DOT/FAA/TC-26/14

Federal Aviation Administration
William J. Hughes Technical Center
Aviation Research Division
Atlantic City International Airport
New Jersey 08405

Artificial Intelligence (AI) for Integrated Safety Assessment Model (ISAM)

June 2026

U.S. Department of Transportation
Federal Aviation Administration

NOTICE

This report is available at the Federal Aviation Administration William J. Hughes Technical Center's Full-Text Technical Reports page: actlibrary.tc.faa.gov in Adobe Acrobat portable document format (PDF).

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT/FAA/TC-26/14		

4. Title and Subtitle	5. Report Date
Title of Report: Artificial Intelligence (AI) for Integrated Safety Assessment Model (ISAM)	June 2026
	6. Performing Organization Code ANG-E272
7. Author(s) Jirat Bhanpato, Chetan Chandra, Xiao Jing, Kshitij Sawant, Timilehin Oderinde, Barbara Sampaio Felix, Alexia Payan, Michelle Kirby, Dimitri Mavris	8. Performing Organization Report No.
9. Performing Organization Name and Address Aerospace Systems Design Laboratory The Daniel Guggenheim School of Aerospace Engineering Georgia Institute of Technology 620 Cherry Street, Atlanta, GA 30332	10. Work Unit No. (TRAIS)
	11. Contract or Grant No. A 692M15-23-F-00178
12. Sponsoring Agency Name and Address FAA Office of Accident Investigation and Prevention Safety Data and Analytics Division, AVP-200 800 Independence Ave SW Washington, DC 20591	13. Type of Report and Period Covered
	14. Sponsoring Agency Code AVP-200

15. Supplementary Notes
Research Sponsor – Shane Bertish and Dereck Wilson, AVP-200
Research Project Technical Monitor – Dr. Huasheng Li, ANG-E272

16. Abstract

This project aims to advance risk-informed decision-making at the FAA by developing machine learning techniques for extracting safety-relevant information from incidents and accident reports and enhancing large language model (LLM) capabilities for navigating aviation regulatory text. The work addresses two primary challenges: the labor-intensive process of manually extracting safety data from Aviation Safety Reporting System (ASRS) and National Transportation Safety Board (NTSB) narratives for use in the Integrated Safety Assessment Model (ISAM), and the inability of general-purpose language models to accurately interpret the hierarchical structure and cross-references within Title 14 of the Code of Federal Regulations (CFR). To improve safety data extraction, the project developed two datasets. The first is a question answering (QA) dataset designed to enable precise supervision for fine-tuning LLMs to extract events, contributing factors, and operational conditions from narrative text. The second consists of synthetic safety reports generated using a fine-tuned LLM informed by a domain-specific knowledge graph (KG) built from FAA aircraft data to eliminate aircraft specification errors and improve multilabel classification performance. To enhance regulatory reasoning, the project built a regulation-aware knowledge graph and developed a retrieval-augmented generation (RAG) system with improved retrieval precision for semi-structured regulatory content. An evaluation framework was also implemented by adapting retrieval-augmented generation assessment (RAGAS) metrics for regulatory contexts and introducing a custom answer completeness metric to address the “missing zone problem” of incomplete retrieval. Finally, the FAA-provided Daedalus multi-agent RAG system was migrated to local infrastructure using the Model Context Protocol (MCP) and extended with a completeness-evaluation agent that provides actionable diagnostics by distinguishing between retrieval and generation incompleteness.

17. Key Words	18. Distribution Statement
Aviation Safety, Regulation, Knowledge Graph, Artificial Intelligence, Machine Learning, Large Language Model, Integrated Safety Assessment Model (ISAM)	This document is available to the U.S. public through the National Technical Information Service (NTIS), Springfield, Virginia 22161. This document is also available from the Federal Aviation Administration William J. Hughes Technical Center at actlibrary.tc.faa.gov.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	134	

Contents

Figures

Tables

Acronyms

Acronym	Definition
API	Application Programming Interface
ASC	Aircraft Specification Consistency
ASRS	Aviation Safety Reporting System
AUC	Area Under Curve
BART	Bidirectional and Auto-Regressive Transformer
BERT	Bidirectional Encoder Representations from Transformers
CAROL	Case Analysis and Reporting Online
ESD	Event Sequence Diagram
FAA	Federal Aviation Administration
FT	Fault Tree
GRD	Ground
GT	Georgia Tech / Georgia Institute of Technology
ICAO	International Civil Aviation Organization
IFLT	In Flight
ISAM	Integrated Safety Assessment Model
KG	Knowledge Graph
LLM	Large Language Model
ML	Machine Learning
MLM	Masked Language Modeling
NAS	National Airspace System
NER	Named Entity Recognition
NLI	Natural Language Inference
NLP	Natural Language Processing
NLU	Natural Language Understanding
NSP	Next Sentence Prediction
NTSB	National Transportation Safety Board
QA	Question Answering
RAG	Retrieval-Augmented Generation
RAGAS	Retrieval-Augmented Generation Assessment
RDF	Resource Description Framework
ROC	Receiver Operating Characteristic

TTR	Type-Token Ratio
UNK	Unknown
WWM	Whole Word Masking
URI	Uniform Resource Identifier

Executive summary

This project advances the Federal Aviation Administration's (FAA's) ability to perform risk-informed decision-making by developing new datasets, analytical tools, and system-level integrations that improve the extraction of information from narrative reports and the interpretation of complex regulatory text. The work combines natural language processing (NLP), structured knowledge representation, and multi-agent retrieval-augmented generation (RAG) to create a more reliable, transparent, and scalable foundation for AI-enabled aviation safety analysis.

A key objective of the project was to enhance the FAA's capacity to leverage unstructured incident and accident narratives from the Aviation Safety Reporting System (ASRS) and the National Transportation Safety Board (NTSB). To support this goal, the team developed two major datasets. The first is a curated question answering dataset containing 244 quality-controlled question–answer pairs designed to enable precise supervision for fine-tuning large language models (LLMs). This dataset improves the ability of automated systems to extract events, contributing factors, operational conditions, and other safety-relevant details directly from narrative text.

The second dataset consists of synthetic safety reports generated using a fine-tuned Mistral-7B model informed by a domain-specific knowledge graph built from FAA aircraft data. This approach addresses limitations in synthetic data generation with respect to hallucination of aircraft specifications. The synthetic reports were shown to support improved multilabel classification of NTSB sequences of events. The dataset eliminated specification errors (Aircraft Specification Consistency (ASC) score of 1.000) and improved downstream classification performance from a macro F1-score of 0.76 to 0.83, demonstrating the value of knowledge-constrained generation in safety-critical applications.

In addition to narrative analysis, the project advanced regulatory reasoning capabilities by developing a regulation structure-aware knowledge graph (KG) aligned with Title 14 of the Code of Federal Regulations (CFR). The KG encodes hierarchical relationships, cross-references, and canonical identifiers, enabling more accurate retrieval and citation of regulatory requirements. This structured representation serves as a foundation for LLM-enabled analysis that is verifiable.

These capabilities were integrated into the Daedalus multi-agent RAG system. The integration included migrating the Daedalus codebase to local infrastructure, enabling multi-source retrieval through the KG, and implementing a novel completeness-evaluation agent that distinguishes between retrieval and generation incompleteness. This capacity provides actionable diagnostics

for improving answer quality and system reliability, moving the system closer to operational viability.

The project's contributions significantly improve the technical resources available for AI-driven aviation safety analysis. They provide the foundation for future work, including expanding the regulatory knowledge graph to encompass advisory circulars and other FAA documents and further evaluating KG-augmented retrieval performance. This report details both completed areas of work and highlights preliminary progress on others.

1 Introduction

The objective of this project is to use information from aviation incident and accident reports, together with regulatory text, to enhance risk-informed decision-making at the Federal Aviation Administration (FAA) and improve aviation safety. To achieve this goal, the project pursued several complementary efforts.

The first set of activities focused on developing machine learning methods capable of extracting safety-relevant facts from incident and accident narratives. Although these reports contain essential details for understanding how safety-critical events unfold, their unstructured format makes large-scale fact extraction challenging. By leveraging information from the Aviation Safety Reporting System (ASRS) and National Transportation Safety Board (NTSB) reports, the project aims to support more informed and data-driven safety decisions.

A second set of activities focused on enabling accurate representation and querying of regulatory text. While regulations are more structured than reports (organized into parts, chapters, and sections, etc.) general-purpose large language models (LLMs) struggle to navigate these documents in a way that respects their hierarchical organization and cross-referenced relationships.

This report details the project's contributions across these two areas of work. The first effort, described in Section 2.1, involved creating a dedicated question answering (QA) dataset that allows automated models to identify key facts directly from narrative text and retrieve meaningful safety information quickly and reliably. The second effort developed synthetic incident and accident reports using a fine-tuned Mistral-7B model informed by a domain-specific knowledge graph constructed from FAA aircraft data. The generation of synthetic data addresses the severe class imbalance in aviation safety datasets. Our approach, described in Section 2.2, unlike conventional data-augmentation methods, was able to capture the domain's complex technical dependencies resulting in technically accurate and contextually consistent synthetic reports.

The project also explored the use of retrieval-augmented generation (RAG) combined with a regulation-structure-aware knowledge graph (KG) to improve an LLM's ability to navigate Title 14 of the Code of Federal Regulations (CFR) and understand relationships among regulatory provisions. The resulting KG reflects the CFR hierarchy, preserves canonical section identifiers, and represents cross-references as explicit, traversable links. This structured representation is intended to serve as the foundation for LLM-enabled retrieval, enabling models to retrieve, cite, and explain regulatory requirements directly from the graph rather than relying solely on surface-

level pattern matching over raw text. By grounding model behavior in an explicit regulatory structure, the system aims to support precise retrieval, transparent citations, and consistent interpretation of rules across operations and certification, while reducing the risk of hallucinated or unverifiable responses. The technical details of the KG-RAG system, the KG construction process, and the metrics used to evaluate their performance are provided in Sections 2.3 and 2.4.

Additional work, described in Section 2.5, was undertaken to integrate the project's advancements into the Daedalus multi-agent RAG system. This included migrating the FAA-provided Daedalus codebase to local infrastructure, adapting the system to support knowledge-graph-based retrieval and multi-source integration, and developing a novel completeness-evaluation agent capable of distinguishing between retrieval incompleteness and generation incompleteness. This diagnostic capability provides actionable insights that improve both answer quality and system design. Together, these contributions position the project's methods and tools for incorporation into a production-ready platform suitable for government and enterprise deployment.

Overall, the efforts described in this report lay the groundwork for a more reliable, transparent, and scalable approach to leveraging narrative safety data and regulatory text within advanced AI systems. By combining improved information extraction, structured regulatory reasoning, and robust multi-agent evaluation, the project advances the state of the art in risk-informed decision support and sets the stage for future operational integration within the FAA.

1.1 Summary of previous work

Previous efforts focused on automating the extraction and structuring of aviation safety information from accident and incident narratives for integration into the FAA's Integrated Safety Assessment Model (ISAM). These efforts aimed to assist the manual process of reading NTSB and ASRS narratives and translating them into ISAM's event sequence diagrams and fault trees with a pipeline built around a domain-specific language model, Aviation-BERT, and an expanded Aviation Knowledge Graph (Aviation-KG) based on the ICAO ADREP taxonomy. Together, these components were designed to serve as a "translation layer" between free-text safety reports and the formal models used for system-level safety risk analysis.

On the modeling side, Aviation-BERT was fine-tuned for several downstream tasks, including multilabel classification of occurrence categories, multilabel classification of phase of flight, and named entity recognition (NER). For occurrence categories and flight phases, datasets were constructed primarily from ASRS and harmonized NTSB metadata, and Aviation-BERT consistently outperformed a generic BERT-base model across F1, ROC-AUC, and Hamming

loss metrics, particularly for high-frequency classes. For NER, a large synthetic training corpus was created using 423 narrative-like templates populated with realistic aviation entities (airports, aircraft models, operators, quantities, dates, times, etc.), and the resulting model was validated on manually annotated NTSB narratives. The Aviation-BERT-NER model achieved weighted F1 scores of approximately 95–96%, outperforming other aviation/aerospace NER approaches and demonstrating that carefully designed synthetic data can significantly reduce manual labeling effort.

In parallel, Aviation-KG was expanded from its original 3,775 NTSB reports to more than 25,000 accidents by systematically ingesting 14 factual attributes (e.g., accident number, city, aircraft model, engine type, phase, event code) from NTSB metadata. This expansion more than doubled the number of axioms and object property assertions while preserving the original ontology structure. For narrative-driven information, the NTSB "findings" hierarchy was analyzed, and a two-layer hierarchical classification model was developed using Aviation-BERT to reconstruct sequences of events: a first-layer multi-label classifier predicted broad event families (loss of control, collision with object, loss of engine power, weather encounter, other), and a second-layer multiclass model (demonstrated for collision-with-object events) distinguished finer collision types. Aviation-BERT substantially outperformed a larger zero-shot BART NLI model, underscoring the value of domain-specific fine-tuning for specialized safety tasks.

Finally, two complementary QA efforts were introduced. The first was an Aviation-KG QA module that mapped natural-language questions to SPARQL via a hybrid Aviation-BERT+CNN model, enabling graph-based answers to queries such as "Which accidents involved in-flight bird strikes?" The second was an extractive QA model fine-tuned on a mixture of SQuAD 2.0 (Rajpurkar, Zhang, Lopyrev, & Liang, 2016) and a large aviation-specific QA dataset created through manual labeling, template-based generation, and LLM-assisted (Mistral/Llama) question generation with strict post-processing to enforce word-for-word answers. These combined efforts demonstrated that integrating Aviation-BERT, an expanded Aviation-KG, and QA models provides a strong foundation for automated pipelines capable of extracting, organizing, and querying safety facts at scale, ultimately supporting more efficient and accurate ISAM quantification and aviation safety analysis. Future work was identified to broaden datasets, enrich event hierarchies, incorporate flight-phase integration, and explore RAG-style LLM integration.

1.2 Report on project publications

1.2.1 Past publications

Below is a list of papers published from this project to date:

- Kshitij Sawant, Xiao Jing, Chetan Chandra, Timilehin P. Oderinde, Barbara Sampaio Felix, Jirat Bhanpato, Dimitri N. Mavris. "Structure-Aware Retrieval for CFR Title 14: A Knowledge Graph and LLM-Enabled Approach," AIAA 2026-4229 *AIAA AVIATION FORUM 2026*. June 2026. https://doi.org/10.2514/6.2026-4229
- Timilehin P. Oderinde, Chetan Chandra, Leslie Albertoli, Jirat Bhanpato, Mayank V. Bendarkar and Dimitri N. Mavris. "Aviation Safety QA Dataset for Extracting Knowledge From Incident Reports," AIAA 2025-3248. *AIAA AVIATION FORUM AND ASCEND 2025*. July 2025. https://doi.org/10.2514/6.2025-3248
- Xiao Jing, Jirat Bhanpato, Mayank V. Bendarkar and Dimitri N. Mavris. "KG-Enhanced Synthetic Report Generation for Addressing Class Imbalance in Aviation Safety Data," AIAA 2025-3250. *AIAA AVIATION FORUM AND ASCEND 2025*. July 2025. https://doi.org/10.2514/6.2025-3250
- Chetan Chandra, Yuga Ojima, Mayank V. Bendarkar, Dimitri N. Mavris. "Aviation-BERT-NER: Named Entity Recognition for Aviation Safety Reports," *Aerospace* 2024, *11*, 890. October 2024. https://doi.org/10.3390/aerospace11110890
- Xiao Jing, Kshitij Sawant, Mayank V. Bendarkar, Lidya R. Elias and Dimitri N. Mavris. "Expanding Aviation Knowledge Graph Using Deep Learning for Safety Analysis," AIAA 2024-4603. *AIAA AVIATION FORUM AND ASCEND 2024* . July 2024. https://doi.org/10.2514/6.2024-4603
- Chetan Chandra, Xiao Jing, Mayank V. Bendarkar, Kshitij Sawant, Lidya Elias, Michelle Kirby and Dimitri N. Mavris. "Aviation-BERT: A Preliminary Aviation-Specific Natural Language Model," AIAA 2023-3436. *AIAA AVIATION 2023 Forum*. June 2023. https://doi.org/10.2514/6.2023-3436
- Xiao Jing, Akul Chennakesavan, Chetan Chandra, Mayank V. Bendarkar, Michelle Kirby and Dimitri N. Mavris. "BERT for Aviation Text Classification," AIAA 2023-3438. *AIAA AVIATION 2023 Forum*. June 2023. https://doi.org/10.2514/6.2023-3438

1.2.2 Planned publications

The following paper from this technical report is being prepared for publication:

- Xiao Jing, Chetan Chandra, Timilehin P. Oderinde, Kshitij Sawant, Barbara Sampaio Felix, Jirat Bhanpato, Dimitri N. Mavris. “Retrieval-Augmented Generation (RAG) System for CFR Title 14 Regulations.” *TBD.* 2026.

1.3 Background and literature

Recent advances in NLP have highlighted the importance of integrating structured knowledge sources with LLMs to improve reasoning and retrieval capabilities. Among these approaches, KGs have emerged as a powerful mechanism for representing complex relationships and enabling structured access to domain-specific information. When combined with RAG, KGs provide a way to move beyond simple text matching by supporting subgraph retrieval and context-aware reasoning. This integration is particularly relevant for regulatory domains, where hierarchical structures, cross-references, and semantic dependencies must be preserved to ensure accurate interpretation. This section provides an overview of the literature relevant to this work, highlighting some of the technical ideas that were used to support the current project.

1.3.1 Knowledge graph with retrieval augmented generation (KG-RAG)

The use of a KG as an external knowledge source for LLMs has been extensively used in literature. This use is most often associated with RAG, where contextual information is first indexed, to organize the information from multiple knowledge sources, and then retrieved according to a user query. The retrieved data is then used by an LLM to generate an answer to the user query. The use of KGs as part of the RAG framework changes the text retrieval to subgraph retrieval. Multiple strategies can be used in this subgraph retrieval. A summary of these strategies can be found in *Knowledge Graph Combined with Retrieval-Augmented Generation for Enhancing LMs Reasoning: A Survey* (Wang & Shi, 2025). The authors of this paper summarize the use of (1) path-based expansion retrieval, (2) query-based subgraph pattern matching, and (3) embedding-based semantic retrieval. This paper also discusses the use of KGs in the generation step; that is, considerations on the processing of graph data for the answer generation step.

1.3.2 Knowledge graph representation of regulations

The effectiveness of KG-RAG depends heavily on the quality of the KG representation, as it directly influences the accuracy of LLM-generated responses. To assess best practices, relevant literature on KG construction for regulatory text was reviewed.

KGs have been applied to regulatory domains in various ways. For example, Joshi & Saha have developed a knowledge graph to represent the CFR (2020). The objective of this research was to take the initial step towards enabling analytics and question answering of all CFR titles. To do so, the authors used the semantic relationships between legal elements of the CFRs to build a KG that attempts to capture facts and rules of the legal documents. In *A semantically rich framework for knowledge representation of code of federal regulations* (Joshi & Saha, 2020), the methodology for constructing the KG followed a multi-step process. First, each section of the regulations documents was summarized to generate more concise representations of the text. Using these summaries, the top-k topics of the regulations were extracted. The finalized topics served as the entities of the KG. Next, key terms were extracted for each entity and used to assign topic labels. Word2Vec was employed to capture semantic similarity among terms, enabling computation of the likelihood that specific words were compatible or related within the broader text. For each entity, semantically similar terminology was identified and incorporated to support ontology development. The KG was then populated with the validated entities, their definitions, and associated terminology, and relationships between entities were derived by analyzing the most frequent entity–relation occurrences in the text. Validation from a legal expert and authoritative legal references were used to verify the quality of a few steps of this methodology. The generated KG was not tested using a large language querying mechanism, therefore its usefulness for querying remains unverified.

