Models (LLMs) in specialized NOTAM parsing systems.

2.2 Large Language Models

The rapid advancement of large language models (LLMs)(Zhao et al., 2023) has driven transformative progress across specialized domains. NOTAM analysis presents unique challenges that require the integration of three critical capabilities: **information extraction**, **tabular understanding**, and **complex instruction following**. Transformer-based architectures (Brown et al., 2020; Chowdhery et al., 2022), enhanced through breakthroughs in parameter scaling (Rae et al., 2021; Le Scao et al., 2022), demonstrate exceptional few-shot learning capabilities particularly suited for aviation domains with sparsely labeled data (Xu et al., 2023).

Information Extraction techniques have evolved into two paradigms: in-context learning via prompt engineering (Li et al., 2023) and supervised fine-tuning with instruction-aware datasets (Wang et al., 2023). While innovations such as code-style prompting (Sainz et al., 2024) and hierarchical schema representations (Li et al.,

2024) enhance output consistency, conventional methods underperform when processing aviation terminology with dynamic semantic constraints and regional expression variations.

Tabular Understanding methodologies have transitioned from schema-dependent Text2SQL systems (Zhong et al., 2017) to neurosymbolic approaches exemplified by TableRAG (Chen et al., 2024). Although TableRAG's query expansion mechanisms mitigate large-scale table processing challenges, two persistent limitations remain in NOTAM contexts: context window constraints during full-table encoding and cell localization inaccuracies under schema sparsity.

Complex Instruction Following research demonstrates that progressively intensified constraints enhance model compliance (Mukherjee et al., 2023; Luo et al., 2024). Frameworks like Conifer's progressive learning (Sun et al., 2024) show potential for multi-level constraint handling. Building upon these insights, we employ curriculum learning strategies to optimize model performance against instruction constraint heterogeneity caused by random complexity distribution in dataset samples.

3 The Proposed Framework

3.1 Problem Formulation

Given an input NOTAM text sequence $X = [x_1, \dots, x_n]$ and a collection of aviation reference tables $\mathcal{T} = \{T_1, \dots, T_m\}$, our objective is to extract structured aviation information through a knowledge-enhanced generative framework. Formally, the task is defined as maximizing the conditional probability:

$$p_\theta(Y \mid X, P, K) = \prod_{i=1}^{m} p_\theta(Y_i \mid X, P, K, Y_{<i}), \quad (1)$$

where $Y = [Y_1, \dots, Y_m]$ denotes the target structured output sequence, θ represents the parameters of the large language model (LLM), P encapsulates task-specific prompts and instructions, and $K = \kappa(X, \mathcal{T})$ corresponds to factual knowledge retrieved from $\mathcal{T}$.

3.2 Framework Overview

As illustrated in Fig. 2, our framework operates through three synergistic stages: (1) The *Retrieval Stage* grounds predictions in aviation domain knowledge via dynamic table retrieval; (2) The *Optimization Stage* enables iterative self-improvement of the foundation model through adaptive preference learning; (3) The *Inference Stage* ensures robust parsing via diversified input generation and consensus decoding. This architecture systematically addresses domain-specific challenges in NOTAM analysis, including knowledge grounding, error propagation, and operational stability.

3.3 Knowledge-Guided TableRAG

To ensure factual consistency in NOTAM parsing results, this study proposes a knowledge graph-enhanced Table Retrieval-Augmented Generation framework (KG-TableRAG). The methodology integrates real-time updated aviation infrastructure data tables to address critical limitations of conventional TableRAG in specialized domains. Traditional table retrieval methods exhibit domain-specific retrieval bias due to insufficient structural knowledge representation in aviation. For instance, "runway closure" events may involve implicit cross-table correlations with lighting systems and navigation equipment, which conventional vector retrieval mechanisms fail to capture. Furthermore, existing REACT-based table retrieval methods suffer from multi-step reasoning inefficiencies, rendering them impractical for time-sensitive operational scenarios.

The proposed KG-TableRAG framework enhances TableRAG (Chen et al., 2024) performance through systematic integration of knowledge graphs. While leveraging open-source methodologies for automated knowledge graph construction, manual refinements were applied to portions of the automatically generated graphs to optimize performance, given the limited availability of structured corpora. Upon receiving raw NOTAMs (Notices to Airmen), the framework employs LLMs to decompose queries, executes graph queries based on extracted keywords, and subsequently performs vector searches. For a detailed illustration of the domain knowledge graph architecture, refer to Figure 4 in Appendix B.