Li, Qian, & Taoxiong created a representation of the Criminal Code of the People's Republic of China and Annotated Code by using a pre-fix fine-tuned LLM to generate knowledge triplets based on the legal text (2024). The knowledge triplets were obtained using a set of nine predefined entities and two predefined relationships. The authors of the text evaluated the capabilities of the LLM to perform named entity recognition and relation classification using labeled data. The results showed that their approach achieved over 90% accuracy, recall, and F1 score. The KG obtained was not used in a decision support tool or language querying mechanism. A drawback of this approach is the need for predefined entities and relationships and labeled data and computational resources required for LLM fine-tuning.

2 Artificial intelligence for ISAM

2.1 Aviation safety question answering dataset for extracting knowledge from incident reports

Aviation incident and accident reports contain vital information needed to understand how safety-critical events unfold, but their unstructured narrative format makes it challenging to extract specific facts at scale. Developing a dedicated question answering dataset addresses this

challenge by enabling automated models, such as LLMs, to identify these facts directly from text and retrieve meaningful safety information quickly and reliably. This provides a foundation for tools that can assist analysts in understanding event sequences, contributing conditions, and safety outcomes.

To produce this dataset, the Methodology section presented below describes the full process used to design questions, select relevant report passages, and generate validated extractive answers. Following this, the Dataset Analysis section evaluates key properties of the final dataset, offering insights into its scale, variability, and relevance to aviation safety tasks.

2.1.1 Methodology

Figure 1 provides an overview of the methodology. The subsequent subsections elaborate on the steps followed to construct the QA dataset.

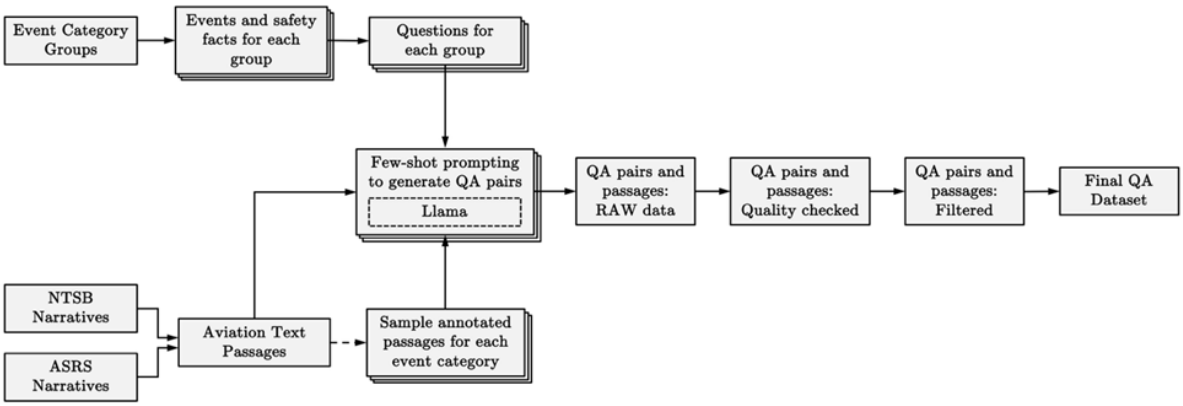

Figure 1. Methodology overview for creating QA dataset

2.1.1.1 Question bank

The first step in creating a QA dataset involved generating a large bank of questions focused on aviation-specific events and safety facts. Using the ISAM ESDs, 37 broad event categories were defined to cover a wide range of incidents and scenarios. Each event category was characterized by a unique initiating event; however, several of these initiating events led to nearly identical sequences of pivotal events and end states. To reflect these commonalities, the 37 event categories were further grouped based on the overall similarity in event progression. For example, while the categories “Unstable approach” and “ATC event during landing” have distinct initiating events, the subsequent sequences of events showed substantial overlap. As a result, these categories were grouped together (see Table 1). For each group, a list of questions was developed to capture both the initiating and pivotal events, as well as the end states.

The questions were designed to elicit fact-based answers, and short, open-ended question formats were deliberately avoided. For example, a question such as “What happened to the aircraft after a loss of power occurred?” is more effective at extracting specific information than a general question like “What happened?” Additionally, to maintain the extractive nature of the dataset, questions that could be answered with Yes/No were not included.

To enhance diversity in question phrasing, each question was supplemented with two rephrased variants that yielded the same expected answer. For example, “How did the turbulence affect the structure of the aircraft?” was rephrased as “In what ways was the aircraft’s frame affected by the turbulence?” and included as a question variant within the same event category group.

In total, a question bank of 244 unique questions was developed to cover events and end states across all groups. Some events appeared in multiple groups, and thus certain questions from the bank were shared among them. For instance, unique questions related to “Personal injury” and “Loss of control” were applicable to a wide range of event categories and were therefore included in several groups. A breakdown of the number of relevant questions for each event category group is provided in Table 1. This breakdown reflects the number of applicable questions selected from the bank to comprehensively represent each group’s event sequences. Due to the overlap in events between groups, a simple sum of question counts in Table 1 should not be interpreted as the number of unique questions in the question bank.

Table 1. Number of questions for each event category group

Event Categories*	Number of Questions (including variations)
1. Aircraft system failure during take-off, **2.** ATC event during take-off, **3.** Aircraft directional control by flight crew inappropriate during take-off, **4.** Aircraft directional control system failure during take-off, **5.** Incorrect configuration during take-off, **6.** Single engine failure during take-off, **7.** Pitch control problem during take-off, **8.** Take-off from a taxiway, **9.** Runway incursion involving incorrect presence of single aircraft for takeoff	42
10. Aircraft takes off with contaminated flight surface, **11.** Ice accretion on aircraft in flight	27
12. Aircraft encounters wind shear after rotation	27
13. Flight crew member spatially disoriented, **14.** Flight crew incapacitation	27
15. Aircraft encounters moderate or greater turbulence, or other atmospheric event, not related to wake, **16.** Wake vortex encounter during departure, **17.** Wake vortex encounter en-route, **18.** Wake vortex encounter during arrival	42
19. Loss of control in-flight (LOC-I) due to inappropriate aircraft handling	24
20. Single engine failure in-flight	36
21. Aircraft handling by flight crew inappropriate during landing roll	30
22. Aircraft directional control related systems failure during landing roll	30
23. Landing on a taxiway	30
24. Landing on wrong runway	30
25. Conflict on taxiway or apron	21
26. Unstable approach, **27.** Aircraft calculated weight and balance outside limits during approach, **28.** Aircraft handling by flight crew inappropriate during flare, **29.** ATC event during landing	39
30. Aircraft encounters wind shear during approach or landing	42
31. Flight control system failure, **32.** Airspeed, altitude or attitude display failure	12
33. Potential conflict with terrain or obstacle	21
34. Fire on-board aircraft	24
35. Aircraft are positioned on collision course	24
36. Cracks in the aircraft pressure boundary	21
37. Runway incursion involving a conflict	21

***** Note: Event categories and groupings are derived from the FAA's ISAM Event Sequence Diagrams (Noh & Shortle, 2018), (Roelen, et al., 2016).

2.1.1.2 Aviation text passages

Given the specialized nature of the QA dataset and its relevance to the aviation domain, the passages chosen for question answering were selected from NTSB and ASRS reports, which provide written accounts of aviation safety incidents and accidents.

2.1.1.2.1 The NTSB database

The NTSB is an independent U.S. federal agency responsible for investigating aviation accidents and incidents, publishing its findings, and maintaining a publicly accessible database of records dating back to 1962. After an accident, a preliminary report is entered into the NTSB Aviation Accident Database, which is later replaced by a detailed, finalized report, including identified causes, once the investigation concludes. The database is publicly accessible (National Transportation Safety Board, 2025) and organized into three time periods: (1) pre-1982, (2) 1982-2008, and (3) 2008 to the present. For incidents after 2008, users can access CAROL (Case Analysis and Reporting Online), an interactive tool that enables the retrieval of accident reports in JSON format (National Transportation Safety Board, 2025). The metadata includes text-based columns such as analysis narratives, factual narratives, and probable cause, which provide a comprehensive description of events leading to the specific incident. The text data columns with narratives used in the present work for passage selection are: `narr_accf` and `narr_accp`. The specific metadata columns used for passage down-selection are: `phase_no`, `eventsoe_no`, `finding_code`, `occurrence_code`, `occurrence_description`, and `finding_description`. A total of 80,749 NTSB reports were considered.

2.1.1.2.2 The ASRS database

The ASRS, developed by NASA, collects voluntary reports of safety-related incidents within the aviation industry, with all personally identifiable information removed. Its primary goal is to pinpoint safety issues within the national airspace system (NAS) to enable preventative measures. The ASRS operates on a proactive model, encouraging aviation professionals – such as pilots, air traffic controllers, and maintenance personnel – to submit reports that highlight potential safety risks before accidents occur. This proactive reporting provides insight into causal factors often absent in accident investigations, including risk-raising elements, detection methods, and attempts to resolve issues.

Throughout its nearly 40-year history, the ASRS has collected over two million reports covering a diverse range of technical and human-related safety events (NASA, 2021). Each entry in the ASRS database includes a narrative describing unique incidents, along with 91 columns of metadata detailing the conditions surrounding each event. This metadata, provided by both

experts and reporters, covers variables such as the type of operation, aircraft type, reporter qualifications, weather, airspace type, and other specific characteristics. Critical incident details appear in the narrative sections, which describe events before, during, and after each occurrence. Accessible through the ASRS website (NASA, 2025), this semi-structured dataset allows users to filter flight narratives and metadata by various factors, including date, aircraft type, environment, location, individuals involved, and event assessments. In the present work, the following columns containing narratives are used for passage selection: Reporter 1 Narrative, Reporter 2 Narrative, Reporter 1 Callback, Reporter 2 Callback, and Reporter 1 Synopsis. The specific metadata columns used for passage down-selection are: Environment-Weather Elements/Visibility, Flight Phase, and Events-Anomaly. A total of 218,285 reports were considered.

2.1.1.3 QA pair generation

A semi-automated approach was used to generate QA pairs for aviation text passages. The automated aspect of this approach involved using the Llama model to first extract answers to the questions prepared in Section 2.1.1.1 from isolated passages in Section 2.1.1.2 through few-shot prompting. This process is detailed below.

2.1.1.3.1 Passage down-selection

Before generating QA pairs, the large corpus of passages described in Section 2.1.1.2 was divided into smaller, manageable lists based on the 37 event categories. Metadata columns in the NTSB and ASRS databases were used as filters to narrow down passages that aligned with the subject of each event category.

In subsequent steps of the QA pair generation process, these down-selected passage lists helped avoid the impractical task of extracting answers to every question from every passage. For instance, it would be highly unlikely to find answers to turbulence-related questions in a passage describing a runway excursion incident. This makes it more efficient to focus questions on down-selected passages with closely related topics.

2.1.1.3.2 Manually annotated sample passages

To prepare for few-shot prompting with Llama in the next step, manually annotated sample passages were created for each event category group. First, 7 to 10 passages per group were chosen from the down-selected passage lists (prepared in Section 2.1.1.3.1). Answers to the corresponding questions (prepared in Section 2.1.1.1) were then manually extracted word for word from each passage. If a passage did not contain an answer to a particular question, the

answer was marked as “NO ANSWER” during annotation. From the initial 7 to 10 passages, three were selected for each group with the most answers to the questions.

All variations of the same question were considered during this process. Each selected sample passage was paired with three sets of question variations, resulting in three distinct sets of QA pairs per passage. This ultimately yielded, for every event category group, three manually annotated sample passages, each represented in triplicate with different phrasings of the questions but identical corresponding answers.

2.1.1.3.3 Generate QA pairs with Llama

Automatic generation of QA pairs was performed using a model from the Llama 3 family (Grattafiori, et al., 2024). Specifically, the 70-billion-parameter, instruction-tuned Llama 3.3 model (Meta, 2024) was used in its 6-bit K-quantized (Gerganov, 2023) form to accelerate inference while maintaining a reasonable level of accuracy. The use of this quantized version provided additional convenience, as inference could be performed on a single NVIDIA A100 80GB GPU without requiring complex parallel processing.

Few-shot prompting was implemented by presenting inference examples to the model in a conversation format, as illustrated in Figure 2. For each event category group, example prompts began with a basic set of guidelines in a prompt prefix, followed by a sample passage from Section 2.1.1.3.2, along with its corresponding questions. The Llama model was then provided with the manually annotated QA pairs for that passage, formatted in JSON. This pattern, consisting of instructions and example responses, was repeated twice using the remaining two sample passages, allowing the model to understand the task requirements. In the final set of instructions, the same prompt prefix was used but paired with an unseen passage from the down-selected list in Section 2.1.1.3.1, along with a unique context ID assigned to it.

The generation process was run separately for each variation of the questions and for each event category group. The resulting QA pairs, each tagged with a unique ID for the passage, were saved in JSON format for processing in subsequent steps.

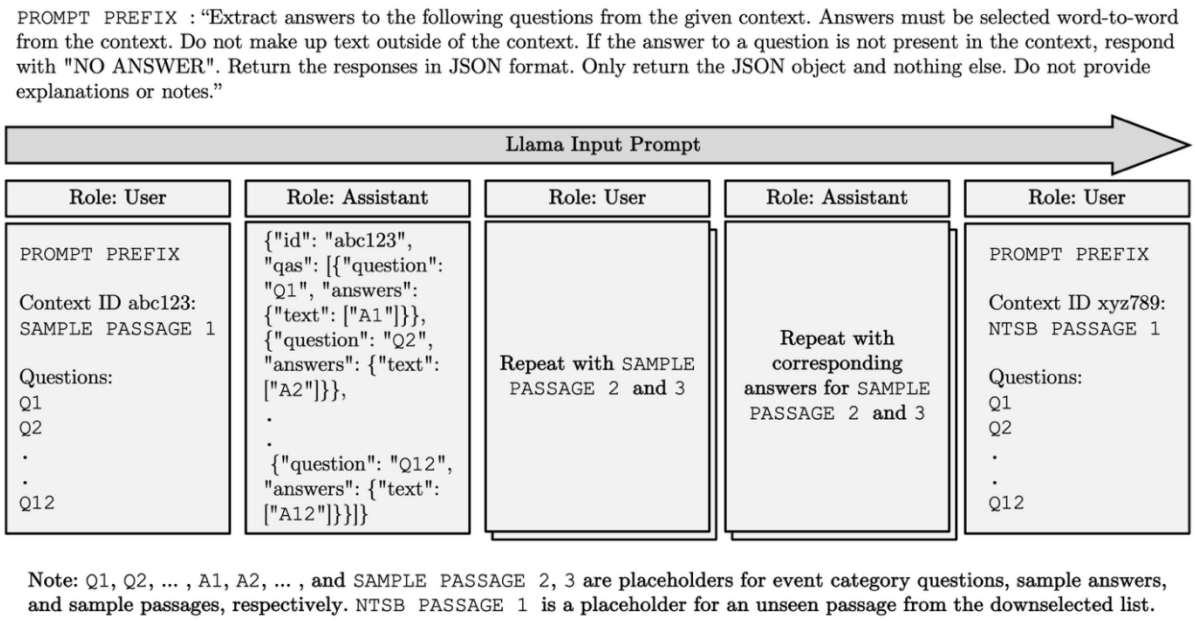

Figure 2. Llama input prompt

2.1.1.4 QA pair quality check

A preliminary scan of the QA pairs generated by the Llama model in Section 2.1.1.3 revealed certain deficiencies in answer quality, necessitating a full-scale quality check of all generated data. Given the large volume of data, manual re-annotation of low-quality answers was not considered feasible. Instead, two approaches were adopted: text replacement and simple removal of poor-quality answers, as detailed below.

Despite strict instructions in the base prompt to extract verbatim answers from the passage text, the Llama model frequently generated responses that were not present word-for-word in the passages. Rather than discarding these responses outright, a Python code was developed to intelligently extract the most relevant passage segments that closely matched the model's answers. The code's text comparison algorithm searched for word patterns and sequences within the corresponding passage to identify segments that best represented the model's answers, even when phrasing or word order differed slightly. The algorithm ensured that the matched segments were not just superficial but semantically meaningful. In cases where suitable exact match could not be found, the QA pair was discarded.

Model hallucinations were also observed. In some cases, the model not only produced irrelevant answers but also generated self-constructed questions. While the exact-match code addressed

hallucinated answers, QA pairs containing questions not found in the original question bank were discarded.

After filtering for exact matches and verifying relevance to the predefined question bank, the remaining QA pairs were manually reviewed as part of the quality check. QA pairs with unsatisfactory answers were discarded; Table 2 provides representative examples with justifications. Particular attention was given to very short answers consisting of only a few words, as these were prone to inaccurate marking of the answer's start position – an essential step in the subsequent data labeling process (Section 2.1.1.5). For instance, the answer "One" might be a valid exact match to the question "How many engines failed during flight operation?" However, since the word "One" may appear multiple times in unrelated contexts within the passage, determining its correct start position becomes overly complex. Consequently, QA pairs with such short answers were also discarded.

After the quality check was complete, all QA pairs containing the placeholder text "NO ANSWER" were removed. It is important to note that only individual QA pairs with unsatisfactory answers were discarded throughout this process, not entire passages.

Table 2. Examples of discarded QA pairs in quality check

Event Question	Sample Discarded Answers	Reason	Sample Retained Answers
How did the turbulence affect the structure of the aircraft?	1) "The right wing hit a construction sign and was substantially damaged", "the airplane was consumed by a post impact fire"	1) The damage is due to an uncontrolled landing or impact, not actually from turbulence.	1) "damage was observed on the right spar"
	2) "airframe ice began to accumulate"	2) Contaminant buildup is covered in another event category and is unrelated to effect on structure due to turbulence.	2) "WE SAW THAT PANELS INSIDE THE CABIN HAD FALLEN OFF"

Event Question	Sample Discarded Answers	Reason	Sample Retained Answers
What caused the turbulence experienced by the aircraft?	1) "turbulence", "severe turbulence", "light to moderate turbulence"	1) These answers do not really provide the cause for turbulence.	1) "wake turbulence"
Where did the aircraft ultimately end up?	1) "we climbed back up to FL380", "touched down", "we landed", "the gate"	1) There is no mention of a specific location such as airport, area of crash, runway/taxiway, ground etc.	1) "they touched down on the gravel runway", "WE LANDED ABOUT 200 FT FROM THE RUNWAY", "Atlanta"
What process was followed to evacuate passengers?	1) "RETURNED TO THE GATE", "REBOARDED ALL PAX LATER ON ANOTHER ACFT", "told passengers to remain seated"	1) Such answers provide no indication of an evacuation, or lack thereof.	1) "taxied back to the gate with no further event"
How many injuries occurred due to turbulence onboard?	1) "injuries", "injuries reported"	1) No true indication of the number of injuries, or any injuries if at all.	1) "no injuries", "neither one of the pilots were injured"
	2) "pilot was killed", "fatal injuries", "The certificated commercial pilot was fatally injured"	2) Fatal injuries or death are a consequence of uncontrolled flight into terrain, not turbulence.	2) "3 PAX AND 1 FA HAD MINOR INJURIES"

2.1.1.5 QA dataset preparation

The extensive lists of passages and QA pairs in Section 2.1.1.4 were further processed to create the final dataset.

QA pairs for the three variations of each question were first merged such that all three variants shared the same answer. This was done by cross-referencing the question variations for each passage and shortlisting the first available answer, in order of appearance: variation one, then two, then three. In cases where a passage contained QA pairs for only one or two variations of a question, the missing answers were populated using the available answer from another variation. For example, a passage might have an answer for the question, “What types of injuries were reported among the passengers or crew?” However, after the quality check and clean-up described in Section 2.1.1.4 , no answers remained for the other two variations – “What was the nature of injuries sustained by those on board?” and “What kind of injuries did the passengers and crew suffer?” In such cases, the answer from the first question was assigned to the remaining variations.