Implementation specifics include explicit mappings between knowledge nodes and table columns, such as dynamically binding the graph relationship `[Airport]→[Owns]→[Runway]` to operational columns like `RWY-STATUS`. This design constrains the search space to mitigate interference from irrelevant columns. A lightweight single-step inference mechanism replaces traditional multi-

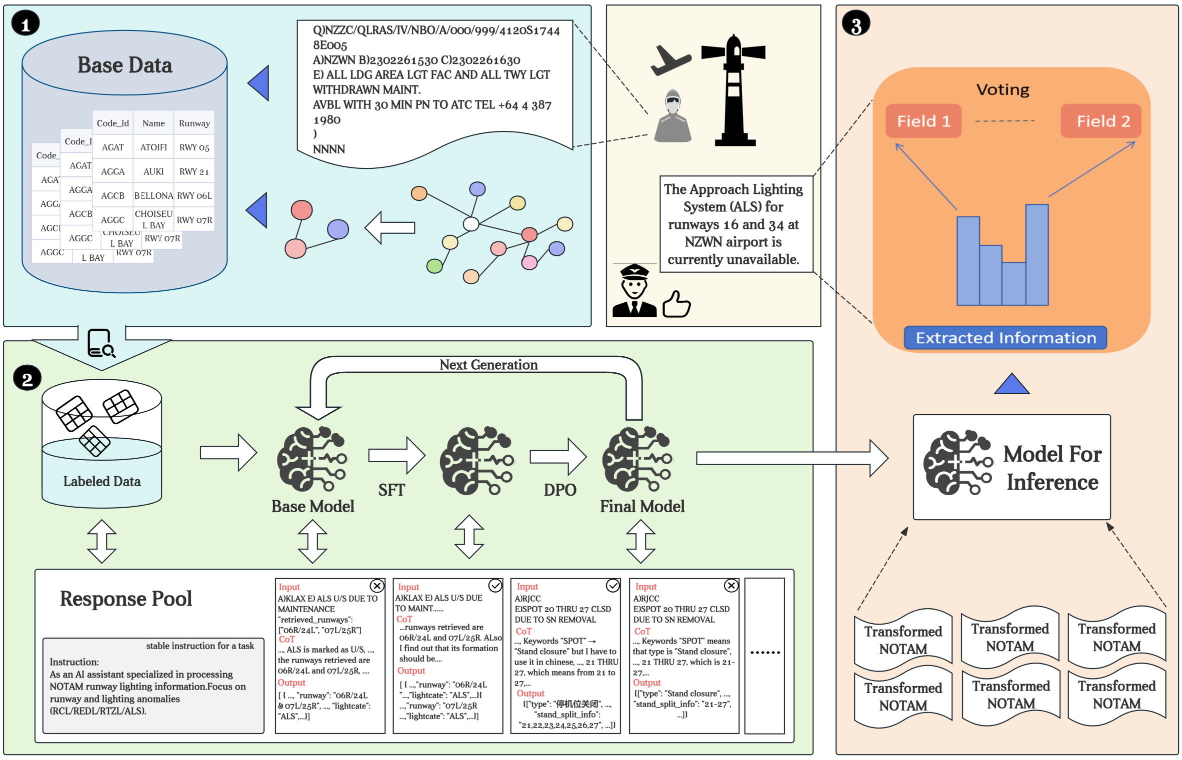

Figure 2: Overall framework of the proposed . (1) Retrieval Stage: The final outputs are based on a set of base tables that represent real-world conditions, e.g., the number of runways at an airport. (2) Optimization Stage: Our foundational model gains proficiency in handling complex instructions within NOTAM analysis scenarios through iterative self-evolution. (3) Inference Stage: We rephrase the original NOTAM without altering its core content and then extract information from multiple texts to determine the final answer via a voting mechanism.

round decision processes by utilizing predefined graph paths (e.g., the chained pattern *"Restriction Type-Impacted Equipment-Applicable Time Period"*) for direct Cypher query generation.

The operational implementation prioritizes SMOLAGENTS over REACT due to their enhanced efficiency in task-specific processing. SMOLAGENTS demonstrate superior computational efficiency and scalability, particularly in complex query processing within knowledge graph-integrated systems. This strategic substitution streamlines operational workflows while improving system robustness and responsiveness. Collectively, these enhancements yield measurable improvements in both accuracy and operational efficiency during NOTAM information retrieval and analysis.

3.4 Self-Evolving Supervised and Preference Optimization

Initialization Setup Our iterative optimization framework is initialized with three components:

- **Data Partitioning**: Annotated dataset $\mathcal{D}_0 = \{(x \circ K, Y^*)\}$ is split into 8:2 training-test sets, where x is the raw NOTAM text, $K = \kappa(x, \mathcal{T})$ denotes retrieved aviation knowledge, and Y^* is the structured output annotation.
- **Base Model**: An untuned open-source base model π_{base} serves as the initial model.
- **Response Repository**: An indexed set $\mathcal{R} = \{(x, Y^*, \hat{Y})\}_{x \in \mathcal{X}}$ stores model responses with correctness labels across iterations.

Iterative Optimization Loop

Each iteration consists of supervised fine-tuning (SFT) and dynamic preference optimization (DPO) stages. The workflow is illustrated in Fig. 2 (see Algorithm 1 in Appendix A).