After merging the question variations, passages were filtered to retain only those in which at least 70% of the questions for the corresponding event category had valid answers. For instance, passages in the “single engine failure in flight” category with fewer than 25 QA pairs were filtered out, as this category contained 36 questions in total.

The merging and filtering process resulted in a final dataset comprising 13,658 passages and 351,662 QA pairs. To maintain a SQuAD-like structure, the start position (i.e., the character offset from the beginning of the passage) was recorded for each answer using an automated code. Additionally, the context ID of each passage was suffixed with the text “qXX”, where “XX” denotes the serial number of the question. This was used as a unique identifier for each QA pair within the corresponding passage. Details of the full dataset structure and format are provided in Section 2.1.2.3.

Note that this dataset was intentionally not split into subsets, such as train, validation, or test, in order to provide flexibility to the user. Users may choose a split proportion and strategy based on their specific requirements. A recommended approach is to perform a stratified split based on passages grouped by event category. This ensures that each subset contains unique passages while maintaining equal representation of event categories. Stratification can also be applied at the question level, if desired.

2.1.2 Dataset analysis

This section analyzes the curated aviation-specific QA dataset, focusing on its utility and relevance for aviation safety fact extraction. This analysis aims to evaluate key characteristics of the dataset, providing insights into its structure, content, and potential applications for safety-critical tasks in aviation.

2.1.2.1 Distribution of QA dataset

The word count in both passages and answers was analyzed to better understand the variability in the lengths of the reports reviewed and the responses generated.

Figure 3 shows the distributions of passage length in the dataset with an average length of 745 words. The distribution of context lengths exhibits significant skewness, with a majority of contexts having lengths below 2000 words, while a smaller, yet distinct subset exceeds this length. To provide greater clarity and effectively visualize the frequency and distribution within each of these ranges, the dataset was divided into two subsets, one for contexts with fewer than or equal to 2000 words, and another for contexts exceeding 2000 words. The variation in passage length further reflects the diverse nature of the incident reports considered. Shorter passages might focus on specific events or details, while longer ones might provide comprehensive accounts of the incidents. This diversity can enrich the dataset, making it more representative of real-world aviation scenarios.

The distribution curve of the answer lengths shown in Figure 4 reveals a long-tailed distribution, with the majority of answers falling within a short span of fewer than or equal to 25 words. However, a smaller subset of answers extends significantly beyond this length, representing more detailed or multi-fact responses, particularly for complicated safety situations. To better capture and visualize this variation, the data was partitioned into two ranges: answers with 25 words or fewer, and those exceeding 25 words. This variation implies that the dataset can be used in both short- and long-answer QA models.

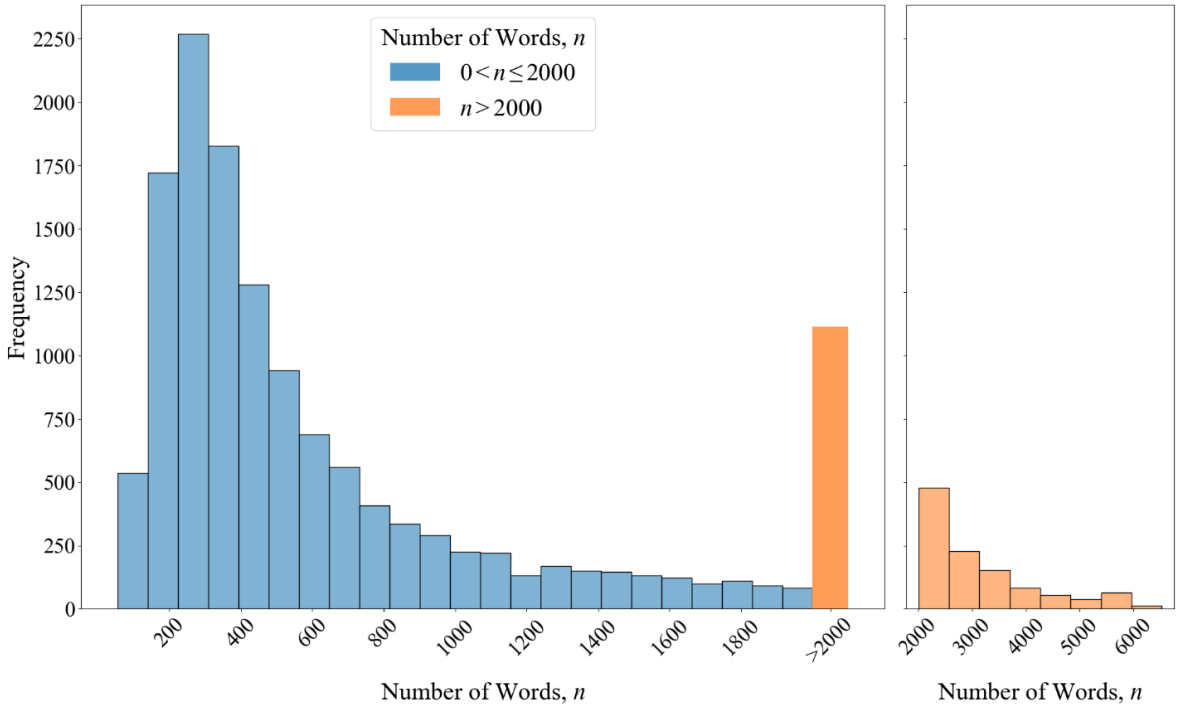

Figure 3. Distribution of passage length in QA dataset

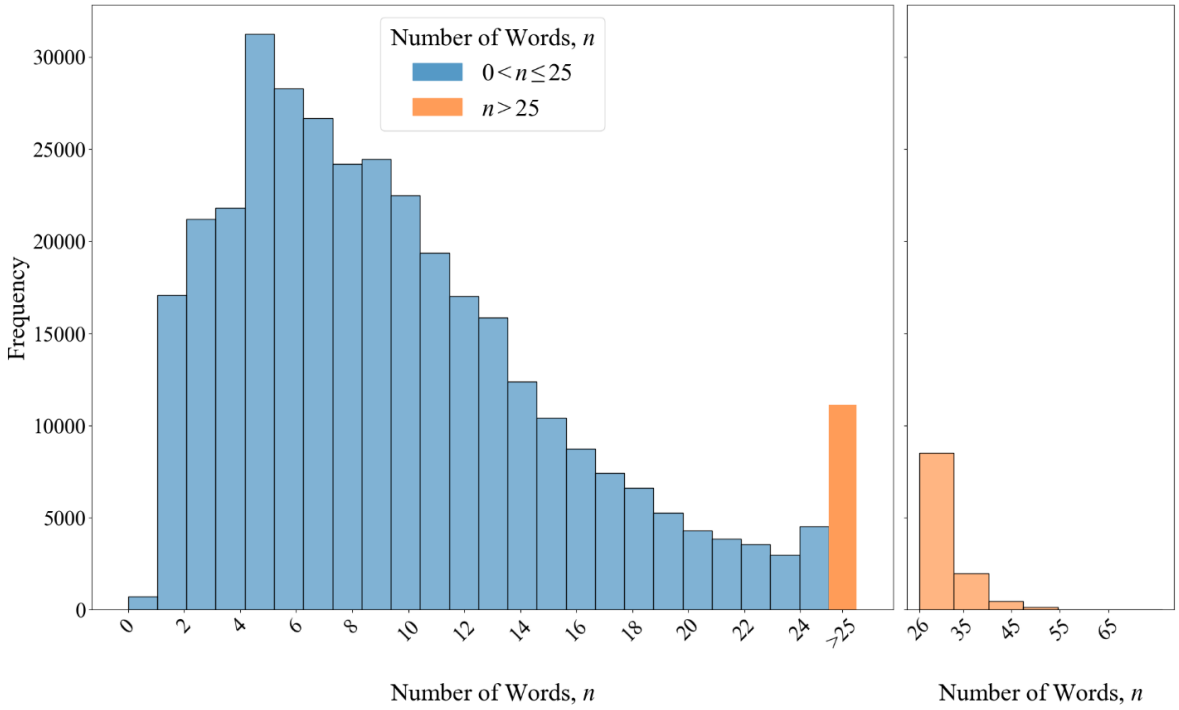

Figure 4. Distribution of answers length in QA dataset

2.1.2.2 Categorization of question types

The questions contained in the database were further categorized based on the type of information they sought, such as fact-based, causal, process-oriented or situational with their respective percentages of the entire database as described in Table 3.

Table 3. Categorization of question types in QA dataset

Category	Description	Example	Percentage
Fact-based	Questions that aim to extract specific facts or information from the report, typically about incident or event, often without requiring further analysis or interpretation	How many engines failed during flight operation? What happened to the power on board during the engine failure?	45%
Causal	Questions that seek to identify the root causes of certain events, behaviors, or outcomes in the incident. They explore underlying factors that contributed to the situation.	What caused the engine failure? Why was the braking ineffective during the event?	13%
Process-oriented	Questions that inquire about the actions taken by the flight crew or aircraft's behavior in response to the incident. They are more about steps taken during the event, often focusing on sequence of actions	What did the flight crew do to restore power after an engine failure? What did the flight crew do to make the aircraft continue flight after a single engine failure?	26%
Situational	Questions that focus on the broader context or conditions surrounding the event, including timing or specific effects on the aircraft and crew. They often reflect the consequences or conditions at a particular time during the event	What condition affected the aircraft's ability to abort takeoff? How did the turbulence affect the structure of the aircraft?	16%

The dataset contains questions that span across a broad range of event categories, with the length of the answers providing valuable insight into the complexity of the safety issues at hand. Using the question category described earlier, a box plot of the average answer word count for each

category was generated, as shown in Figure 5. This reveals some interesting results, for example, process-oriented questions often require answers with more in-depth descriptions of the response of the flight crew to an incident and also the sequence of events, while queries regarding specific facts or information tend to have shorter answers. This variation in response length is crucial for developing models that can handle both simple and complex safety queries.

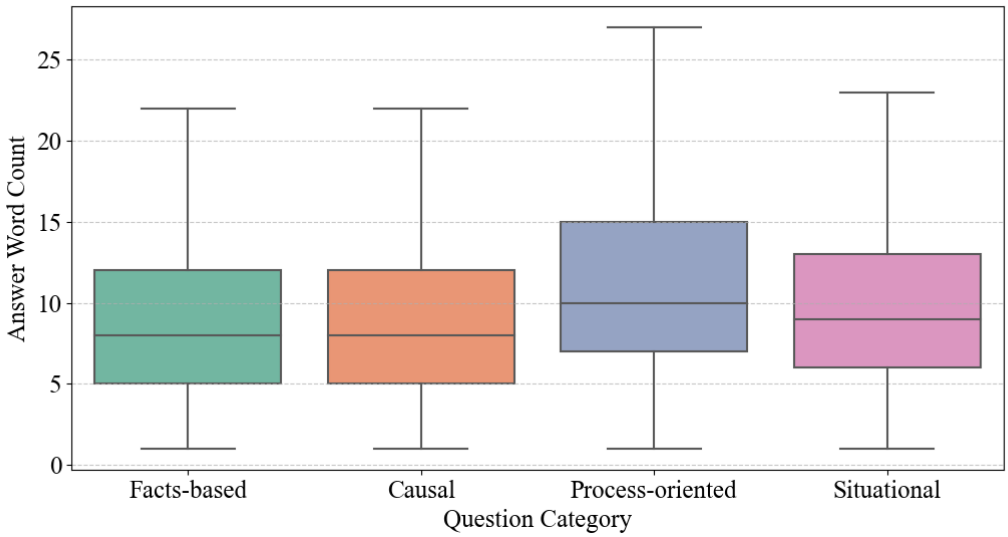

Figure 5. Average answer word count per question category in QA dataset

Figure 6 presents a visualization highlighting the most frequent terms found within the questions of the aviation-specific QA dataset. Using a word cloud, it visually emphasizes the dataset's strong focus on aviation-safety terminology. Prominent words such as "aircraft," "crew," "control," "turbulence," "takeoff," and "passengers" indicate the domain-specific nature and semantic richness of the curated question set. This visualization demonstrates that the dataset predominantly covers critical scenarios and themes pertinent to aviation safety, underscoring its potential utility for developing robust extractive QA models tailored to the aviation domain.

Figure 6. Most frequent terms in QA dataset questions

2.1.2.3 Dataset structure

The created aviation-specific QA dataset is formatted similarly to widely used extractive QA benchmarks such as SQuAD (Rajpurkar, Zhang, Lopyrev, & Liang, 2016). Each entry in the dataset comprises structured elements clearly outlined in Table 4, with a practical example illustrating each component.

Table 4. Structure of the Aviation Safety QA Dataset

Field	Subfield	Type	Description	Example
data	-	List	Collection of topic-wise QA passages	`["event_category": "...", "paragraphs": [...]]`
	event_category	String	Broad category describing the aviation safety event type	`"Single engine failure in flight"`
	paragraphs	List	Collection of paragraphs containing context and QA pairs	`["qas": [...], "context": "..."]`

Field	Subfield	Type	Description	Example
	context	String	Narrative passage from incident reports (source text)	`"On March 20, 2008, about 0750..."`
	qas	List	Collection of question answer pairs relevant to the given context	`["question": "...", "id": "...", "answers": [...]]`
	question	String	Question designed to elicit specific safety information from the passage	`"How did the engine fail?"`
	id	String	Unique identifier assigned to each question answer pair	`"20080611X00830sub00q00"`
	answers	List	Extractive answers directly referencing a span in the context	`["text": "...", "answer_start": ...]`
	text	String	Exact span extracted from the context as the answer	`"experienced a total loss of engine power"`
	answer_start	Integer	Starting character position of the answer within the context passage	`166`

Each entry in the dataset is organized under an event_category, corresponding to specific aviation safety scenarios. Within each event_category, individual paragraphs include a detailed context, extracted from authentic aviation safety reports (NTSB and ASRS narratives), and multiple question answer pairs (qas). Each question answer pair comprises a unique identifier (id), a carefully crafted question, and one answer precisely extracted from the context. The answer entry explicitly provides the exact textual span (text) and the corresponding character starting position (answer_start) within the context passage. This explicit, span-based annotation supports efficient training and evaluation of extractive QA models tailored specifically for the aviation safety domain.

2.1.3 Summary of findings

The resulting aviation QA dataset constitutes a large, structured resource designed to support extractive retrieval of safety-critical information from incident and accident narratives. The final dataset includes 244 unique questions expanded with rephrased variants, covering factual, causal, process-oriented, and situational information. These categories are well represented, giving the dataset balanced coverage of safety-relevant information types. The dataset also contains 13,658 passages from ASRS and NTSB reports matched with 351,662 QA pairs, reflecting broad coverage of aviation events across multiple operational phases and hazard types. Each question is linked to an exact answer span within the passage, enabling precise supervision for LLM fine-tuning. Analysis of the dataset further shows substantial diversity in text length and structure, mirroring the variability found in real aviation reports.

The dataset provides a strong foundation for building and evaluating aviation-specific LLM-based tools. Its size, diversity, and quality-controlled answers make it well suited for training models to identify events, contributing factors, and operational outcomes directly from narrative safety reports, supporting more data-driven safety assessments.

2.2 Knowledge graph-enhanced synthetic report generation for addressing class imbalance in aviation safety data

The effective analysis of aviation safety data is frequently hindered by inherent class imbalance and the inability of traditional augmentation methods to capture the domain's complex technical interdependencies. While LLMs offer a promising solution for synthetic data generation, their propensity for hallucination undermines their reliability in safety-critical applications. To address these challenges, this work proposes a novel approach that synergizes a fine-tuned Mistral-7B model with a domain-specific KG constructed from FAA aircraft data. By employing a dynamic retrieval and prompting strategy, our method grounds generation in verified structured knowledge, effectively producing technically accurate and contextually consistent synthetic reports to supplement underrepresented event categories.

2.2.1 Methodology

Our methodology comprises three main components strategically designed to address the challenge of class imbalance in aviation safety data while maintaining technical accuracy. First, we perform comprehensive data preprocessing and analysis, focusing on NTSB reports spanning from 1989 to 2007. Second, we construct a specialized aircraft knowledge graph from the FAA Aircraft Characteristics Database (Federal Aviation Administration, 2024), implementing a structured ontology using the Resource Description Framework (RDF) (Cyganiak, Wood, &

Lanthaler, 2014) – a W3C standard for representing structured information as graph-based triples – to capture complex relationships between aircraft specifications, operational parameters, and technical configurations. Third, we develop a KG-enhanced report generation system utilizing a fine-tuned Mistral-7B model, incorporating a novel dynamic knowledge integration strategy that efficiently queries and incorporates domain-specific knowledge during the generation process. This integrated approach, illustrated in Figure 7, enables the generation of high-quality synthetic reports while maintaining lower computational overhead compared to traditional Graph-RAG methods (see a detailed comparison in Section 2.2.1.3.2). The system specifically targets underrepresented event combinations, contributing to a more balanced dataset for aviation safety analysis while ensuring technical accuracy through continuous knowledge graph consultation. An overview of our methodology is shown in Figure 7, which illustrates the interaction between data preprocessing, knowledge graph construction, and the report generation system.

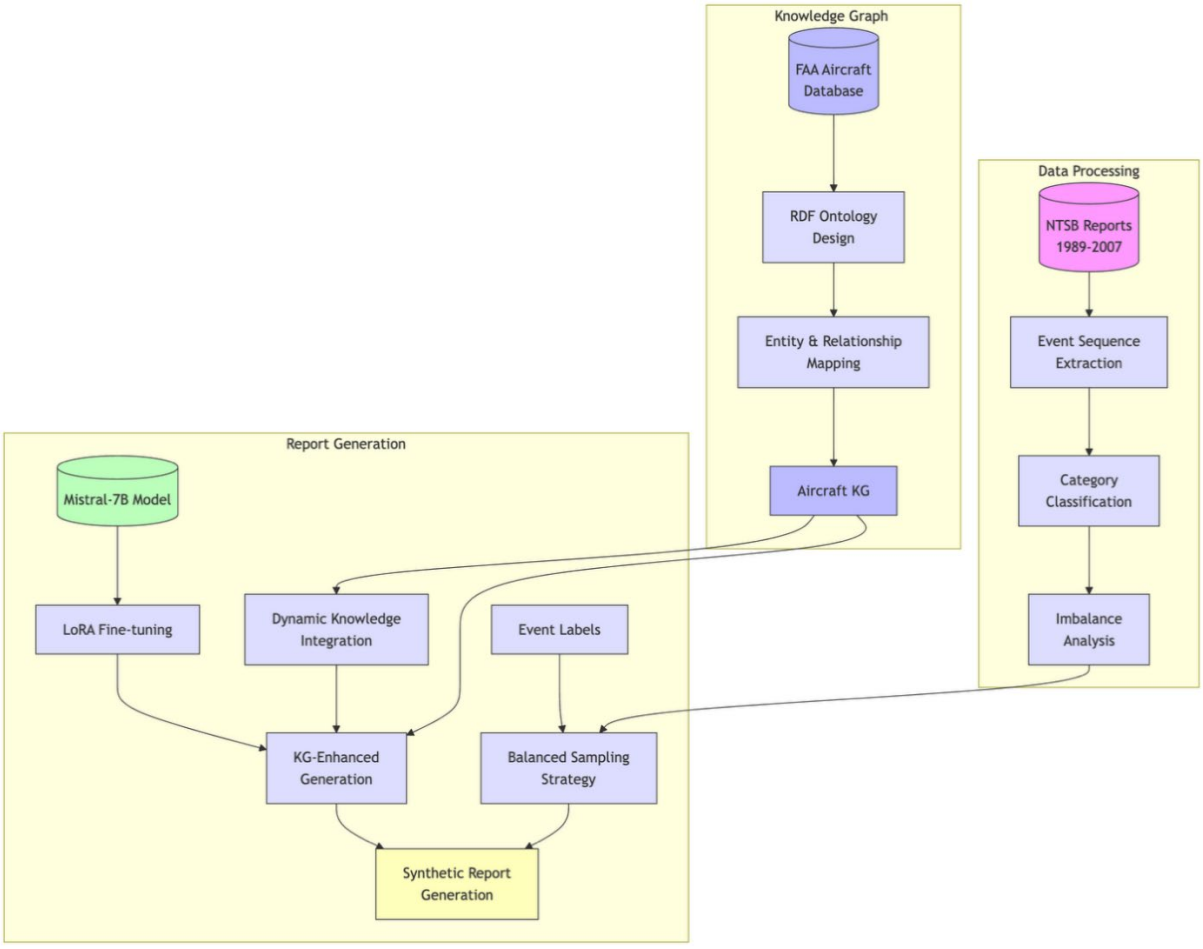

Figure 7. Overall architecture of the proposed methodology for KG construction

2.2.1.1 Data sources

2.2.1.1.1 The NTSB corpus

The NTSB maintains a comprehensive database of aviation accidents and incidents (Federal Aviation Administration, Office of Aviation Medicine, 2001). For our study, we utilized NTSB reports containing detailed narratives and event classifications. Each report includes metadata such as aircraft specifications, flight conditions, and most importantly, event sequences categorized according to the ICAO ADREP taxonomy (International Civil Aviation Organization (ICAO), 2010).