In the first stage, we generate responses $\hat{Y}$ for inputs $x \circ K$ using the current model π_{current}. Next, we compare $\hat{Y}$ with the golden labels Y^* to update the repository $\mathcal{R}$ with both correct ($\mathcal{Y}_x^*$) and incorrect ($\mathcal{Y}_x^-$) responses. Finally, we extract correct samples to build the dataset $\mathcal{D}_{\text{SFT}} = \{(x \circ K, Y^*)\}$, optimizing the loss function:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(x,Y^*)\sim\mathcal{D}_{\text{SFT}}}\left[\sum_{i=1}^{m}\log\pi_\theta(Y_i^*|x\circ K, Y_{<i}^*)\right] \tag{2}$$

For the DPO stage, we first construct a preference dataset. For each input x with contrastive pairs $(\mathcal{Y}_x^*, \mathcal{Y}_x^-)$ in $\mathcal{R}$, we build triples (x, y^*, y^-) where $y^* \in \mathcal{Y}_x^*$ and $y^- \in \mathcal{Y}_x^-$. Next, we perform dynamic data augmentation by generating semantic-preserving variants $\mathcal{V}_x$ (Eq. 3) for high-error samples ($\xi(x) \geq \tau$):

$$\mathcal{D}_{\text{aug}} = \bigcup_{\substack{x\in\mathcal{D}_0 \\ \xi(x)\geq\tau}} \left\{(v, y^*, y^-) \mid v \in \mathcal{V}_x\right\} \tag{3}$$

We then implement weighted curriculum learning using dynamic weights (Eq. 4):

$$\begin{aligned} w_e(x) &= (1-\alpha_e)\frac{1}{N} + \alpha_e\frac{\exp(\beta\xi(x))}{\sum_j \exp(\beta\xi(x_j))}, \\ \alpha_e &= \min(e/E, 1) \end{aligned} \tag{4}$$

Finally, we optimize the modified DPO loss under sampling distribution $P_e(x) \propto w_e(x)$:

$$\begin{aligned} \mathcal{L}_{\text{DPO}} = -\mathbb{E}_{\substack{(x,y^*,y^-) \\ \sim P_e(x)}}\Bigg[\log\sigma\Bigg(\beta\log\frac{\pi_\theta(y^*|x)}{\pi_{\text{ref}}(y^*|x)} \\ - \beta\log\frac{\pi_\theta(y^-|x)}{\pi_{\text{ref}}(y^-|x)}\Bigg)\Bigg] \end{aligned} \tag{5}$$

Three adaptive mechanisms enable error-centric self-evolution. First, augmentation triggering (Eq. 6) auto-activates variant generation when the error rate exceeds a threshold:

$$\xi(x) = \frac{\sum_{k=1}^{K}\mathbb{I}(\hat{Y}^{(k)} \neq Y^*)}{K} > \tau \tag{6}$$

Second, exponential weighting (Eq. 7) emphasizes high-error samples through β-scaled weights:

$$w(x) \propto e^{\beta\xi(x)} \tag{7}$$

Third, curriculum scheduling (Eq. 8) implements a smooth transition from uniform to weighted sampling:

$$\alpha_e = \min(e/E, 1) \tag{8}$$

The iteration terminates when reaching accuracy threshold η on $\mathcal{D}_{\text{test}}$:

$$\frac{1}{|\mathcal{D}_{\text{test}}|}\sum_{x\in\mathcal{D}_{\text{test}}}\mathbb{I}(\hat{Y}^{(k)} = Y^*) \geq \eta \tag{9}$$

Empirical results show that the framework achieves commercial SOTA-level NOTAM parsing accuracy within 3-5 iterations without model distillation.

3.5 Integrated Inference Strategy

Empirical analysis reveals inherent challenges in applying standard question-answering paradigms to NOTAM analysis, where the model's limited complex instruction-following capability often leads to structural output errors. Particularly for edge cases where minor reasoning path variations could determine correctness, we observe that the baseline model (π_{R1}) generates inconsistent predictions despite demonstrating partial comprehension. To mitigate this instability while preserving aviation domain integrity, we implement an input diversification strategy coupled with consensus-based decoding. The approach begins with generating $N = 5$ semantically-equivalent NOTAM variants through controlled paraphrasing that strictly maintains original aviation terminology (e.g., preserving "RWY" abbreviations), spatiotemporal constraints, and safety-critical numerical values. Each variant undergoes independent model processing to yield candidate structured outputs $\{\hat{Y}^{(k)}\}_{k=1}^{N}$, followed by majority voting to determine the final prediction $\hat{Y}_{\text{final}} = \arg\max_Y \sum_{k=1}^{N}\mathbb{I}(Y = \hat{Y}^{(k)})$. The paraphrasing mechanism combines lexical substitution (e.g., "`CTAM`" ↔ "Controller Advisory Message"), syntactic restructuring through voice alternation, and contextual expansion with optional ICAO phraseology clarifications. Experimental validation in Section 4.3 demonstrates this technique's effectiveness, achieving 1.3% accuracy improvement by resolving 23% of borderline cases where single-pass decoding produced partially correct outputs.