In a previous study, we expanded the Aviation Knowledge Graph (Aviation-KG) using these reports by integrating deep learning models like Aviation-BERT (Chandra, et al., 2023) to enrich the understanding of complex aviation safety narratives (Jing, Sawant, Bendarkar, Elias, & Mavris, 2024). This integration has facilitated the identification of hidden patterns in aviation incidents, enhancing both reactive and proactive safety measures. The expanded Aviation-KG further supports multi-layered classification tasks, offering detailed analysis capabilities for event sequences and enabling advanced query mechanisms to extract critical insights from aviation datasets.

Our analysis revealed a significant class imbalance across the event categories. We identified five main event categories in our previous work (Jing, Sawant, Bendarkar, Elias, & Mavris, 2024): Loss of Control, Collision with Object, Loss of Engine Power, Encounter with Weather, and Other Events. Distribution analysis showed that certain event combinations, particularly those involving multiple concurrent events, were substantially underrepresented in the dataset, with some combinations appearing in fewer than 25 instances. Table 5 presents the detailed distribution of event combinations before synthetic data augmentation, where each row represents scenarios with different numbers of concurrent events per incident.

For each number of concurrent events, the total possible unique combinations for that event count, the number of incident records with that many events, and how many of those combinations appear fewer than 25 times in the dataset are shown along the columns. The table reveals a clear pattern of increasing imbalance as event complexity grows. While single-event scenarios have no underrepresented combinations, multi-event scenarios show progressive data scarcity. Notably, 75% of four-event combinations fall below the 25-instance threshold, and only 84 total records involve four concurrent events. This distribution highlights the challenge of analyzing complex multi-event aviation incidents, where the most critical and potentially catastrophic scenarios involving multiple simultaneous failures are severely underrepresented in the historical data.

In this study, we utilized a total of 24,571 NTSB reports spanning from 1989 to 2007. We applied the predefined labels for the five main event categories: Loss of Control, Collision with Object, Loss of Engine Power, Encounter with Weather, and Other Events (Jing, Sawant, Bendarkar, Elias, & Mavris, 2024). This consistency allows for a direct comparison of findings and ensures that the methodologies developed previously remain applicable to the dataset.

Table 5. Distribution of event combinations before synthetic data augmentation

Number of Events	Total Combinations	Records	Underrepresented Combinations (<25 Instances)
1	5	8714	0
2	9	12100	2
3	9	3664	1
4	4	84	3
Total	27	24562	6

2.2.1.1.2 FAA Aircraft Characteristics Database

To enhance the factual accuracy and contextual relevance of our generated reports, we integrated the FAA Aircraft Characteristics Database (Federal Aviation Administration, 2024). This database serves as a critical resource, offering a wide range of detailed specifications for various aircraft, which are essential for ensuring that the generated synthetic reports adhere to real-world operational standards. By leveraging this dataset, we aim to minimize errors and inconsistencies, particularly in the technical aspects of the reports.

The FAA Aircraft Characteristics Database includes the following key components:

- **Aircraft manufacturers and models**: Provides comprehensive information on the make and model of various aircraft, allowing for accurate representation of specific aircraft types in synthetic narratives.
- **Engine types and configurations**: Details on engine models, including the number of engines and their configurations (e.g., turbofan, piston), are crucial for accurately reflecting the propulsion systems and operational capabilities of different aircraft.

- **Physical characteristics**: Includes critical dimensions such as wingspan, length, and height, which are important for understanding the aircraft's physical constraints and performance in various scenarios.
- **Operational parameters**: Information such as maximum takeoff weight (MTOW), range, and service ceiling provides operational context, ensuring that the generated reports align with realistic flight and performance characteristics.

The FAA Aircraft Characteristics Database is enabling the generation of high-quality synthetic reports that not only enhance data balance but also maintain the technical and contextual rigor necessary for aviation safety analysis. Our aircraft-specific KG is constructed based on methodologies outlined in our previous study (Jing, Sawant, Bendarkar, Elias, & Mavris, 2024), which introduced the expanded Aviation-KG. Due to the ongoing development and expansion of this comprehensive Aviation-KG, the current study primarily showcases integration of aircraft-specific data. In future iterations, we intend to incorporate richer aviation safety information, leveraging broader contextual and causal relationships to further enhance the accuracy and applicability of our synthetic report generation framework.

2.2.1.2 Knowledge graph construction

KGs provide a structured representation of domain knowledge through entities and their relationships. For aviation safety data, we employ RDF (Cyganiak, Wood, & Lanthaler, 2014), a W3C standard that represents information as subject-predicate-object triples. RDF enables machine-readable knowledge representation and supports efficient querying through SPARQL (World Wide Web Consortium (W3C), 2013), making it particularly suitable for encoding the complex relationships inherent in aviation data (Keller, 2019).

2.2.1.2.1 Ontology design

An ontology defines the formal structure of knowledge representation, specifying the types of entities, their properties, and permissible relationships. In RDF-based systems, ontologies provide the schema that governs how knowledge is organized and interconnected (Hegde, et al., 2024). Our aircraft ontology captures the hierarchical and associative relationships between aircraft specifications, operational parameters, and technical configurations.

We developed a specialized ontology for aircraft characteristics using RDF (Cyganiak, Wood, & Lanthaler, 2014). The ontology structure is formalized as follows:

Definition 1 (Aircraft Knowledge Graph) *Let G = (V, E, R) be a directed labeled graph where:*

- *V represents the set of vertices (entities)*
- *E represents the set of edges (relationships)*
- *R represents the set of relationship types*

The ontology hierarchy consists of:

- Classes: $\mathcal{C}$ = *{Aircraft, AircraftModel, Manufacturer, EngineType, NumEngines, Wingspan}*
- Object Properties: $\mathcal{O}$ = *{hasModel, hasManufacturer, hasEngineType, hasNumEngines, hasWingspan}*
- Data Properties: $\mathcal{D}$ = *{icaoCode, wingspanValue}*

Algorithm 1 Knowledge Graph Construction

Require: FAA Aircraft Database $\mathcal{D}_{FAA}$
Ensure: Aircraft Knowledge Graph G
1: Initialize empty graph G
2: **for** each aircraft $a \in \mathcal{D}_{FAA}$ **do**
3: Create aircraft class node v_a
4: Add manufacturer relationship r_m
5: Add engine type relationship r_e
6: Add specifications relationships r_s
7: $G \leftarrow G \cup \{v_a, r_m, r_e, r_s\}$
8: **end for**
9: **return** G

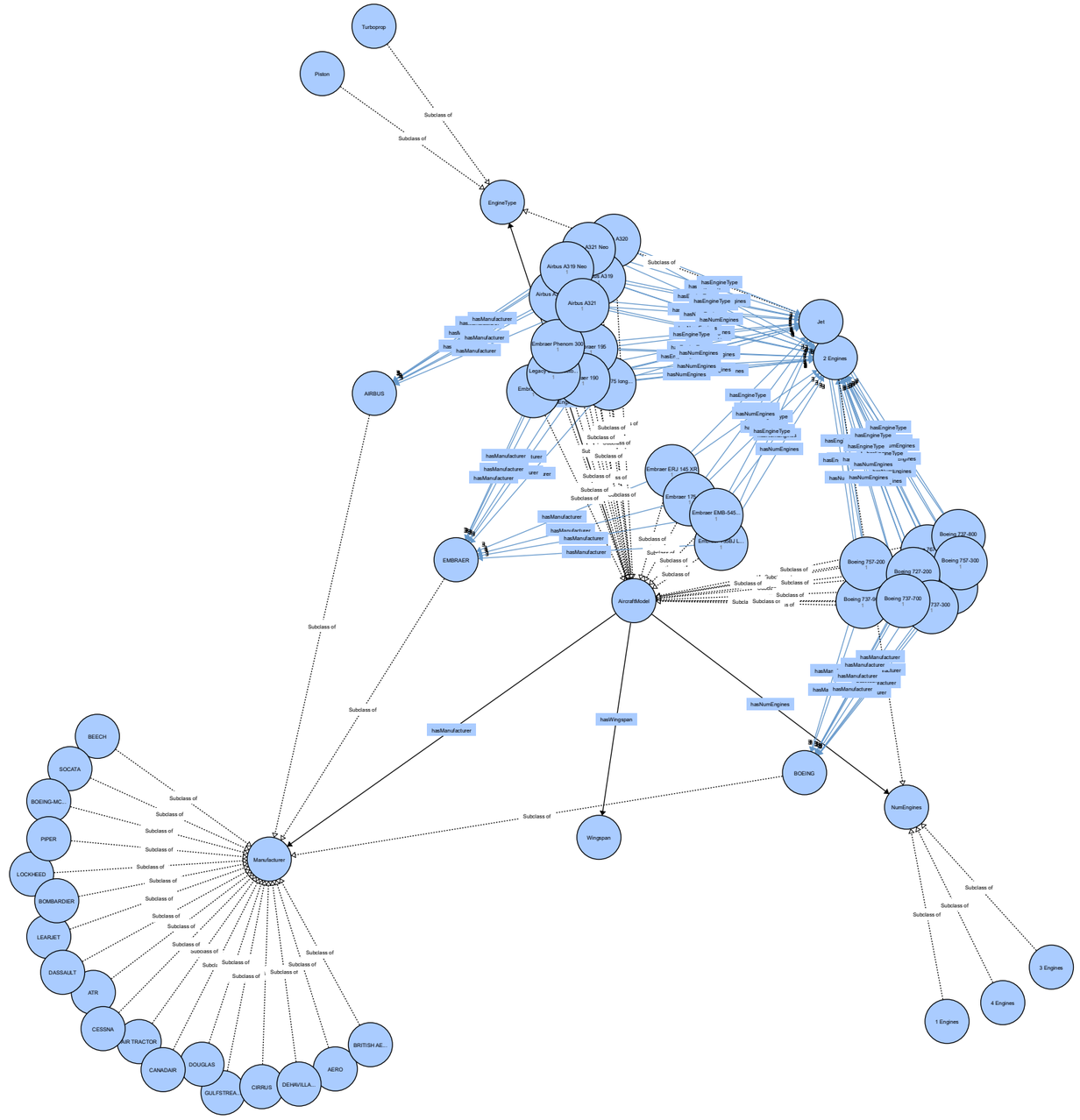

Figure 8. Overview of the Aircraft-KG, illustrating entities such as Aircraft, EngineType, and Manufacturer, along with their relationships and key data properties

Figure 8 provides an overview of the constructed Aircraft-KG. This visualization highlights the key entities, such as Aircraft, EngineType, and Manufacturer, along with their relationships and associated data properties. The diagram serves as a foundational reference for understanding the structure and scope of the KG used in our methodology.

To illustrate the structural clarity of the graph, Figure 9 presents a zoomed-in view of a representative subgraph centered on the AircraftModel class and its associated attributes. In this view, we highlight how each aircraft model is connected to its corresponding Manufacturer via the hasManufacturer property and to the NumEngines class via the hasNumEngines property, which is further refined into specific categories such as 1 Engine, 2 Engines, etc.

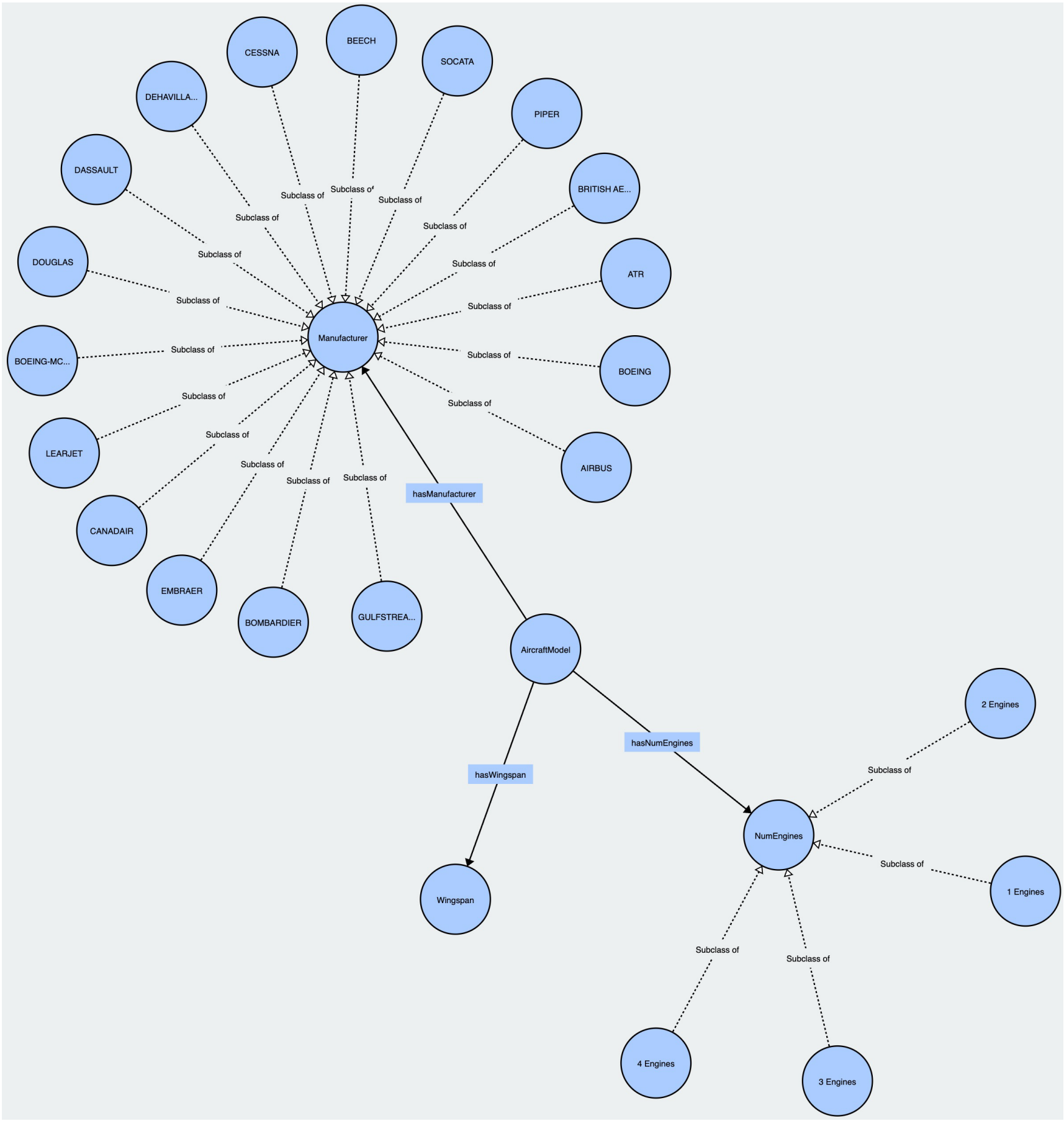

Figure 9. Zoomed-in view of the Aircraft -KG highlighting AircraftModel-level relationships

2.2.1.3 Model architecture and training

2.2.1.3.1 Base model selection and fine-tuning

We selected the Mistral-7B-Instruct-v0.2 model as our base model (et al., 2023). This model, designed with an advanced transformer decoder architecture, provides improvements in attention mechanisms, enhanced context handling, and efficiency in domain-specific tasks. Mistral models have been shown to excel in tasks requiring contextual understanding and nuanced text generation, making them well-suited for generating aviation safety reports.

To efficiently fine-tune the Mistral model on our domain-specific data, we employed Parameter-Efficient Fine-Tuning (PEFT) (Mangrulkar, et al., 2022) using Low-Rank Adaptation (LoRA) (Hu, et al., 2022). LoRA allows us to adapt pre-trained language models to new tasks by optimizing only a small set of additional parameters while keeping the base model frozen. This significantly reduces the computational and memory overhead compared to full fine-tuning, making it particularly suitable for large models like Mistral.

Our fine-tuning process utilized a dataset of 24,571 NTSB report narratives with corresponding multi-label event classifications, forming structured input text as described in Section 2.2.1.1.1. Fine-tuning on this real-world dataset allows the model to better learn the underlying patterns and distributions inherent in aviation safety data, ensuring that the generated outputs align more closely with actual incident characteristics and contextual nuances.

The LoRA configuration was as follows:

- Rank decomposition dimension (r): 8
- LoRA scaling factor (α): 32
- Target modules: query, key, value, and output projections
- Dropout rate: 0.1

Additional training parameters included:

- Training epochs: 1
- Batch size (per device): 1
- Gradient accumulation steps: 4
- Learning rate: 2×10^{-5}

- Warmup steps: 500
- Weight decay: 0.01
- Evaluation strategy: Steps (evaluated every 500 steps)
- Maximum sequence length: 2048 tokens

Given the substantial computational requirements of fine-tuning large models such as Mistral, and considering LoRA's efficiency in rapidly adapting pre-trained weights (Hu, et al., 2022), training for a single epoch is justified, as it sufficiently captures domain-specific patterns without risking overfitting or incurring unnecessary computational costs. The training process was conducted on GPUs with mixed precision (fp16) to accelerate computations and reduce memory usage. We utilized Hugging Face's Trainer Application Programming Interface (API) (Hugging Face, n.d.) to streamline the fine-tuning pipeline with the Mistral-7B-Instruct-v0.2 model (Hugging Face, 2023), ensuring efficient resource management and robust model updates. This approach not only minimized computational costs but also maintained the model's performance, enabling rapid adaptation to the aviation safety domain. The fine-tuned model was subsequently saved for use in synthetic report generation, resulting in outputs that significantly improved contextual relevance and factual consistency.

2.2.1.3.2 Dynamic knowledge integration

We propose a novel dynamic knowledge integration strategy that differs from traditional RAG methods (Gao, et al., 2023), such as Microsoft's GraphRAG (Edge, et al., 2024), Facebook's RAG (Lewis, et al., 2020), and hybrid models like DRAGON (Yasunaga, et al., 2022). While these approaches enhance language models by retrieving relevant external knowledge during generation, they often rely on pre-computed dense embeddings or static graph retrieval, resulting in several limitations:

- **Pre-computation overhead**: Graph-RAG methods require converting all graph entities and relationships into dense vector embeddings before deployment.
- **Quadratic complexity**: Dense similarity search across embedded graph nodes scales as $O(n^2)$ with the number of entities.
- **Storage requirements**: Maintaining embeddings for large KGs demands substantial memory resources.
- **Update inflexibility**: Adding new knowledge requires re-computing embeddings for affected graph regions.

In contrast, our approach leverages the structured nature of RDF knowledge graphs to perform targeted queries with logarithmic complexity. By utilizing SPARQL queries on indexed RDF triples, we achieve $O(\log n)$ retrieval complexity while maintaining the ability to access precise, up-to-date information without pre-computation.

Our dynamic retrieval strategy leverages RDF-based query optimization techniques, which have been shown to significantly improve query processing efficiency in large RDF graphs (Huang, Abadi, & Ren, 2011). Moreover, recent advancements in dynamic knowledge integration highlight the effectiveness of real-time data queries for large-scale knowledge graphs, further validating our approach (Su, Tang, Wu, Ai, & Liu, 2024). Our approach is detailed in Algorithm 2:

Algorithm 2 Dynamic Knowledge Integration

Require: Knowledge Graph G, Event Labels L, Aircraft Model M
Ensure: Enhanced Generation Prompt P
$A \leftarrow \text{QueryAircraftInfo}(G, M)$
$S \leftarrow \text{ExtractSpecifications}(A)$
$P \leftarrow \text{ConstructPrompt}(L, S)$
return P

Our method provides several advantages:

- **Reduced computational overhead**: Unlike GraphRAG (Edge, et al., 2024) or Facebook's RAG (Lewis, et al., 2020), which requires extensive resource allocation for pre-computing document embeddings or dense graph traversal, our method dynamically retrieves and integrates knowledge in real time.
- **Enhanced factual consistency**: By leveraging a domain-specific KG, we ensure that the generated content is grounded in verified aviation data.
- **Scalability and adaptability**: The system can efficiently scale to incorporate additional knowledge sources or adapt to updated datasets.

2.2.1.4 Report generation process

The report generation process combines event labels with the retrieved aircraft information, structured as a conditional language modeling task:

$$P(y|L,S) = \prod_{t=1}^{T} P(y_t|y_{<t}, L, S)$$

where y is the generated report, L represents the input event labels, and S denotes the specifications knowledge retrieved from the aircraft knowledge graph. This formulation ensures that the model generates reports that are both contextually relevant and technically accurate.

We optimized the generation parameters as follows:

- Temperature (τ): 0.7
- Top-k sampling: $k = 50$
- Nucleus sampling: $p = 0.95$
- Repetition penalty factor: 1.1

Algorithm 3 Balanced Report Generation

Require: Event Combinations E, Threshold θ
Ensure: Synthetic Reports R

```
for each combination c ∈ E do
    if Frequency(c) < θ then
        n ← CalculateRequiredSamples(c)
        for i = 1 to n do
            M ← SampleAircraftModel(G)
            P ← GeneratePrompt(c, M)
            R ← R∪ GenerateReport(P)
        end for
    end if
end for
return R
```

This balanced report generation ensures that underrepresented event combinations are supplemented with synthetic reports that adhere to NTSB report formats, improving data balance without compromising factual accuracy.

2.2.1.5 Compute resources

The KG-enhanced synthetic report generation was executed on the PACE supercomputing cluster at the Georgia Institute of Technology using nodes equipped with NVIDIA H200 GPUs (139.8 GB GPU memory), Intel Xeon Platinum 8562Y+ processors with 64 CPU cores (Partnership for an Advanced Computing Environment (PACE), 2017).

2.2.2 Results and discussions

We present our results in three main areas: (1) synthetic data generation statistics and computational requirements, (2) evaluation of synthetic report quality including technical accuracy and lexical diversity, and (3) impact on aviation safety classification performance. Our

findings demonstrate that the KG-enhanced approach addresses class imbalance while maintaining high technical accuracy and linguistic diversity.

2.2.2.1 Synthetic data generation statistics

Our synthetic data generation employed a dynamic threshold-based approach to identify underrepresented event combinations while considering the realistic distribution patterns observed in original NTSB reports. This methodology ensures that we generate combinations with 1 to *N* events based on real data distribution analysis, avoiding unrealistic scenarios where all five event categories occur simultaneously.

Our approach utilizes a 30th percentile threshold to dynamically identify underrepresented combinations, ensuring that we address genuine data imbalance rather than artificially rare occurrences. While our primary goal is to target minority classes, we also generate additional samples for moderately represented combinations to maintain overall dataset coherence and prevent overfitting to extreme cases (Krawczyk, 2016). This balanced augmentation strategy considers both the frequency of occurrence and the practical significance of various event combinations in aviation safety analysis.

The synthetic data generation produced a total of 4060 additional reports with distribution as shown in Table 6, where each report may apply to multiple event categories.

Table 6. Distribution of generated synthetic reports by event category

Event Category	Original Count	Synthetic Count	Total After Augmentation
Encounter with Weather	1285	3060	4345
Loss of Engine Power	6629	3032	9661
Loss of Control	8105	2371	10476
Collision with Object	7185	2229	9414
Other Events	21038	1565	22603

Given the average length of NTSB reports and the complexity of generating contextually accurate aviation safety narratives, we set the maximum token generation to 2048 tokens. Despite utilizing NVIDIA H200 GPU, the generation process required approximately 75 GPU hours to produce 4060 high-quality synthetic reports, reflecting the computational demands of high-quality domain-specific knowledge-enhanced text generation.

2.2.2.2 Evaluation of synthetic report quality

2.2.2.2.1 Qualitative analysis of generated reports

Initial experiments with the **base fine-tuned Mistral model** revealed several limitations in generating aviation safety reports. A notable example of hallucination was observed in a generated report: "...a Cessna TU-172A, N6085M, piloted by an airline transport rated pilot, sustained substantial damage during a forced landing... The pilot reported that both engines had failed..."

This example demonstrates two critical issues:

- Fabrication of an invalid aircraft designation ("TU-172A"), which does not correspond to a valid Cessna model
- Internal technical inconsistency in the generated narrative, as it refers to "both engines" failing while using a Cessna 172-like designation associated with single-engine aircraft

In contrast, the integration of our Aircraft-KG significantly improved the factual accuracy of generated reports. Consider the following example from the KG-enhanced model: "On September 22, 2004, at approximately 1315 central daylight time, a Piper PA-34-200T, N245JK, sustained substantial damage during landing... The pilot reported that during final approach, the left engine began running rough and lost power, requiring a single-engine landing procedure..."

Key improvements include:

- Accurate aircraft model specifications (Piper PA-34-200T is a valid twin-engine aircraft)
- Consistent technical details (appropriate reference to twin-engine emergency procedures)
- Proper integration of aircraft-specific operational characteristics

While our KG-enhanced approach addresses factual accuracy, parallel work has explored an LLM agent framework to automate quality assessment and reduce manual review requirements in synthetic report generation (Jing, Bhanpato, Bendarkar, & Mavris, 2025).

2.2.2.2.2 Quantitative assessment of technical accuracy

While our KG-enhanced approach provides structured knowledge to guide the generation process, it remains crucial to verify the technical accuracy of generated reports. Recent studies have shown that even when language models are provided with explicit factual information through retrieval or prompting, they may still produce hallucinated content that deviates from the provided knowledge (Shuster, Poff, Chen, Kiela, & Weston, 2021; Dziri, Milton, Yu, Zaiane, &

Reddy, 2022). This phenomenon underscores the necessity of systematic evaluation to ensure that our synthetic reports maintain consistency with the aircraft specifications in our KG.

To quantitatively evaluate the technical accuracy of our generated reports, we computed the ASC score as an initial assessment of factual accuracy. We randomly sampled 200 synthetic narratives and evaluated the validity of aircraft model references against our knowledge graph data, which contains 387 validated aircraft models from the FAA database, as mentioned in Section 2.2.1.1.2.

The ASC score measures the proportion of aircraft the model mentions that correspond to valid aircraft types in our KG. For a given narrative, the ASC score is formally defined as:

$$\mathrm{ASC}(n) = \frac{\sum_{m \in M_n} I(m \in \mathcal{A}_{\mathrm{K}})}{|M_n|}$$

where n represents a synthetic narrative, M_n is the set of aircraft the model mentions extracted from narrative n, $\mathcal{A}_{\mathrm{K}}$ denotes the set of valid aircraft model identifiers in our KG, and $\mathrm{I}(\cdot)$ is the indicator function. For narratives with no aircraft model mentions, we assign $\mathrm{ASC(n)} = 1.0$.

The macro-averaged ASC score across all evaluated narratives is:

$$\mathrm{ASC}_{\mathrm{macro}} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{ASC}(n_i)$$

where N is the number of synthetic narratives evaluated.

The results demonstrate high aircraft model validity:

- Macro-averaged ASC score: 1.000
- Aircraft models with perfect validity: 100%

This ASC score of 1.000 indicates that all aircraft model identifiers mentioned in the synthetic reports correspond to valid aircraft types in our knowledge graph, demonstrating the effectiveness of our KG-enhanced approach in preventing the generation of nonexistent aircraft models – a common hallucination issue in unconstrained language generation. While this metric focuses on model validity rather than detailed specification consistency, it represents a critical first step in ensuring factual accuracy for safety-critical aviation applications. A more comprehensive evaluation framework could extend this approach to validate detailed aircraft specifications (e.g., engine count, performance parameters) against KG data, providing deeper insights into technical consistency.

2.2.2.3 Lexical diversity assessment

Lexical diversity assessment evaluates the variety and richness of vocabulary in the generated text, serving as a key indicator of text quality and naturalness. In synthetic text generation, high lexical diversity suggests that the model can produce varied expressions rather than repetitive patterns, which is essential for creating training data that captures the linguistic complexity of real-world documents (van der Lee, Gatt, van Miltenburg, Wubben, & Krahmer, 2019). For domain-specific applications like aviation safety reporting, maintaining appropriate lexical diversity while preserving technical accuracy presents a unique challenge.

Traditional approaches to synthetic data generation, including template-based methods and few-shot prompting with LLMs, often struggle to balance diversity with domain consistency (Brown, et al., 2020; Gatt & Krahmer, 2018). Template-based approaches typically produce formulaic content with limited variation, while few-shot LLM generation may introduce hallucinations or drift from domain-specific terminology (Li, Galley, Brockett, Gao, & Dolan, 2016; Holtzman, Buys, Du, Forbes, & Choi, 2020). This limitation has motivated the development of more sophisticated generation approaches that balance factual accuracy with linguistic diversity (Dhingra, Faruqui, Parikh, Chang, & Cohen, 2019; Goodrich, Rao, Liu, & Saleh, 2019).

2.2.2.3.1 Diversity metrics

We employ three established metrics to quantify lexical diversity following best practices in text generation evaluation (van der Lee, Gatt, van Miltenburg, Wubben, & Krahmer, 2019; Celikyilmaz, Clark, & Gao, 2020):

Distinct-n: Measures the ratio of unique n-grams to total n-grams (Li, Galley, Brockett, Gao, & Dolan, 2016):

$$\text{Distinct-n} = \frac{|\text{unique n-grams}|}{|\text{total n-grams}|}$$

Vocabulary Entropy: Quantifies the uniformity of word distribution (Shannon, 1948):

$$H = -\sum_{w \in V} p(w) \log_2 p\,(w)$$

where $p(w)$ is the probability of word w in vocabulary V.

Type-Token Ratio (TTR): Measures vocabulary richness as the ratio of unique words to total words (Templin, 1957):

$$\text{TTR} = \frac{|\text{unique words}|}{|\text{total words}|}$$

2.2.2.3.2 Results and analysis

Table 7 presents the comparative diversity analysis between real and synthetic narratives, evaluated on all 4060 synthetic reports and equal-sized random samples of real NTSB narratives. Due to the absence of comparable synthetic aviation safety report datasets, we benchmark against the original NTSB data as the gold standard.

Table 7. Lexical diversity metrics comparison

Metric	Real Data	Synthetic Data
Distinct-1	0.0343	0.0306
Distinct-2	0.2198	0.1911
Distinct-3	0.4840	0.4510
Type-Token Ratio	0.0343	0.0306
Vocabulary Entropy	10.04	10.21

The relatively low TTR values (0.0343 for real data) may appear surprising but are characteristic of technical aviation reports, which rely heavily on standardized terminology and fixed phraseology for safety and clarity (Tanguy, Tulechki, Urieli, Hermann, & Raynal, 2016). Our synthetic reports maintain 89.2% (Distinct-1), 86.9% (Distinct-2), and 93.2% (Distinct-3) of the diversity observed in real reports. This close alignment primarily demonstrates the effectiveness of fine-tuning on domain-specific data, which enables the model to learn the natural linguistic patterns of NTSB reports while avoiding the repetitive outputs common in template-based or generic few-shot generation approaches (Gehrmann, et al., 2021).

Notably, the synthetic narratives achieve slightly improved vocabulary entropy (10.21 vs 10.04), suggesting more balanced lexical distribution. This enhancement likely stems from the synergistic effect of fine-tuning and knowledge graph integration: the fine-tuning process captures the linguistic style of aviation reports, while the KG provides accurate coverage of aircraft-specific terminology, helping to reduce repetition and improve consistency in technical references (Logan, Liu, Peters, Gardner, & Singh, 2019). The consistent performance across different n-gram levels indicates that our approach maintains diversity at both word-level and phrase-level granularities (Hashimoto, Zhang, & Liang, 2019).

These results demonstrate that our approach effectively balances two competing objectives: maintaining lexical diversity comparable to real aviation reports while ensuring domain-specific accuracy. The close alignment with real data diversity metrics, combined with perfect technical accuracy (ASC = 1.000), validates the effectiveness of combining fine-tuning with structured knowledge integration for safety-critical applications (Guu, Lee, Tung, Pasupat, & Chang, 2020; Khandelwal, Levy, Jurafsky, Zettlemoyer, & Lewis, 2019).

2.2.2.4 Impact on classification performance

Building upon our previous work with Aviation-BERT (Chandra, et al., 2023) for multilabel classification of NTSB reports' sequence of events (Jing, et al., BERT for Aviation Text Classification, 2023; Jing, Sawant, Bendarkar, Elias, & Mavris, 2024), we evaluated the impact of synthetic data on classification performance. This approach utilizes downstream classifier models fine-tuned on Aviation-BERT, a domain-specific deep learning model. These fine-tuned classifiers have shown superior performance compared to the general-purpose BERT-base-uncased model in multi-label classification tasks. For a detailed discussion on the fine-tuning process of domain-adapted BERT models, readers are encouraged to consult the authors' prior work (Jing, et al., BERT for Aviation Text Classification, 2023).

Table 8 and Table 9 present the classification performance before (Jing, Sawant, Bendarkar, Elias, & Mavris, 2024) and after incorporating the synthetic data:

Table 8. Classification performance before synthetic data augmentation

Category	Precision	Recall	F1-score	Support
Loss of Control	0.78	0.76	0.77	846
Collision with Object	0.84	0.85	0.84	808
Loss of Engine Power	0.87	0.95	0.91	752
Encounter with Weather	0.76	0.23	0.35	210
Other Events	0.93	0.96	0.94	2175
Macro F1-score		0.76		

Table 9. Classification performance after synthetic data augmentation

Category	Precision	Recall	F1-score	Support
Loss of Control	0.76	0.76	0.76	1081
Collision with Object	0.80	0.84	0.82	950
Loss of Engine Power	**0.94**	0.85	0.89	949
Encounter with Weather	0.75	**0.78**	**0.77**	435
Other Events	0.90	0.92	0.91	2220
Macro F1-score		**0.83**		

The most notable improvement was observed in the Encounter with Weather category, where the F1-score increased substantially from 0.35 to 0.77, primarily due to a significant improvement in recall from 0.23 to 0.78. The macro F1-score reflects a significant improvement from 0.76 to 0.83 post-augmentation, indicating better balance across all categories. These results demonstrate the effectiveness of our KG-enhanced synthetic data generation approach in addressing class imbalance. The substantial improvement in the Encounter with Weather category validates our methodology's ability to generate high-quality synthetic reports for minority classes. The 120% improvement in this category's F1-score represents a critical advancement for aviation safety analysis, as weather-related incidents are often complex and require accurate identification.

The overall improvement in macro F1-score from 0.76 to 0.83 indicates that our approach successfully addresses the class imbalance problem by significantly improving performance on underrepresented categories. This pattern aligns with established findings in imbalanced learning literature, where effective methods for minority class augmentation typically show substantial improvements in underrepresented categories (He & Garcia, 2009), (Buda, Maki, & Mazurowski, 2018). The trade-off between individual category performance and overall class balance reflects the inherent challenge in imbalanced learning scenarios. Our results suggest that with additional synthetic data generation, further refinements in the balance between minority and majority class performance could be achieved. The current threshold-based approach for identifying underrepresented combinations provides a scalable framework for continued data augmentation as new incident patterns emerge.

2.2.3 Summary of findings

This study successfully mitigates class imbalance in aviation safety analysis through a novel framework integrating a fine-tuned Mistral-7B model with a domain-specific KG. By employing

a dynamic knowledge retrieval strategy, the proposed approach eliminates hallucinations in aircraft specifications, achieving an ASC score of 1.000, while preserving lexical diversity comparable to real-world reports. The strategic augmentation of underrepresented event categories significantly enhanced downstream classification performance, raising the macro F1-score from 0.76 to 0.83 and demonstrating the efficacy of knowledge-constrained generation in safety-critical domains.

2.3 Development of Regulation-KG for structure-aware retrieval of CFR Title 14

In this section, we describe the design and construction of a structure-aware representation of the U.S. Code of Federal Regulations, Title 14 (Aeronautics and Space), tailored for aviation safety analysis. We develop a regulation-focused knowledge graph, Regulation-KG, that mirrors the CFR hierarchy, preserves canonical identifiers for Sections, and exposes cross-references as explicit, traversable relationships. This structured representation is intended to serve as the backbone for LLM-enabled retrieval, allowing models to retrieve, cite, and explain specific regulatory requirements and constraints directly from the graph rather than relying solely on pattern matching over raw text. By anchoring model behavior in an explicit regulatory structure, the system aims to support precise retrieval, transparent citations, and consistent interpretation of rules across operations and certification while reducing the risk of hallucinated or unverifiable responses. The remainder of this section details the data sources, graph schema, and implementation choices that underpin Regulation-KG and its role in the broader ISAM safety analysis workflow.

2.3.1 Methodology

2.3.1.1 Datasets

2.3.1.1.1 The CFR corpus

The CFR is the codification of the general and permanent rules issued by executive departments and agencies of the U.S. Federal government. It is organized into 50 subject-matter titles that cover broad domains such as aeronautics, labor, and environmental protection. Each title is updated on a fixed annual cycle using a staggered schedule across the year, producing an official "annual edition" that serves as a stable legal snapshot. In parallel, the Electronic Code of Federal Regulations (eCFR) provides a continuously updated version that reflects amendments published in the Federal Register, offering a near-real-time view of the regulatory text.

Within each title, the CFR follows a strict outline: Title, Chapter, Subchapter, Part, Subpart, Section, Paragraph (with alphanumeric subparagraphs). Sections carry unit labels (for example, "§ 91.205"), headings, and body text, and they are often accompanied by "Authority" and "Source" notes that indicate the legal basis for the regulation and its amendment history. The corpus also includes appendices and tables, as well as extensive cross-references and "incorporation by reference" of external standards, which allow agencies to make designated portions of consensus standards legally enforceable without reprinting them in full.

Title 14 (Aeronautics and Space) covers civil aviation and commercial space transportation. Its Parts span product certification (Part 21, 34, 36), airworthiness standards for airplanes, helicopters, engines, and propellers (Parts 23, 25, 27, 29, 31, 33, 35), maintenance and continuing airworthiness (Parts 39, 43, 91, 145), personnel and medical certification (Parts 61, 63, 65, 67), airspace and operating rules (Parts 71, 91, 97, 99, 101, 103, 105, 107), air carrier and commercial operations (Parts 119, 121, 125, 135), airports and training organizations (Parts 139, 141, 142, 145, 147), and commercial space licensing and safety (Parts 400 to 460, including 413, 415, 417, 430, 450). At the Section level, the text exposes consistent fields that are useful for analysis and retrieval: a unit label and heading; applicability and scope statements; normative requirements and performance standards; exceptions, alternatives, or special conditions; cross-references to other sections or incorporated standards; and structured notes such as Authority, Source, and editorial notes indicating amendment history or effective dates. Appendices often provide test methods, performance tables, and compliance criteria tied to specific Parts. Together, these fields allow the reader to determine to an extent who is subject to a rule, what must be demonstrated, how compliance is evaluated, which external standards are binding, and how the provision has changed over time.