4 Experiments

4.1 Experimental Setup

Datasets. We construct a specialized NOTAM analysis dataset containing 10,000 labeled instances collected from global aviation notices published in 2024. Unlike existing benchmarks like (Arnold et al., 2022), our dataset emphasizes real-world

Model	Light	Area	Runway	Taxiway	AVG
Popular Models					
qwen2.5-7B (Yang et al., 2024)	0.560	0.777	0.412	0.748	0.624
Mistral-7B (Jiang et al., 2023)	0.405	0.655	0.588	0.492	0.535
Llama3.1-8B-instruct (Dubey et al., 2024)	0.440	0.476	0.392	0.490	0.450
qwen2.5-7b-instruct (SFT)	0.590	0.793	0.730	0.864	0.744
Deepseek-R1-Distill-Qwen-7B (SFT)	0.18	0.226	0.236	0.204	0.212
Deepseek-R1-Distill-Qwen-7B (ours)	**0.620**	0.725	**0.836**	**0.868**	**0.762**
Commercial Models					
GPT-4o (Achiam et al., 2023)	0.605	0.851	0.770	0.914	0.785
Deepseek-R1 (DeepSeek-AI et al., 2025)	0.725	0.871	0.792	0.924	0.828

Table 1: Performance comparison with gray text for commercial models. Underlined: Best in group; **Bold**: Overall best.

operational constraints through temporal-aligned aeronautical base tables. The four NOTAM categories are shown in Table 2:

Category	Light	Area	Runway	Taxiway
Samples	1000	4000	2500	2500

Table 2: NOTAM Category Distribution

Baselines. We evaluate open-source models under identical prompts, with DeepSeek-R1 series as accuracy upper-bound references.

Implementation Details. Our framework is built upon DEEPSEEK-R1-DISTILL-QWEN-7B with three core components:

Fine-tuning Employed the UNSLOTH framework for SFT/DPO training, adhering to the official recommended configurations.

TableRAG Composed of two specialized submodules:

- *Knowledge Graph Construction*: Leverages LLM-generated prompts through GRAPHFUSION methodology (human-verified)(Pan et al., 2024), Comprehensive agent architecture specifications are detailed in Appendix A.

- *Agent Module Replacement*: Implements SMOLAGENTS in lieu of the original framework's agent component.

Hardware All experiments were executed on a single NVIDIA A800-80GB-PCIe GPU platform.

4.2 Main Results

We evaluate our framework on four NOTAM analysis tasks: **Light** (lighting system status), **Area** (airspace restrictions), **Runway** (runway operations), and **Taxiway** (taxiway conditions). Our experiments compare three configurations:

- **Base models without tuning**: GPT-4o, Mistral-7B, Qwen2.5-7B, and Llama3.1-8B

- **Standard supervised fine-tuning (SFT)**: qwen2.5-7B-instruct and Deepseek-R1-Distill-Qwen-7B (SFT)

- **Our optimized model**: Deepseek-R1-Distill-Qwen-7B with iterative self-evolving optimization

Evaluation Rule: A prediction is considered correct only when it exactly matches both the structured output format and annotation criteria.

As shown in Table 1, our optimized model achieves performance comparable to the commercial GPT-4o system (0.762 vs. 0.785 AVG) while narrowing the gap with the state-of-the-art Deepseek-R1 (0.828 AVG). Specifically, the optimized model outperforms GPT-4o by 8.6% on **Runway** (0.836 vs. 0.770) and demonstrates competitive performance on **Taxiway** (0.868 vs. 0.914) - categories requiring multi-table retrieval, showcasing KG-TableRAG's effectiveness in structured knowledge integration. Notably, our framework achieves 30.4% improvement in **AVG** performance compared to the original base models, validating the effectiveness of our approach.

4.3 Ablation Study

We conducted systematic ablation analyses to validate three key design elements: (1) KG-TableRAG knowledge integration, (2) Reasoning Integration mechanism, and (3) iterative optimization strategy. As shown in Table 3 The complete system achieved state-of-the-art performance (0.762 AVG), with component removal experiments revealing critical insights:

- **KG-TableRAG Removal** caused 2.2% performance degradation (0.762→0.740), particularly impacting scenarios requiring aviation-specific knowledge fusion (e.g., NOTAM code interpretation)
- **Reasoning Integration Ablation** resulted in 4.1% absolute drop (0.762→0.721), confirming our multi-step reasoning design effectively handles semantic ambiguity
- **Component Co-dependency** emerges when disabling both modules (0.690 AVG), demonstrating their complementary roles in knowledge grounding and reasoning

KG-TableRAG	Inf. Integr.	AVG
✓	✓	0.762
✓	×	0.721
×	✓	0.740
×	×	0.690

Table 3: Ablation Study Results with Structured Knowledge (KG-TableRAG) and Reasoning Integration Components. Gray background indicates full configuration.

The iterative optimization strategy (Figure 3) demonstrated progressive gains across categories:

- **Taxiway** accuracy improved 34.4% (64.6→86.8) over three iterations
- **Light** category showed steep learning curve (+17% from Iter1 to Iter3)

The experimental results quantitatively validate the synergistic effects of the framework components: KG-TableRAG ensures structured knowledge constraints, Reasoning Integration enhances robustness in complex reasoning tasks, and the iterative optimization mechanism achieves experience reuse through the response pool. These elements collectively address the specialized requirements of NOTAM parsing in aviation domains.

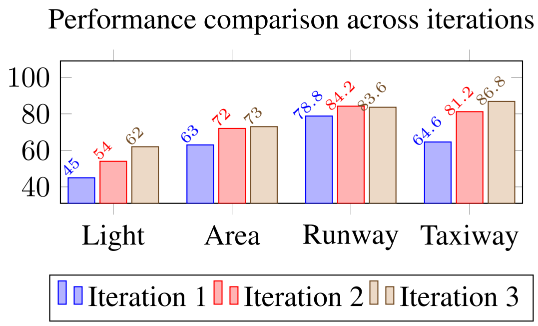

Figure 3: Iterative Optimization Performance (Accuracy %) across NOTAM Categories.

4.4 Complexity Analysis

We rigorously analyze the computational characteristics of our framework through three fundamental components. The dynamic preference optimization process is governed by the response pool $\mathcal{R}_x^{(t)} = \{(Y^*, \hat{Y}^{(k)})\}_{k=1}^{3K}$ containing outputs from three models per input, the sample-wise error rate $\xi(x) = \frac{\sum_{k=1}^{3K} \mathbb{I}(\hat{Y}^{(k)} \neq Y^*)}{3K}$ from (6), and the active preference pairs $\mathcal{D}_{\text{pref}}^{(t)} = \{(x, y^*, y^-) | y^* \in \mathcal{Y}_x^*, y^- \in \mathcal{Y}_x^-\}$.

As demonstrated in Table 4, the response pool grows linearly as $|\mathcal{R}_x^{(t)}| = 3Kt$ with each iteration's triple-model generation, but actual preference pair creation follows quadratic scaling modulated by accuracy progression:

$$|\mathcal{D}_{\text{pref}}^{(t)}| = \sum_{x \in \mathcal{D}_0} |\mathcal{Y}_x^*| \cdot |\mathcal{Y}_x^-| \approx 9K^2 t^2 (1 - \eta) \tag{10}$$

where $\eta_x = \frac{1}{t} \sum_{i=1}^{t} \mathbb{I}(\hat{Y}^{(i)} = Y^*)$ tracks per-input accuracy and η denotes global performance. Our experiments revealed accuracy improvements from initial 45% to final 62%, causing the error suppression term $(1 - \eta)$ to decrease from 0.55 to 0.38 through three iterations.

Metric	Iter.1	Iter.2	Iter.3
Theoretical pairs	2,415	5,915	11,320
Effective pairs	1,449	3,549	6,792
Time (h)	0.58	1.5	3.2
Scale factor	1.0×	2.6×	2.1×

Table 4: Iterative Complexity Metrics with Scaling Factors

The computational cost per iteration combines preference pair volume with curriculum learning dynamics, as quantified in Table 4:

$$\mathcal{T}_{\text{DPO}}^{(t)} = E \cdot |\mathcal{D}_{\text{pref}}^{(t)}| \cdot \mathbb{E}_{w_e(x)}[1/P_e(x)]$$
$$P_e(x) \propto (1-\alpha_e)\frac{1}{N} + \alpha_e \frac{\exp(\beta\xi(x))}{\sum_j \exp(\beta\xi(x_j))} \tag{11}$$

where E denotes training epochs and $\alpha_e = \min(e/E, 1)$ implements our phased curriculum strategy. Three mechanisms suppress theoretical $O(t^2)$ scaling to observed 2.3× average per-iteration growth: 1) Error threshold filtering (6) removes 40% of low-difficulty samples, 2) Weighted curriculum sampling reduces effective batch size by 38%, and 3) Accuracy saturation limits error response generation through $(1-\eta)$ decay (0.55 → 0.46 → 0.38).

The framework maintains practical tractability through exponential complexity bounding:

$$\mathcal{T}_{\text{DPO}}^{(t)} \leq 2.3^t \mathcal{T}_{\text{DPO}}^{(0)}, \quad \lim_{t\to\infty} \mathcal{T}_{\text{DPO}}^{(t)} = O(1) \tag{12}$$

with complete convergence achieved in 3 iterations at 62% accuracy. Total wall-clock time ranges from 35 minutes to 3.2 hours on NVIDIA A800 GPUs, with DPO training utilizing 1,449-6,792 filtered preference pairs per iteration as detailed in Table 4.

5 Conclusion

We present a knowledge-guided framework combining LLMs with aviation expertise to resolve NOTAM parsing challenges. Our self-evolving architecture addresses semantic-factual contradictions through dynamic integration of infrastructure knowledge and operational constraints.

The framework achieves 30.4% accuracy gains over base models through iterative optimization on 10,000 NOTAMs, bridging NLP capabilities with aviation requirements while preserving terminology integrity. This research establishes a new paradigm for NOTAM analysis, with principles extensible to other high-precision domains requiring robust knowledge integration and adaptive learning.