2.3.1.2 Knowledge graph structuring

The FAA CFR Title 14 KG, referred to as Regulation-KG, was constructed from regulatory data that was extracted and organized in JSON format. The hierarchy of this structure was subsequently expanded to enhance the usability of the text by introducing paragraph-level nodes, stabilizing identifiers, and preserving cross-references that were previously embedded only as inline citations. The resulting JSON data was converted into RDF Turtle (TTL) format and deployed using a containerized Neo4j graph database (Monteiro, Sá, & Bernardino, 2023). The resulting RDF data, when containerized in a Neo4j database, provides a flexible environment for Neo4j's Cypher query-based analytics (Francis, et al., 2018). This workflow ensures that the KG remains scalable and extensible for future integration with automated reasoning and LLM interfaces.

The method for building Regulation-KG is described below. By converting the CFR's relatively unstructured legal text as an explicit graph of traversable relationships, Regulation-KG enables further precision in access to any rule dependencies, obligations, or cross-references that are critical to aviation safety analysis.

2.3.1.2.1 Text processing and expanding hierarchy

Text data for Federal regulations is available to developers from the Office of the Federal Register (2026) in three data formats: HTML, XML, and JSON. The HTML format contains the full regulatory text along with basic formatting, organized structural elements, embedded cross-reference hyperlinks, and additional metadata. Unlike the HTML data, the XML format includes regulatory text but does not provide hyperlinks or other metadata useful for indexing. The JSON format contains only the hierarchical heading structure, excluding the underlying regulatory text and the lowest level of detail represented at the Section level.

The XML format was not utilized for this research. HTML was selected as the source format for text extraction because its structural elements facilitated reliable parsing. The text extracted from HTML was then incorporated into the JSON structure by adding a "text" key to each node. This augmented JSON format allowed all regulatory text to be available under consistently indexed headings within a unified hierarchy. In addition, construction of the KG was simplified by leveraging the hierarchical organization inherent in the JSON format.

Figure 10 shows the hierarchy levels present in the JSON format as provided in the source data. These levels vary across Chapters in Title 14, but all ultimately terminate at the Section level. Some Chapters end at the Appendix level; however, from a hierarchical standpoint, Appendices can be treated analogously to Sections. The bulk of the regulatory text is located within Sections and Appendices rather than in higher-level headings. As a result, the extracted text appears as a single large block within each Section or Appendix node, without any delineation or distinction among internal bullet points or list items. For example, Section 91.107 (§ 91.107) contains more than 700 words of highly itemized text, in some locations extending six additional levels deep (e.g., § 91.107(a)(3)(iii)(B)(3)(iv) or § 91.107(a)(3)(iii)(B)(2)(i)). Given the density of critical information and the extensive micro-level cross-referencing within such text, the hierarchy was further refined by introducing structure based on list items within each Section. This refinement added new nodes under Sections and Appendices that correspond to individual list items and contain the specific associated text. These nodes were identified as "Subsection," "Subsubsection," "Subsubsubsection," and so on recursively, as shown in Figure 11. The resulting JSON format increased the granularity of the previously large text blocks and improved the ability to cross-reference individual list items.

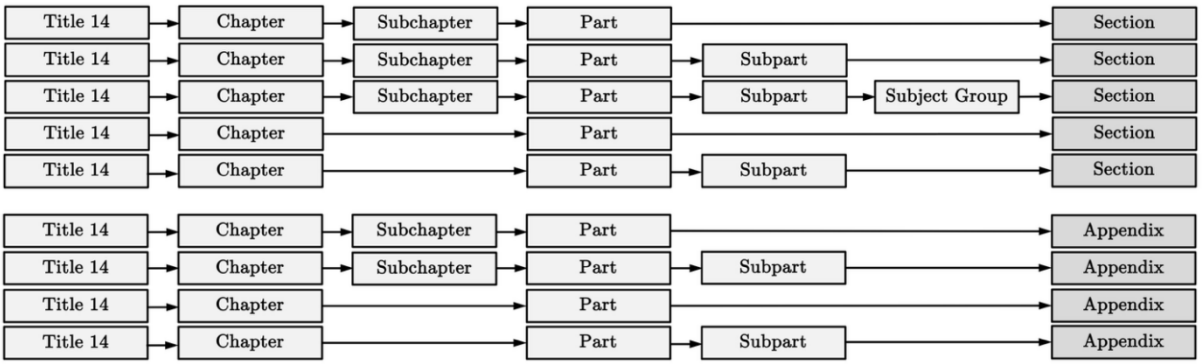

Figure 10. Initial hierarchy of JSON from source data

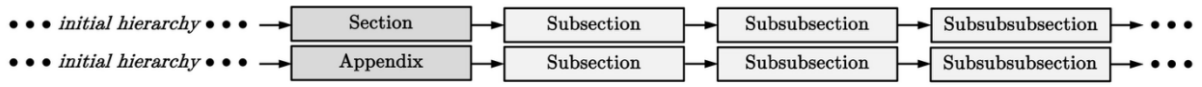

Figure 11. Expanded hierarchy of JSON

An example of the JSON structure with the expanded hierarchy is shown in Figure 12.

"identifier": unique id for each node ; **"label"**: short/general description of the node (blank for levels below Section) ; **"type"**: node type based on hierarchy ; **"text"**: text contained in the node ; **"children"**: any nodes in a level below current


```
{
    "identifier": "title-14",
    "label": "Title 14—Aeronautics and Space",
    "type": "title",
    "text": "Title 14—Aeronautics and Space",
    "children": [
        {
            "identifier": "chapter-I",
            "label": "Chapter I—Federal Aviation Administration, Department of Transportation",
            "type": "chapter",
            "text": "Chapter I—Federal Aviation Administration, Department of Transportation",
            "children": [
                {
                    "identifier": "subchapter-F",
                    "label": "Subchapter F—Air Traffic and General Operating Rules",
                    "type": "subchapter",
                    "text": "Subchapter F—Air Traffic and General Operating Rules",
                    "children": [
                        {
                            "identifier": "91.131",
                            "label": "§ 91.131 Operations in Class B airspace.",
                            "type": "section",
                            "text": "§ 91.131 Operations in Class B airspace.",
                            "children": [
                                {
                                    "identifier": "91.131(b)",
                                    "label": "",
                                    "type": "subsection",
                                    "text": "(b) Pilot requirements.",
                                    "children": [
                                        {
                                            "identifier": "91.131(b)(1)",
                                            "label": "",
                                            "type": "subsubsection",
                                            "text": "(1) No person may take off or land a civil aircraft at an airport within a Class B airsp
                                            "children": [
                                                {
                                                    "identifier": "91.131(b)(1)(ii)",
                                                    "label": "",
                                                    "type": "subsubsubsection",
                                                    "text": "(ii) The pilot in command holds a recreational pilot certificate and has met—",
                                                    "children": [
                                                        {
                                                            "identifier": "91.131(b)(1)(ii)(A)",
                                                            "label": "",
                                                            "type": "subsubsubsubsection",
                                                            "text": "(A) The requirements of § 61.101(d) of this chapter; or"
```


more types: **"part"**, **"subpart"**

more types: **"subsubsubsubsubsection"**, ...

Figure 12. Example of expanded JSON structure

2.3.1.2.2 Initial graph schema

The KG employs a hierarchy-first ontology with the `ns0:` namespace prefix (`http://asdl.gatech.edu/cfr#`) and instantiates node types that mirror the CFR outline exactly through the Section level. The primary classes are `Title`, `Chapter`, `Subchapter`, `Part`, `Subpart`, `Section`, and `Appendix`, as shown in Figure 10. Relations are intentionally minimal at this stage: `hasChild` encodes the vertical parent-child links between adjacent outline levels (e.g., `Part` to `Subpart`, `Subpart` to `Section`), and `refersTo` captures inter-sectional cross-references. Each node carries three core properties that make the graph operational for retrieval and audit: `identifier` (stable, human-readable unit key, e.g., "`91.205`"), `rdfs__label` (the display heading), and `text` (the full, normalized regulatory text for that unit). No subdivision beyond the Section level is introduced in this version; the Section text is treated as a single chunk stored in `ns0__text`.

The deployed graph contains 4,909 typed instances corresponding to the CFR hierarchy: 1 `Title`, 1 `Chapter`, 14 `s`, 91 `Parts`, 402 `Subparts`, 4,259 `Sections`, and 141 `Appendices`. In total, the dataset comprises 47,919 RDF triples. Descriptive properties are well populated: 4,909 nodes have `identifier`, 4,909 have `rdfs__label`, and 4,193 `Sections` (and relevant `Appendix`) nodes include full `text`. Temporal metadata (e.g., `received_on`) are present for 4,403 nodes, although they will be discarded in further evolutions of the graph, as they are not as relevant in their current form. However, they might be considered in the future if need be. Structural connectivity is complete through `Sections`: `hasChild` accounts for 4,767 vertical edges (sum of all parent-child links from `Title` to `Chapter`, `Chapter` to `Subchapter`, `Subchapter` to `Part`, `Part` to `Subpart`, and `Subpart` to `Section`). Lateral connectivity via `refersTo` is already substantive at this baseline, with 10,307 cross-reference edges between provisions.

Figure 13 illustrates a larger subset of the generated Regulation-KG. A smaller subset of the graph highlighting the hierarchy of the KG is shown in Figure 14, where a section is zoomed in for further clarity. Functionally, this hierarchy-first configuration establishes an auditable baseline for later evolution. It guarantees strict outline fidelity (each node's position is unambiguous and every vertical hop is explicit via `hasChild`) and anchors lossless text at the exact unit where obligations reside (the `Section`, with `text` as a single chunk). The presence of `refersTo` enables traversal from a requirement to the provisions it cites or incorporates, while the compact relation vocabulary and stable identifiers (e.g., "`91.205`") support reproducible queries and citation.

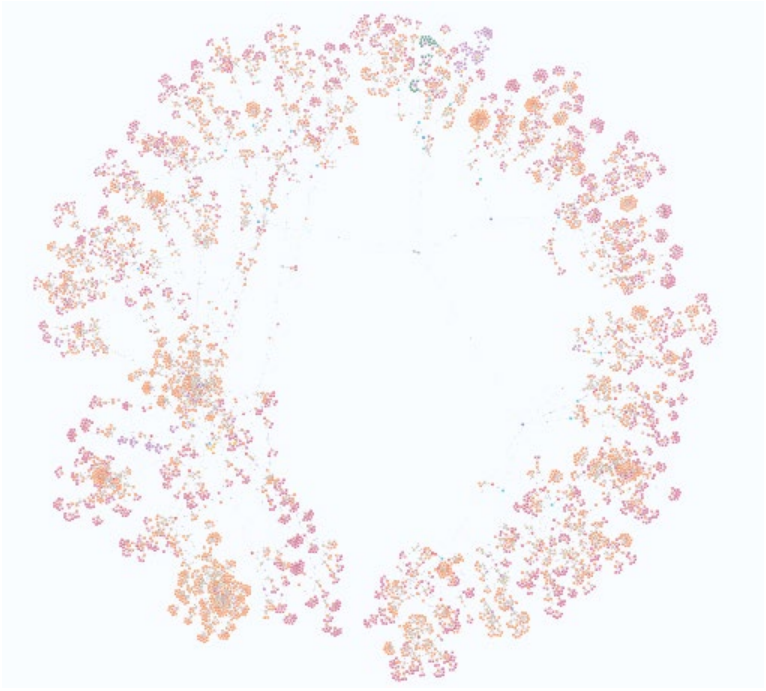

Figure 13. Regulation-KG larger subset visualization (6001 nodes)

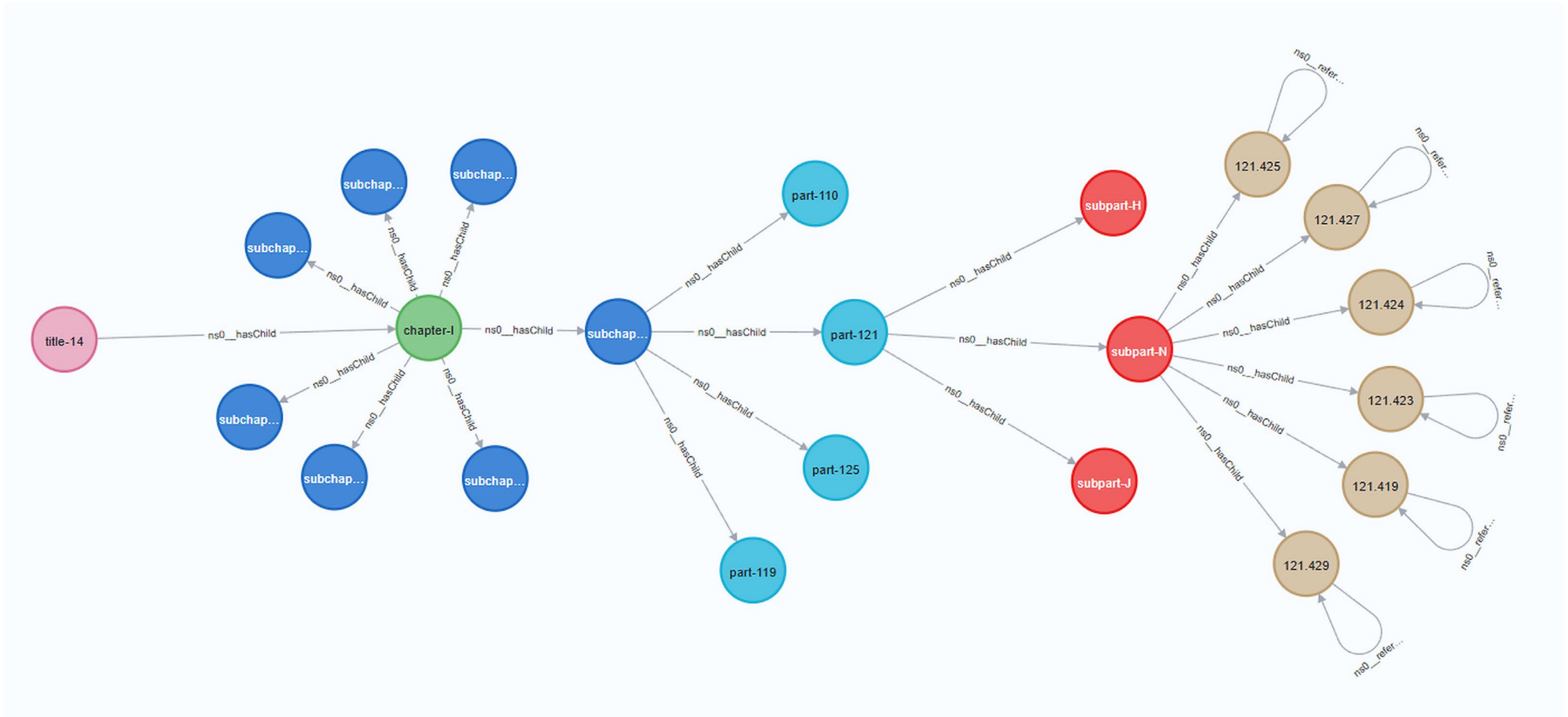

Figure 14. Smaller subset of Regulation-KG visualized (26 nodes)

Cross-reference detection and connectivity: The initial cross-references were extracted only at the Section level using the silcrow symbol and regular expressions that match single- and multi-reference forms (e.g., "§ 91.205," "§§ 25.1329, 25.1327", and common list/range conventions). We normalize typography and whitespace, then apply a resolver that maps each matched token to a canonical `Section identifier` (e.g., "`91.205`", "`25.1329`"), filtering out ambiguous or out-of-scope mentions and connecting a directed `refersTo` edge from the citing to the cited Section. Only Section-to-Section links are produced at this point; paragraph-level citations are deferred to later schema evolution. The addition of these links transforms the graph from a largely tree-structured hierarchy into a densely interlinked network as shown in Figure 13: with 10,307 `refersTo` edges, average Section degree rises markedly, local neighborhoods become richly connected around frequently cited provisions (e.g., § 91.205; § 25.1329), and the number of short simple paths between Parts and Subparts increases sharply. In practical terms, traversal no longer follows a single vertical route; it can fan out laterally across citations, enabling queries that combine hierarchical scope with citation-based evidence and substantially expanding the reach of a single hop.

2.3.1.2.3 Expanded schema of Regulation-KG (Title 14, Chapter I)

The expanded schema resolves structural issues present in the initial graph schema, most notably the unintended duplication of nodes that occurred when Sections and Subparts shared identical identifiers or when paragraph-like content was embedded as inline text rather than represented as distinct nodes. In the initial version, multiple Sections could collapse into a single node if their identifiers appeared identical after text normalization (for example, repeated "Subpart C" blocks across different Parts), and interior paragraphs were not represented as separate entities, leading to overly shallow connectivity and occasional multiple-parent relationships. These issues were eliminated in the expanded schema by generating unique, context-aware Uniform Resource Identifiers (URIs) during JSON parsing, where hierarchical information such as the parent Part and Section identifiers is incorporated directly into each unit's canonical identifier. This ensures that every regulatory element is uniquely addressed and correctly anchored within the hierarchy.

The expanded schema preserves the CFR outline through `Title`, `Chapter`, `Subchapter`, `Part`, `Subpart`, `Section`, and `Appendix`, and introduces paragraph-level structure beneath Sections. Under the same `ns0:` namespace (`http://asdl.gatech.edu/cfr#`), Sections now own child paragraph nodes typed as `Subsection` and consequently deeper paragraph classes such as `Subsubsection` and `Subsubsubsection` where the source enumerates items. Each instance continues to expose `identifier`, `rdfs__label` (when present), and full `text`. For Title 14, Chapter I, the expanded graph comprises 190,116 RDF triples and

43,577 typed instance nodes with identifiers and text. A comparison of nodes between both graphs is shown in Table 10.

Table 10. Class-level node counts in the initial vs. expanded Regulation-KG

Class	Initial Count	Expanded Count
`ns0__Title`	1	1
`ns0__Chapter`	1	1
`ns0__Subchapter`	14	14
`ns0__Part`	91	81
`ns0__Subpart`	402	400
`ns0__Section`	4,259	4,235
`ns0__Appendix`	141	134
`ns0__Subsection`	0	13,391
`ns0__Subsubsection`	0	15,166
`ns0__Subsubsubsection`	0	5,449
`ns0__Subsubsubsubsection`	0	743

2.3.1.3 Graph deployment and query interfacing

We deployed Neo4j Community Edition 5.15 within a Docker container to ensure reproducible deployment across computing environments (Docker Inc., 2024). The container configuration includes two critical plugins: Neosemantics (n10s) for RDF data ingestion (Neo4j Labs, 2023) and Awesome Procedures on Cypher (APOC) for graph algorithms and utilities (Neo4j Labs, 2023). The TTL file is automatically imported during container initialization using a Cypher script that invokes `n10s.rdf.import.fetch()`, transforming RDF triples into Neo4j's native property graph model.

The Neo4j instance exposes two network interfaces: the Bolt protocol for programmatic access and an HTTP-based browser interface for visual exploration. This dual-interface design facilitates both automated querying and human inspection of the KG structure.

We implemented a multi-layered query interface supporting both direct graph queries and AI-enhanced NLP, utilizing the Cypher query language for graph traversal (Francis, et al., 2018).

Cypher query examples: Direct KG queries are expressed in Cypher, Neo4j's declarative graph query language. Example queries demonstrate key access patterns:

```
// Retrieve a specific section with all subsections
MATCH (s)
WHERE s.ns0__identifier = '101.1'
OPTIONAL MATCH (s)-[:ns0__hasChild*1..]->(child)
RETURN s.ns0__identifier, s.ns0__text,
       collect(child.ns0__text) AS children_texts

// Keyword-based full-text search
MATCH (s:ns0__Section)
WHERE toLower(s.ns0__text) CONTAINS 'balloon'
RETURN s.ns0__identifier, s.ns0__text
LIMIT 10

// Cross-reference traversal
MATCH (s)-[:ns0__refersTo]->(ref)
WHERE s.ns0__identifier = '101.1(a)(1)'
RETURN ref.ns0__identifier, ref.ns0__text
```

LLM-enhanced query processing: To bridge the gap between natural language questions and structured graph queries, we integrated Ollama, an open-source framework for running LLMs locally (Ollama, 2024). We deployed the Llama 3.1 model (8B parameters) (Touvron, et al., 2023) on the host machine, accessible via an HTTP API.

The LLM-enhanced query pipeline implements a three-stage process: (1) keyword extraction, where the LLM extracts 2-4 salient search terms from the user's natural language (Jing, et al., BERT for Aviation Text Classification, 2023); (2) graph retrieval, executing Cypher queries using the extracted keywords to identify relevant regulatory sections; and (3) answer generation, where the LLM synthesizes a response grounded in the retrieved regulatory text while citing specific section identifiers.

2.3.1.4 Model Context Protocol integration

To enable standardized tool invocation by LLMs, we implemented a Model Context Protocol (MCP) server (Anthropic, 2024). MCP is an open protocol that allows LLMs to invoke parameterized functions and receive structured responses, facilitating the integration of external knowledge sources into LLM reasoning workflows.

MCP server architecture: Our MCP server exposes five tools via the stdio transport protocol:

1) `get_section(section_id)`: Retrieves the complete text of a specific regulatory section, including all hierarchical subsections.
2) `search_sections(keywords, limit)`: Performs keyword-based search across regulatory text, returning up to `limit` matching sections.
3) `get_cross_references(section_id)`: Traverses `ns0__refersTo` relationships to identify all sections referenced by the specified section.
4) `ask_question(question)`: Orchestrates the AI-enhanced query pipeline as described in Section 2.3.1.3, combining keyword extraction, graph retrieval, and answer generation.
5) `execute_cypher(cypher)`: Provides direct access to the Neo4j database for advanced users requiring custom graph queries.

The MCP server follows the JSON-RPC 2.0 specification (JSON-RPC Working Group, 2013) for request-response messaging and implements proper initialization handshaking to ensure protocol compliance. This architecture enables any MCP-compatible LLM client (such as Claude Desktop, GPT-4 with function calling, or custom agent frameworks) to query the FAA regulations KG without requiring bespoke integration code.

Future integration with agent frameworks: The MCP interface design anticipates integration with multi-agent systems and autonomous reasoning frameworks (Xi, et al., 2023; Wang, et al., 2024). The standardized tool interface enables several advanced use cases: (1) *Regulatory compliance checking*, where an agent can automatically verify if a proposed aircraft operation complies with relevant CFR sections; (2) *Comparative analysis*, where agents can retrieve and compare requirements across different operational categories; and (3) *Explanation generation*, where agents can trace the regulatory reasoning chain by following cross-references through the KG. The separation of concerns between graph querying (Neo4j), NLP (Ollama), and protocol standardization (MCP) provides a modular architecture suitable for future extensions with tools like LangGraph (LangChain AI, 2024), CrewAI (CrewAI, 2024), or Pydantic AI (Pydantic, 2024) agent frameworks.

2.3.2 Preliminary results

2.3.2.1 Descriptive metrics

We compute descriptive metrics directly in Neo4j with Cypher. Class-level counts use queries such as `MATCH (n:ns0__Section) RETURN count(n)`, while relationship distributions use `MATCH ()-[r]->() RETURN type(r), count(*)`. Degree statistics rely on neighborhood sizes (e.g., `size((n)--())`), and hierarchy depth is obtained from bounded variable-length traversals from `ns0__Title` to leaves. Cross-reference density is derived from `ns0__refersTo` edges; entropy of edge usage aggregates relationship counts and applies the information formula. The juxtaposition of these metrics is as shown in Table 11.

Definitions

- **Nodes** $|V|$: total instantiated nodes. A larger $|V|$ indicates a more granular representation of the regulatory corpus (e.g., more paragraph- or clause-level nodes) but also increases storage and query complexity.
- **Edges** $|E|$: total relationships; we also report counts per type (e.g., `hasChild`, `refersTo`). As $|E|$ grows relative to $|V|$, the graph becomes more densely connected, enabling richer traversal but potentially increasing query cost.
- **Node types (count)** : number of distinct labels/classes present. A higher number of node types reflects a more detailed conceptual model (e.g., distinguishing Sections, Paragraphs, Appendices) but can complicate schema management and reasoning.
- **Edge types (count)** : number of distinct relationship types present. More edge types allow finer-grained semantics (e.g., different parent-child relations) but also create a more complex schema that is harder to learn and maintain.
- **Hierarchy depth** : for any Title→leaf path p following `hasChild`, depth(p)=length(p); we report

$$\text{maxDepth} = \max_{\text{p}} \text{depth(p)}, \qquad \text{avgDepth} = \frac{1}{|\mathcal{P}|}\sum_{\text{p}\in\mathcal{P}} depth(\text{p})$$

  Hierarchy depth captures how many levels a user or query must traverse from a root Title to the deepest leaf nodes. Very shallow hierarchies may under-represent structure, whereas excessively deep hierarchies can make navigation and reasoning more complex or inefficient.
- **Average degree (undirected)** :

$$\bar{d} = \frac{2|E|}{|V|}$$

The average degree summarizes how many edges, on average, are incident on a node when ignoring the direction. Higher $\bar{d}$ indicates a more interconnected graph, which can improve connectivity for retrieval but may increase the risk of dense "tangles" that are harder to interpret.

- **Coverage (text)** : share of nodes with `text` populated,

$$\text{coverage} = \frac{|\{\, v \in V : \text{text}(v) \neq \emptyset \,\}|}{|V|}$$

Text coverage indicates how many nodes carry their underlying regulatory text, not just structure. High coverage is desirable, as it ensures that most graph elements are directly usable for retrieval, explanation, and LLM prompting.

- **Cross-reference density (per Section)** :

$$\text{refsPerSection} = \frac{\#\text{refersTo}}{|\mathcal{S}|},\ \text{pctSectionsWithRef} = \frac{|\{s \in \mathcal{S} :\ deg^{+}_{refersTo}(s) > 0\}|}{|\mathcal{S}|}$$

where $\mathcal{S}$ is the set of `Section` nodes. These measures describe how often Sections cite other Sections and how widespread cross-referencing is. Higher values indicate a more interdependent regulatory network, which can improve contextual retrieval but may also involve reasoning about scope and applicability.

- **Edge-type entropy** (edge-usage balance) :

$$H = -\sum_{t \in R} p(t)\ \log_2 p\,(t), \qquad p(t) = \frac{\#\text{edges of type } t}{|E|}, \qquad R = \{\text{edge types}\}$$

Edge-type entropy reflects how evenly different relationship types are used in the graph. Low H means one or two edge types dominate (simpler but less expressive), while higher H indicates a more balanced use of multiple edge types, capturing richer structure at the cost of added schema complexity.

Table 11. Descriptive metrics for the Initial and Expanded graphs (Title 14, Chapter I)

Metric	Initial	Expanded
Nodes $\|V\|$	4,909	43,577
Edges $\|E\|$ (total)	15,074	53,895
`hasChild`	4,767	43,588
`refersTo`	10,307	10,307
Node types (count)	7	16
Edge types (count)	2	2
Hierarchy depth (max)	5	10
Average degree $\bar{d} = 2\|E\|/\|V\|$	3.07	2.47
Coverage (nodes with `text`)	85.40%	100%
Cross-ref density (refs per Section)	2.42	2.43
Edge-type entropy *H* (bits)	0.90	0.70

The expanded graph raises $|V|$ and $|E|$ substantially by adding paragraph-level nodes beneath Sections and connecting them with `hasChild`. This deepens the hierarchy (max depth from 5 to 10) and, because many new nodes are leaves or near-leaves, reduces the average degree $\bar{d}$ even as total edges increase. Coverage improves to 100% when paragraph text is included, meaning that every node in the graph, whether a Section or a paragraph, carries the underlying regulatory text. This is desirable for retrieval and LLM grounding because it eliminates "structure only" nodes and ensures that all traversed paths contain recoverable content. However, it also increases storage cost and may introduce redundancy where multiple nodes share similar text. Cross-reference density per Section stays nearly constant because `refersTo` remains Section-to-Section in both versions. Finally, edge-type entropy *H* declines in the expanded graph, since hierarchical edges account for a larger share of all edges, reflecting a topology that is more tree-dominated while still retaining the lateral citation network.

Overall, these metrics suggest that the expanded KG is more expressive, more text-complete, and structurally more faithful to the CFR's internal organization. The tradeoff is a deeper and more tree-like graph that increases traversal cost but improves recall and interpretability by exposing fine-grained regulatory structure. In practical use, this means that queries targeting specific

obligations or conditions can resolve to the exact paragraph where the requirement is stated rather than relying on Section-level summaries. The expanded hierarchy also might provide clearer provenance for LLM reasoning, as each retrieved node corresponds to a distinct, text-backed regulatory unit. As a result, the richer graph supports more reliable, reproducible, and narrowly scoped regulatory analysis.

2.3.2.2 Broader measures: structural and connectivity diagnostics

We compute these broader metrics in Neo4j with Cypher (Zhong, Wu, Li, Peng, & Wu, 2023). Simple counts use patterns like `MATCH (n:ns0\_\_Section) RETURN count(n)` and `MATCH ()-[r]->() RETURN type(r), count(*)`. Fanout at each level comes from counting children per parent (e.g., Parts per Subchapter, Sections per Subpart). Depth is measured with bounded variable-length paths from `ns0__Title` to leaves. Path uniqueness checks whether a Section has exactly one hierarchical route from Title 14. Path signatures count distinct sequences of edge labels along Title-to-Section chains. Cross-references come from `ns0__refersTo` edges, including average outgoing references per Section. Integrity checks look for multiple parents and cycles under `ns0__hasChild`. These metrics for both the initial and expanded KG are as shown in Table 12.

2.3.2.2.1 Definitions

- **Schema/edge-type complexity**: Distinct relationship types $|R|$ and their usage counts. We also report edge-type entropy

$$H_R = -\sum_{t \in R} p(t)\ \log_2 p\,(t), \qquad p(t) = \frac{\#\text{edges of type } t}{|E|}$$

  A small $|R|$ and low H_R indicate a simple, uniform schema, whereas higher entropy reflects a richer but more complex set of relationship semantics.

- **Level-aware branching**: Average children per parent at each tier; e.g.,

$$\text{avg\_sections\_per\_subpart} = \frac{1}{|SP|} \sum_{sp \in SP} \#\{\, s \mid (sp) \xrightarrow{\text{ns0\_\_hasChild}} (s : \text{ns0\_\_Section}) \,\}$$

  Branching factors reveal how the hierarchy fans out at each level. Higher branching at certain tiers (e.g., Subpart→Section) indicates areas of dense regulatory detail.

- **Average degree (undirected)**: Using the simple graph view,

$$\bar{d} = \frac{2|E|}{|V|}$$

  Average degree summarizes overall connectivity. Higher values indicate a more interconnected graph, while lower values reflect a more tree-like structure.

- **Path uniqueness**: For a Section *s*,

$$\text{numPaths}(s) = \#\{p \mid (t{:}\,\text{ns0\_\_Title}) \xrightarrow{\text{ns0\_\_hasChild}^*} (s)\}$$

Path uniqueness tests whether each Section has exactly one hierarchical route from the Title. The expected value is 1. Values greater than 1 signal duplicate or merged nodes, whereas 1 confirms clean canonical structure.

- **Path signature diversity**: Number of unique edge-type sequences on Title→Section chains; with common-only hierarchy, this is typically 1. Low diversity indicates a uniform hierarchical pattern, while higher diversity would reflect semantically differentiated edge types (e.g., distinct parent-child relations).
- **Cross-reference density**: For Section $\mathcal{S}$,

  $\text{refsPerSection} = \frac{\#\text{ns0\_\_refersTo}}{|\mathcal{S}|}$, $\text{pctSectionsWithRef} = \frac{\#\{s \in \mathcal{S}{:}\,\deg^+_{\text{refersTo}}(s) > 0\}}{|\mathcal{S}|}$

  These measures indicate how much regulatory cross-linking occurs. Dense cross-referencing improves contextual reasoning but increases dependency complexity.
- **Hierarchy integrity**: Counts of multiple parents and cycles under `ns0__hasChild`; both should be 0. Integrity checks ensure the hierarchy is a proper tree. Any multiple parent relationships or cycles imply structural errors that undermine traversal.
- **Type-relation consistency**: Coarse precision that parent/child pairs respect intended tier transitions (Title→Chapter, …, Subpart→Section). High consistency indicates that edges conform to the conceptual structure of the CFR. Deviations signal mislabeling or incorrect hierarchy assignments.

2.3.2.2.2 Changes after expansion and their causes

The largest shifts are driven by the introduction of paragraph-level nodes under Sections. This increases $|V|$ and adds many `hasChild` links, which, as seen in Table 11, deepens the hierarchy from a maximum depth of 5 (Title→Section) to 10 in places where subsections appear. Because the new nodes are mostly leaves or near-leaves, the *average degree* ($\bar{\text{d}} = 2|\text{E}|/|\text{V}|$) decreases (from 3.07 to 2.47) even though the total edge count grows, reflecting a sparser local neighborhood at the newly introduced levels. Branching remains highest from Subpart to Section and is essentially unchanged there (10.59 in both), which is expected, because Section counts did not rise; instead, the expansion adds children beneath Sections. The cross-reference layer remains Section-to-Section in both graphs, so refsPerSection is nearly unchanged (2.42 to 2.43) and pctSectionsWithRef would similarly remain stable; in other words, the citation structure becomes *relatively* lighter compared to the much larger hierarchy, but not absolutely different. Path uniqueness and signature diversity stay at 1, because all hierarchical routes are still unique `hasChild` chains from Title to Section; adding paragraph nodes does not create alternative

Title→Section paths. Finally, integrity checks continue to return 0 for multiple parents and cycles, indicating that the containment semantics are preserved during expansion and that downstream analyses can rely on predictable traversal.

Table 12. Broader structural and connectivity measures for initial vs. expanded graphs (Title 14, Chapter I)

Metric	Initial	Expanded
Distinct relationship types $\|R\|$	2	2
`ns0__hasChild` (count)	4,767	43,588
`ns0__refersTo` (count)	10,307	10,307
Average degree (undirected) $\bar{d} = 2\|E\|/\|V\|$	3.07	2.47
Avg chapters per title	1.00	1.00
Avg subchapters per chapter	14.00	14.00
Avg parts per subchapter	6.50	5.79
Avg subparts per part	4.42	4.94
Avg sections per subpart	10.59	10.59
Path uniqueness (Title → Section)	1	1
Path signature diversity (Title → Section)	1	1
Cross-ref density (refs per Section)	2.42	2.43
Multiple parents under `hasChild`	0	0
Cycles under `hasChild`	0	0

2.3.2.2.3 Preliminary results and system validation

We validated the system through a comprehensive test suite covering all five MCP tools. The test client successfully established a JSON-RPC connection with the MCP server, completed the initialization handshake, and invoked each tool with representative queries.

Test Case 1: Section Retrieval

Query:

```
get_section("101.1")
```

Result: Successfully retrieved Section 101.1 (Applicability) with complete text and seven subsections, including subsection (a)(1) defining moored balloon requirements and subsection (a)(4) specifying unmanned free balloon regulations. The query returned the section hierarchy with proper parent-child relationships preserved.

Test Case 2: Keyword Search

Query:

```
search_sections(["balloon"], limit=3)
```

Result: Returned three relevant sections with high precision: §101.7(a) (hazardous operations prohibition), §101.7(b) (object dropping restrictions), and §101.37(a) (prelaunch notification requirements for unmanned free balloons). The full-text search successfully identified regulatory content containing the keyword "balloon" across different hierarchical levels. Query latency was measured at <200ms, demonstrating efficient graph traversal performance.

Test Case 3: Cross-Reference Traversal

Query:

```
get_cross_references("101.1(a)(1)")
```

Result: Correctly identified one cross-reference relationship: §101.1(a)(1) → §101.7 (Hazardous operations). This test demonstrates successful traversal of `ns0__refersTo` relationships encoded in the KG, validating the integrity of cross-reference links extracted from regulatory text during TTL file construction.

Test Case 4: LLM-Powered Question Answering

Query:

```
ask_question("What are the requirements for moored balloons?")
```

Result: The system extracted keywords using Llama 3.1, retrieved five regulatory sections (§11.25(b)(2), §11.25(b)(5), §129.23(b)(3), §121.547(c)(1), §101.25(d)), and generated a natural language response:

> Based on the retrieved regulatory sections, I can provide information about balloon operations under FAA regulations. According to Section 101.25(d), between sunset and sunrise, a person must obtain prior authorization from the FAA before operating an unmanned free balloon or parachute. While the specific term 'moored balloons' was not found in the retrieved sections, regulations governing unmanned free balloons would typically apply to moored balloon operations. Additionally, Section 101.1(a)(1) defines the applicability of Part 101 to moored balloons exceeding specific size thresholds, which would then require compliance with additional operational restrictions outlined in subsequent sections. For comprehensive requirements specific to moored balloons, I recommend examining Part 101 Subpart A, which contains detailed provisions for kites, unmanned rockets, and unmanned free balloons.

The LLM correctly identified that while the exact phrase “moored balloons” was not present in retrieved sections, related regulations about unmanned free balloons and nighttime flight restrictions (§101.25(d)) provide relevant contextual information. This test demonstrates both the capability and current limitations of the keyword extraction approach, particularly in handling regulatory terminology variations and synonym resolution. This response demonstrates the system's ability to ground generated answers in specific regulatory text while maintaining traceability through section identifiers. The LLM appropriately acknowledged the limitations of the retrieved context and provided actionable guidance for finding more comprehensive information, illustrating the system’s transparency in handling queries where exact matches are not available in the KG.

Test Case 5: Custom Cypher Execution

Query:

```
execute_cypher(
    "MATCH (s:ns0__Section)-[:ns0__refersTo]->(ref)
    RETURN s.ns0__identifier, ref.ns0__identifier
    LIMIT 5"
)
```

Query description: This Cypher query traverses the KG to identify cross-reference relationships between regulatory sections. Specifically, it matches nodes labeled as `ns0__Section` that have outgoing `ns0__refersTo` relationships pointing to other section nodes, then returns the section identifiers for both the source and target sections. This type of graph traversal query

enables regulatory analysts to systematically map citation networks within CFR Title 14, revealing structural dependencies between different regulatory provisions.

Result: Successfully executed the custom Cypher query and returned cross-reference pairs, including:

- §101.1(a)(1) → §101.7 (Hazardous operations)

This demonstrates the system's capability to support advanced users with direct database access for complex analytical queries beyond the predefined tool set.

2.3.2.2.4 Knowledge graph visualization tool

To facilitate intuitive exploration and validation of the KG structure, we developed a static visualization tool that renders local graph neighborhoods using NetworkX (Hagberg, Schult, & Swart, 2008) and Matplotlib (Hunter, 2007). This tool enables researchers to visually inspect the relationships between regulatory sections and verify the correctness of cross-reference links extracted during the TTL-to-Neo4j transformation process.

Visualization Architecture

The visualization tool implements a direct query-and-render pipeline:

1. Connects to the Neo4j database via the Bolt protocol
2. Executes a Cypher query to retrieve the target node and its one-hop neighborhood (nodes directly connected via any relationship type)
3. Constructs an in-memory NetworkX graph representation
4. Renders the graph using Matplotlib's spring layout algorithm with automatic GUI window display

The tool accepts a section identifier as a command-line argument and generates a static visualization showing the local graph topology.