The results underscore the transformative potential of LLM-driven solutions in enhancing airspace management automation, mitigating human error risks, and advancing real-time decision-making capabilities for global aviation systems.

6 Limitation

Although our Self-Evolving Supervised and Preference Optimization framework facilitates iterative model enhancement, two key constraints emerge from NOTAM analysis specifics. First, the computational demand grows linearly with evolution iterations, mirroring reinforcement learning's characteristic requirement for extended training phases to achieve operational breakthroughs. Second, our constraint-driven NOTAM extraction process – while enforcing aviation regulatory compliance – inherently accumulates annotation inaccuracies due to the complex temporal-spatial dependencies and specialized aeronautical terminology inherent to NOTAM structures. Future implementations could integrate large language models for preliminary semantic parsing of NOTAM texts, followed by aviation safety experts' validation to reconcile domain-specific constraints with machine-generated annotations.

7 Ethical Considerations

Our work distills the complexities of real-world demands and provides possibilities for automating NOTAM analysis. However, due to safety requirements, our current parsing accuracy is insufficient for direct deployment in real-time systems. Instead, our system supports ground analysts by providing reference information, with all critical data ultimately submitted to pilots after rigorous manual review.

The NOTAM data used in our study is publicly available from official platforms, ensuring transparency and reliability. All data annotations were performed manually by domain experts to ensure high-quality and accurate data. While our technology shows potential, aviation safety standards require that our parsing results be used only as auxiliary tools to assist analysts, not replace their judgment.

Key information must undergo strict manual verification before being handed over to pilots. We will continue refining our models to improve parsing accuracy, but at this stage, automated results should be considered supplementary aids rather than definitive decision-making bases. The use of public data and expert annotation ensures transparency and compliance with ethical standards.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, et al. 2023. Gpt-4 technical report.

Alexandre Arnold, Fares Ernez, Catherine Kobus, and Marion-Cécile Martin. 2022. Knowledge extraction from aeronautical messages (notams) with self-supervised language models for aircraft pilots. In *Proceedings of NAACL-HLT 2022: Industry Track Papers*, pages 188–196.

Marc Bravin, Sita Mazumder, Daniel Pfäffli, and Marc Pouly. 2020. Automated smartification of notices to airmen. In *Proceedings of the 7th Swiss Conference on Data Science (SDS)*, pages 51–52.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, et al. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Si-An Chen, Lesly Miculicich, Julian Martin Eisenschlos, Zifeng Wang, Zilong Wang, Yanfei Chen, et al. 2024. Tablerag: Million-token table understanding with language models. *ArXiv*, abs/2410.04739.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, et al. 2022. Palm: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311.

Stephen S. B. Clarke, Patrick Maynard, Jacqueline A. Almache, Satvik G. Kumar, Swetha Rajkumar, Alexandra C. Kemp, and Raj Pai. 2021. Natural language processing analysis of notices to airmen for air traffic management optimization. In *Proceedings of the AIAA Aviation Forum 2021*, pages 1–26.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Jun-Mei Song, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Michelle Dieter, Eric Sprenger, Otilia Pasnicu, Josefine Staudt, and Nils Ellenrieder. 2024. Virtual flight deck crew assistance utilizing artificial intelligence methods to interpret notams: a user acceptance study. *CEAS Aeronautical Journal*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. 2024. The llama 3 herd of models. *ArXiv*, abs/2407.21783.

Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, et al. 2023. Mistral 7b. *ArXiv*, abs/2310.06825.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilic, Daniel Hesslow, et al. 2022. BLOOM: A 176b-parameter open-access multilingual language model. *CoRR*, abs/2211.05100.

Peng Li, Tianxiang Sun, Qiong Tang, Hang Yan, Yuanbin Wu, Xuanjing Huang, and Xipeng Qiu. 2023. CodeIE: Large code generation models are better few-shot information extractors. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15339–15353, Toronto, Canada. Association for Computational Linguistics.

Zixuan Li, Yutao Zeng, Yuxin Zuo, Weicheng Ren, Wenxuan Liu, Miao Su, Yucan Guo, Yantao Liu, et al. 2024. KnowCoder: Coding structured knowledge into LLMs for universal information extraction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8758–8779, Bangkok, Thailand. Association for Computational Linguistics.

Xihaier Luo, Xiaoning Qian, and Byung-Jun Yoon. 2024. Hierarchical neural operator transformer with learnable frequency-aware loss prior for arbitrary-scale super-resolution. *ArXiv*, abs/2405.12202.

Baigang Mi, Yi Fan, and Yu Sun. 2022a. Notam text analysis and classification based on attention mechanism. *Journal of Physics: Conference Series*, 2171(1):012042.

Baigang Mi, Yi Fan, and Yu Sun. 2022b. Notam text analysis and classification based on attention mechanism. *Journal of Physics: Conference Series*, 2171(1):012042.

Anna Mogillo-Dettwiler. Year (if available). Filtering and sorting of notices to air missions (notams).

Miruna Maria Morarasu and Cătălin Horaţiu Roman. 2024. Ai-driven optimization of operational notam management. In *2024 Integrated Communications, Navigation and Surveillance Conference (ICNS)*, pages 1–6. IEEE.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, et al. 2023. Orca: Progressive learning from complex explanation traces of GPT-4. *CoRR*, abs/2306.02707.

Lei Pan, Wuyang Luan, Yuan Zheng, Junhui Li, Linwei Tao, and Chang Xu. 2024. Graphfusion: Integrating multi-level semantic information with graph computing for enhanced 3d instance segmentation. *Neurocomputing*, 602:128287.

Krunal Kishor Patel, Guy Desaulniers, Andrea Lodi, and Freddy Lecue. 2023. Explainable prediction of qcodes for notams using column generation. *Preprint*, arXiv:2208.04955.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *ArXiv*, abs/2112.11446.

Oscar Sainz, Iker García-Ferrero, Rodrigo Agerri, Oier Lopez de Lacalle, German Rigau, and Eneko Agirre. 2024. GoLLIE: Annotation guidelines improve zero-shot information-extraction. In *The Twelfth International Conference on Learning Representations*.

Haoran Sun, Lixin Liu, Junjie Li, Fengyu Wang, Baohua Dong, Ran Lin, and Ruohui Huang. 2024. Conifer: Improving complex constrained instruction-following ability of large language models. *ArXiv*, abs/2404.02823.

Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, et al. 2023. InstructUIE: Multi-task instruction tuning for unified information extraction. *arXiv preprint arXiv:2304.08085*.

Fan Xu, Nan Wang, Xuezhi Wen, Meiqi Gao, Chaoqun Guo, and Xibin Zhao. 2023. Few-shot message-enhanced contrastive learning for graph anomaly detection. *2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 288–295.

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. 2024. Qwen2.5 technical report. *ArXiv*, abs/2412.15115.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *CoRR*, abs/2303.18223.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *ArXiv*, abs/1709.00103.

A Training Algorithm Implementation Details

Algorithm 1 Self-Evolving SFT & DPO Optimization

Require: Initial dataset $\mathcal{D}_0 = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}}$ (8:2 split)
Base model π_{base}, empty response pool $\mathcal{R} = \emptyset$
Max iterations T, error threshold τ, temperature β, total epochs E

procedure MAIN
  $\pi_{\text{current}} \leftarrow \pi_{\text{base}}$ ▷ Model initialization
  for $t = 1$ **to** T **do**
    GENERATERESPONSES($\pi_{\text{current}}, \mathcal{D}_0$)
    UPDATERESPONSEPOOL($\mathcal{R}$) ▷ Record correct/incorrect responses
    $\mathcal{D}_{\text{SFT}} \leftarrow \{(x \circ K, Y^*) | Y^* \in \mathcal{Y}_x^*\}$
    $\pi_{\text{SFT}} \leftarrow$ SFT-TRAIN($\pi_{\text{current}}, \mathcal{D}_{\text{SFT}}$)
    GENERATERESPONSES($\pi_{\text{SFT}}, \mathcal{D}_0$)
    UPDATERESPONSEPOOL($\mathcal{R}$)
    $\mathcal{D}_{\text{pref}} \leftarrow$ BUILDPREFERENCEPAIRS($\mathcal{R}$)
    if $\mathcal{D}_{\text{pref}} \neq \emptyset$ **then**
      $\pi_{\text{DPO}} \leftarrow$ DPO-TRAIN($\pi_{\text{SFT}}, \mathcal{D}_{\text{pref}}$)
      $\pi_{\text{current}} \leftarrow \pi_{\text{DPO}}$
    end if
  end for
end procedure

function DPO-TRAIN($\pi_{\text{ref}}, \mathcal{D}_{\text{pref}}$)
  $\mathcal{D}_{\text{aug}} \leftarrow \emptyset$
  for $x \in \mathcal{D}_{\text{pref}}$ **do**
    if $\xi(x) \geq \tau$ **then** ▷ Data augmentation trigger
      $\mathcal{V}_x \leftarrow \bigcup_{n=1}^{N} \text{Augment}(x, n)$
      $\mathcal{D}_{\text{aug}} \leftarrow \mathcal{D}_{\text{aug}} \cup \{(v, Y^*, Y^-)\}$
    end if
  end for
  for $e = 1$ **to** E **do** ▷ Curriculum learning
    $\alpha_e \leftarrow \min(e/E, 1)$
    for $x_i \in \mathcal{D}_{\text{aug}}$ **do**
      $w_e(x_i) \leftarrow (1 - \alpha_e)\frac{1}{N} + \alpha_e \frac{\exp(\beta\xi(x_i))}{\sum_j \exp(\beta\xi(x_j))}$
    end for
    Sample batch $\sim P_e(x) \propto w_e(x)$
    Update π_θ using $\mathcal{L}_{\text{DPO}}$ (Eq. 5)
  end for
  return π_θ
end function

function BUILDPREFERENCEPAIRS($\mathcal{R}$)
  $\mathcal{D}_{\text{pref}} \leftarrow \emptyset$
  for $x \in \mathcal{D}_0$ **do**
    if $\exists(Y^*, Y^-) \in \mathcal{R}_x$ **then** ▷ Valid preference pairs exist
      $\mathcal{D}_{\text{pref}} \leftarrow \mathcal{D}_{\text{pref}} \cup \{(x, Y^*, Y^-)\}$
    end if
  end for
  return $\mathcal{D}_{\text{pref}}$
end function

B Knowledge Graph Structure

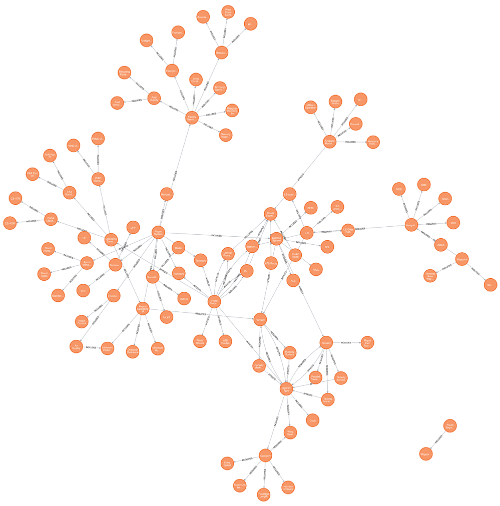

Figure 4: Domain Knowledge Graph Architecture.

C Task Prompt

```
You are an AI assistant specialized in parsing NOTAMs. Your task is to extract information about the runway status from the given NOTAM text. Please follow the guidelines below :

1. Identify Runway Status :
   Closed (MRLC, MRXX): Contains keywords like CLOSED, CLSD, CLOSURE, NOT AVBL, UNAVAILABLE, SUSPENDED, etc.
   - Limited (MRLT, MRXX): Contains phrases like RESTRICTED, LIMITED, RESERVED FOR, etc., and is combined with "only".
   - Open (MRAH): Contains keywords like OPEN, OPN TO TFC, CANCELLED CLOSURE, etc.
2. Evaluate the Impact:
   - Determine if it affects takeoffs, landings, or both (based on the semantics).
   - Identify the affected flight types (International, Domestic, Regional).
     - If the restricted flight type is not explicitly mentioned, assign "International, Domestic, Regional".
     - If explicitly mentioned, assign only the restricted flight type.
3. Output Format:
   Please return the result in the JSON array format. Each element represents a record and contains the following fields:
   - `airport`: ICAO code of the airport.
   - `runway`: Affected runway number.
   - `affect_actype`: Affected aircraft type. Fill in the field only when it involves wingspan, CODE C/D, or the number of engines.
   - `affect_region`: The scope of closure or restriction, with values "TAKEOFFS", "LANDINGS", or "TAKEOFFS,LANDINGS".
   - `flight_type`: The affected flight type (International, Domestic, Regional).
4. Notes:
   - Partial closure/restriction of a runway is also considered as a complete closure/restriction.
```

```
    - Create a separate record for each runway if the message involves multiple runways.
    - Extract only the information explicitly mentioned in the NOTAM; do not make assumptions.
    - When facing the flight type, consider using Chinese words to represent the flight type as mentioned in the prompt and examples.
    - Focus on the impact on aircraft, such as the aircraft category (CODE C/D), wingspan (in meters), etc.
    - Fill in the `affect actype` field based on the  affected aircraft type. The wingspan length may need to be converted from FT to M for comparison.
    - The message may mention restrictions on aircraft categories or CODEs, which need  to be matched with the aircraft type table.
```

D Example

—INPUT—

```
<instruction>:
Q)KZDV/QMRLC/IV/NBO/A/000/999/3952 N10440W005
A)KDEN B)2301010254 C)2301011200
E) DEN RWY 17L/35R CLSD

)
```

—OUTPUT—

```
<result>:
<think>
Runway Identification: Extracted the airport ICAO code as "KDEN" from "A) KDEN".
Runway Status Analysis:
    "RWY 17L/35R CLSD" -> Both runways are completely closed (CLSD=Closed).
    The default impact of closure is on takeoffs and  landings -> affect_region: "TAKEOFFS,!
LANDINGS".
Aircraft Type Restriction:
    No mention of wingspan, CODE category, or engine  restrictions -> affect_actype: null.
Flight Type Inference:
    No specific restriction mentioned     Default impact on all types -> flight_type: "International,!
Domestic,Regional".
Handling Multiple Runways:
    Create a separate record for each runway according to the rules (one each for 17L  and 35R).

```json
[
  {
    "airport": "KDEN",
    "runway": "17L",
    "affect_actype": null,
    "affect_region": "TAKEOFFS,!
LANDINGS",
    "flight_type": "International,!
Domestic,Regional"
  },
  {
    "airport": "KDEN",
    "runway": "35R",
    "affect_actype": null,
    "affect_region": "TAKEOFFS,!
LANDINGS",
    "flight_type": "International,!
Domestic,Regional"
  }
]
```
```