The Cypher query used for neighborhood extraction is:

```
MATCH (n {ns0__identifier: $section_id})-[r]-(m)
RETURN n, r, m
LIMIT 100
```

This query pattern retrieves the central node `n` (matching the specified section identifier), all relationships `r` connected to it (regardless of direction or type), and the neighboring nodes `m`. The

`LIMIT 100` clause prevents excessive rendering overhead for highly connected nodes while ensuring comprehensive coverage of typical regulatory section neighborhoods.

Visual Encoding

The visualization employs a color-coding scheme to distinguish node roles: the central query node (e.g., §101.21) is rendered in sky blue, while neighboring nodes appear in light gray. Node labels display the `rdfs__label` property when available, falling back to `ns0__identifier` for nodes without human-readable labels. The spring layout algorithm positions nodes to minimize edge crossings and reveal structural patterns, with adjustable parameters (`k=0.8`, `iterations=50`) tuned for regulatory graph characteristics.

Figure 15 presents an example visualization for §101.21 (Operating Limitations for Certain Small Unmanned Aircraft). The central node (sky blue) represents the target regulatory section, while neighboring nodes (light gray) show directly connected sections through hierarchical relationships. The graph structure reveals two child subsections (101.21(a) and 101.21(b)) and the parent organizational unit (Subpart C-Amateur Rockets). The graph reveals the hierarchical structure through three key relationships: (1) the parent-child relationship to §101.21(a) (a subsection defining specific limitations), (2) the parent-child relationship to §101.21(b) (another subsection addressing compliance requirements), and (3) the hierarchical connection to the parent Subpart C (Amateur Rockets) node. Notably, the self-loop edge on the central node indicates a reflexive relationship in the knowledge graph schema, potentially representing metadata or versioning information. This local topology exemplifies the dual nature of the knowledge graph: hierarchical `ns0__hasChild` relationships create the regulatory taxonomy, while the parent subpart connection maintains organizational context.

Figure 15. KG visualization of §101.21 and its one-hop neighborhood

Validation and Quality Assurance

The visualization tool serves multiple purposes in the system development workflow: (1) *Schema validation*, verifying that the RDF-to-property graph transformation correctly preserved hierarchical and cross-reference relationships; (2) *Cross-reference verification*, enabling manual inspection of `ns0__refersTo` relationships to ensure they match the source regulatory text; and (3) *Coverage assessment*, identifying isolated nodes or missing relationships that may indicate data quality issues. During system development, we used this tool to inspect over 50 representative sections across different CFR parts, deducing that the 10,307 cross-reference relationships and the complete hierarchical structure were correctly encoded.

2.3.3 Summary of findings

In summary, this section demonstrates that explicitly encoding the native structure of CFR Title 14 into a regulation-focused KG, and preserving stable identifiers and cross-references, yields practical benefits for aviation safety analysis and tool-mediated retrieval. The hierarchy-first schema, followed by the paragraph-level expansion, deepens the graph while maintaining outline fidelity and a compact edge vocabulary. Descriptive metrics show how the expansion increases granularity without compromising structural integrity, and, together with the MCP interface, provide an auditable path from natural-language questions to grounded, citation-bearing answers that can support both day-to-day decisions and broader safety barrier and risk quantification workflows.

2.3.3.1 Future work

The work will extend cross-reference modeling beyond section-to-section links to capture paragraph-level citations, citation ranges, chained and conditioned references, and incorporation by reference of external standards. This includes resolving short-form references (e.g., “this section,” “paragraph (a)”) using local context and disambiguating citations that span Titles and Parts, with the goal of improving local connectivity, recall for narrowly scoped queries, and path-aware reasoning about whether a citation is mandatory, conditional, or informational. A second line of work will increase structural richness in a controlled way by modeling additional regulatory elements as first-class nodes and relations, introducing edition-aware identifiers, and adding domain/range constraints to keep traversal predictable. On the systems side, future efforts will deepen MCP-LLM integration via typed, parameterized tools, retrieval planning, and agent-style workflows for compliance checking and comparative requirement analysis, with benchmarking that covers task-oriented metrics (query selectivity, exploration overhead, type-

relation consistency) as well as latency and throughput. The long-term objective is an end-to-end explainable pipeline in which an LLM can plan, retrieve, verify, and cite with fine-grained control over regulatory structure and cross-references.

2.4 Retrieval-augmented generation system and evaluation framework for CFR Title 14 regulations

Aviation regulatory compliance requires precise, verifiable answers to complex questions about Federal aviation regulations. While traditional keyword search and structured query methods work well for known-item lookups, they struggle with conceptual questions that lack explicit regulatory terminology. To address this gap, we developed a RAG system that combines semantic understanding with regulatory text retrieval. The system employs a hybrid architecture that routes queries through either direct citation lookup for explicit references or semantic vector search for natural language questions, then grounds generated answers in retrieved regulatory context using LLMs. This section presents the system architecture, implementation methodology (including a novel summary-enhanced chunking strategy), demonstration results across multiple query types, and a comprehensive evaluation framework adapted from Retrieval-Augmented Generation Assessment (RAGAS) (Es, James, Espinosa-Anke, & Schockaert, 2024) that extends beyond traditional faithfulness and relevancy metrics to provide granular diagnosis of retrieval and generation quality in regulatory RAG systems.

2.4.1 Retrieval-augmented generation motivation

2.4.1.1 The challenge

Existing methods for querying aviation regulations, such as keyword search and Cypher templates, are powerful tools for structured, known-item queries. However, many real-world questions are nuanced, conceptual, and lack obvious keywords. These queries require a deeper understanding of the dense, interconnected regulatory text rather than simple pattern matching.

2.4.1.2 Objective: a supplementary solution

We developed a RAG system that addresses these limitations through three key capabilities:

- Semantic Understanding: The system understands user queries semantically, capturing intent beyond exact keyword matches.
- Flexible Retrieval: The system retrieves relevant regulatory text even when exact regulatory terms are not used in the query.

- Contextual Reasoning: The system reasons over retrieved context to provide comprehensive and verifiable answers with proper citations.

2.4.2 System architecture

2.4.2.1 Hybrid query processing strategy

The RAG system implements a dual-path architecture optimized for different query types (Figure 16).

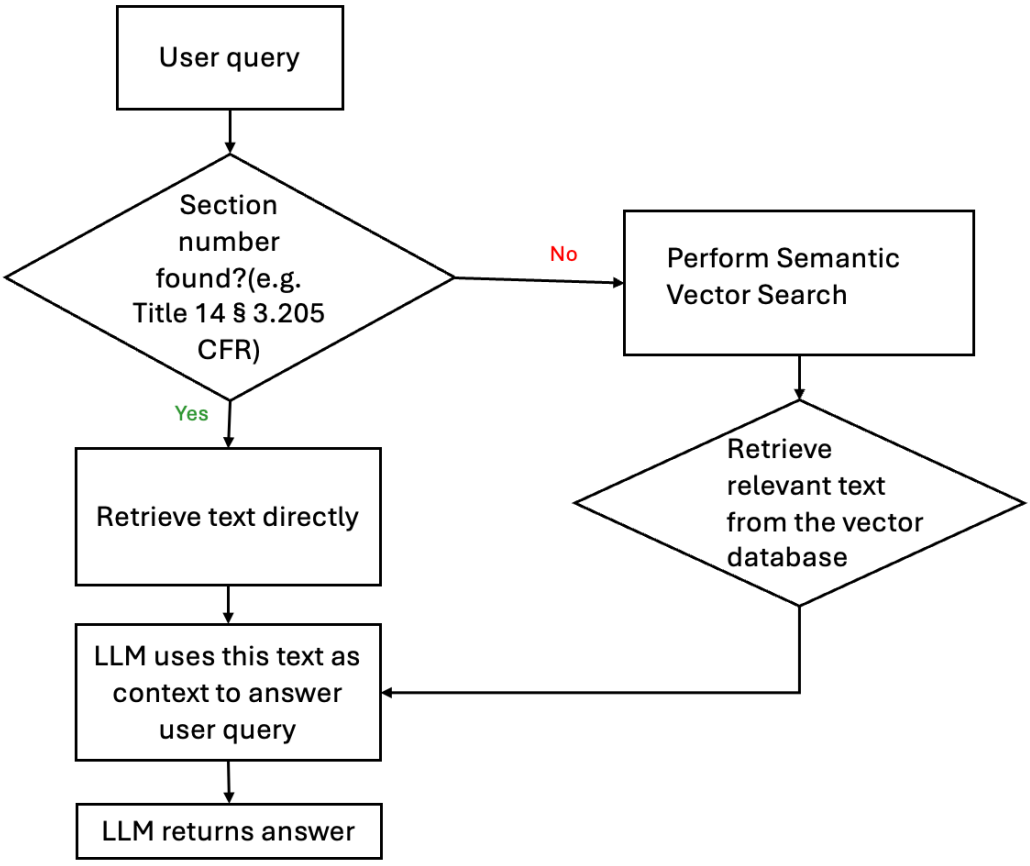

Figure 16. RAG flow chart

Path 1 – Direct Citation Lookup: When a user query contains an explicit section reference (e.g., "Title 14 § 3.205 CFR"), the system performs pattern matching to detect the citation. Upon detection, the system bypasses semantic search entirely and retrieves the regulatory text directly from the database. This deterministic approach ensures exact matching for citation-based queries while significantly reducing computational overhead.

Path 2 – Semantic Vector Search: For queries without explicit citations, the system proceeds with semantic retrieval. The query is first embedded into a dense vector representation, then

compared against pre-computed embeddings of regulatory text chunks stored in the vector database. The system retrieves the most semantically similar passages based on cosine similarity scores.

Response Generation: Regardless of retrieval path, the retrieved regulatory text serves as context for the language model. The LLM synthesizes this context with the original query to generate a comprehensive, grounded answer.

2.4.3 Implementation methodology

2.4.3.1 End-to-end pipeline architecture

The complete RAG pipeline consists of five sequential stages (Figure 17: Semantic Retrieval RAG Architecture):

Regulation Text Loading: All section nodes from CFR Title 14 are extracted from structured CFR documents and loaded into the processing pipeline. The system preserves the hierarchical structure inherent in regulatory text (parts, subparts, sections, and paragraphs).

Text Splitting: The loaded content is divided into logical chunks corresponding to hierarchical regulation units. Rather than using arbitrary character or token limits, the chunking strategy follows the natural structure of regulations. Text is divided into logical chunks labeled as (a), (b), (c), etc., corresponding to subsections. This approach maintains semantic coherence while respecting context window constraints for the embedding model.

Embedding Chunks: Each text chunk is transformed into a dense vector representation using the Nomic-embed-text model. This model generates 768-dimensional embeddings optimized for semantic similarity tasks. The embeddings capture semantic relationships and domain-specific terminology inherent in aviation regulations, enabling the system to match queries with relevant text based on conceptual similarity rather than keyword overlap.

Store Embeddings: Generated vectors are stored and indexed in Facebook AI Similarity Search (FAISS) (Johnson, Douze, & Jégou, 2019; Douze, et al., 2024), a specialized library designed for efficient similarity search over large-scale embedding collections. FAISS enables sub-linear time complexity for nearest neighbor retrieval through approximate search algorithms, ensuring fast query response times even with tens of thousands of embedded chunks.

Query-Time Retrieval and Generation: At query time, user queries undergo the same embedding transformation using Nomic-embed-text. The query embedding is used to retrieve the most relevant text chunks from the FAISS index based on cosine similarity. As shown in the example (Figure 17), when users ask about specific regulatory requirements (such as

applicability criteria for production certificates), the system retrieves the corresponding regulatory text.

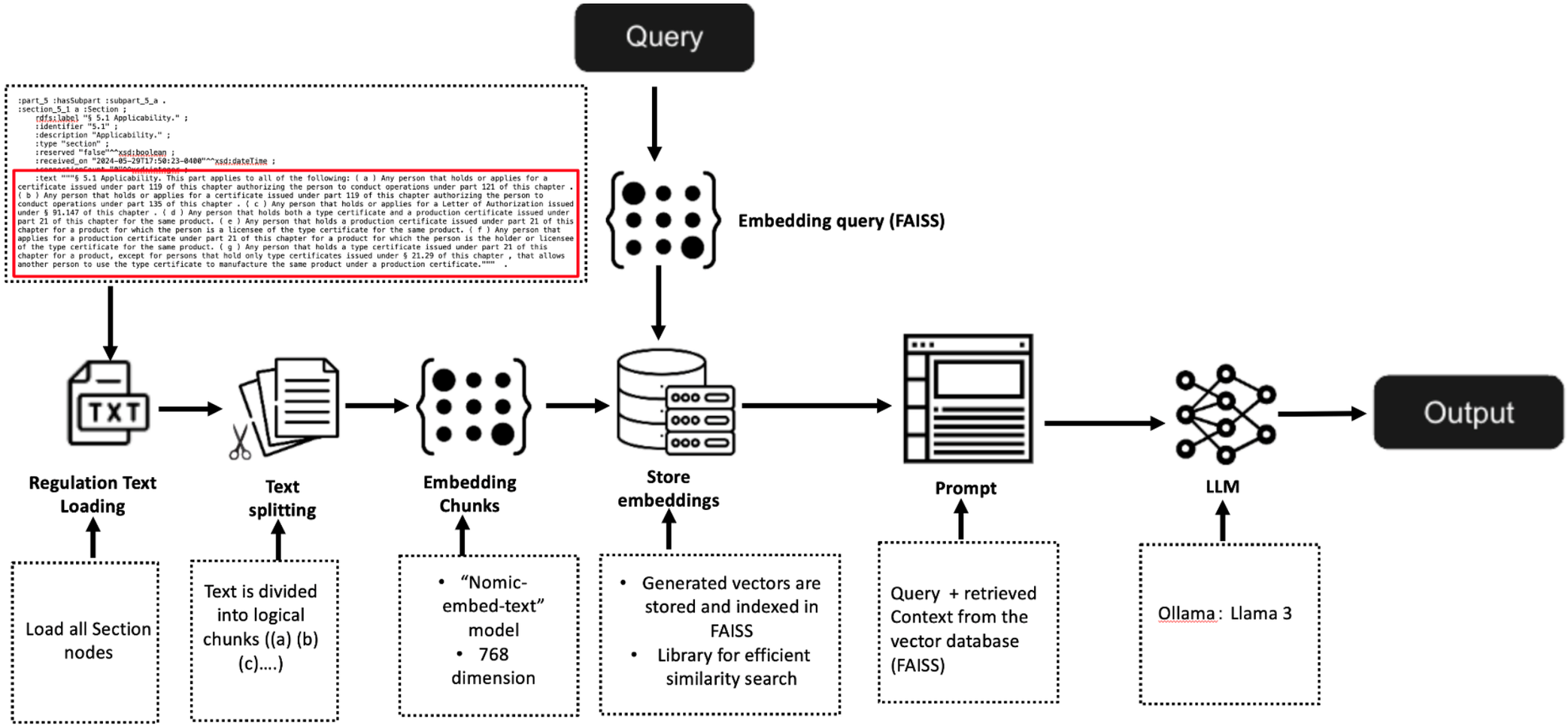

Figure 17. Semantic retrieval RAG architecture

The retrieved context is concatenated with the original query and fed into the LLM (Ollama: Llama 3) (Ollama, 2025) via a carefully designed prompt template. The LLM synthesizes the retrieved regulatory text with its parametric knowledge to generate accurate, contextually grounded responses. This architecture ensures that responses are traceable to specific regulatory sources, reducing hallucination risks while maintaining natural language generation capabilities.

The technical specifications of the RAG system are summarized in Table 13.

Table 13. Technical specifications of the RAG system

Component	Specification
Embedding Model	Nomic-embed-text
Embedding Dimension	768
Vector Database	FAISS (Facebook AI Similarity Search)
Similarity Metric	Cosine Similarity
Language Model	Llama 3 (deployed via Ollama)
Retrieval Method	Hybrid (deterministic lookup + semantic search)
Query Processing	Pattern matching for citations, sematic embedding for natural language

2.4.4 Preliminary results and demonstration

2.4.4.1 Semantic RAG answering Demo 1

Query:

"Summarize the rules on TSO duration and transferability"

Retrieved Context from Vector Database (Top-3 Retrieval):

The semantic search retrieved three candidate text segments:

- **Text 1:** § 21.614 Transferability
- **Text 2:** § 21.613 Duration

- **Text 3:** § 21.601 Applicability and definitions

LLM Answer Grounded in Context:

The system generated a response based on the provided context, summarizing the rules on Technical Standard Order (TSO) duration and transferability. The response correctly cited the relevant sections (14 CFR § 21.613 for duration and 14 CFR § 21.614 for transferability).

Key Observations:

- **Intelligent Filtering and Synthesis:** While the retriever provided three candidate texts, the LLM identified and synthesized information from only the two most relevant sources (§ 21.613 on duration and § 21.614 on transferability), effectively filtering out the less relevant content (§ 21.601).
- **Verifiable and Citable Answers:** The final answer is grounded in evidence and accurately cites the specific regulations (Source: 14 CFR § 21.613, 14 CFR § 21.614).

2.4.4.2 Semantic RAG answering Demo 2

Demo 2.1: Tackling More Difficult Questions from Private Pilot Exam

Query:

"Two-way radio communication must be established with the Air Traffic Control facility having jurisdiction over the area prior to entering which class airspace?"

LLM Answer:

The system correctly identified the answer as **Class C airspace**. According to 14 CFR § 91.130(c), each person operating an aircraft in Class C airspace must meet specific two-way radio communications requirements: arrival or through flight requires establishing two-way radio communications with the Air Traffic Control facility providing air traffic services prior to entering that airspace and thereafter maintain those communications while within that airspace.

Key Observation:

- **Effective Synthesis for Complex Questions:** The system demonstrates the ability to answer real-world questions from the Private Pilot Exam, showing practical applicability.

Demo 2.2: Demonstrating Honesty

Query:

"What are the lighting requirements for drones at night?"

System Response:

- The semantic search retrieved the top three candidates, but none were relevant to the question.
- There is no text in the database relevant to drone-specific regulations.

LLM Answer:

"Unfortunately, there is no specific mention of drones or UAVs in any of the provided regulations, including those related to lighting. For specific guidance on drone regulations, I recommend consulting with the Federal Aviation Administration (FAA) or reviewing relevant laws and regulations."

Key Observation:

- **Faithfulness:** The LLM exhibits "honesty" by refusing to answer questions outside its knowledge base, preventing hallucination and maintaining trustworthiness.

2.4.4.3 Case study: multi-hop retrieval

Query:

"What are VFR visibility requirements?"

Retrieved Context from Vector Database:

The top three candidates retrieved were not directly relevant for the initial semantic search. However, the system identified a related cross-reference pointing to § 91.155, which specifies the basic Visual Flight Rules (VFR) weather minimums.

LLM Answer (Initial):

The LLM could not answer based on the context provided but identified the key clue, § 91.155.

Process:

- Cross-reference was extracted and appended to the candidate list
- Second-round retrieval was performed

LLM Answer (Final):

According to 14 CFR § 91.155, the basic VFR weather minimums are as follows:

- For aircraft other than helicopters:
    - Day: 1 statute mile clear of clouds
    - Night (except as provided in § 91.155(b)): 3 statute miles 500 feet below
- For helicopters:
    - Day: 1/2 statute mile clear of clouds
    - Night (except as provided in § 91.155(b)): 1 statute mile clear of clouds

Analysis:

Strength – Multi-Hop Retrieval Enhances Robustness: The system can follow cross-references to discover the correct context, even when the initial semantic search is imperfect. This demonstrates a form of iterative retrieval that improves answer quality.

Limitation – The Final Answer Is Still Not Perfect: Even with the correct context, the LLM's ability to synthesize complex, conditional information (like the VFR weather table) can be a bottleneck. The answer it produced was factually incomplete because it failed to mention the critical conditions of airspace class and altitude in § 91.155.

2.4.4.4 Summary chunking enhancement: a preprocessing strategy for improved RAG accuracy

2.4.4.4.1 Limitations of traditional chunking

Traditional chunking strategies in RAG systems, while effective for many applications, face three critical limitations when applied to regulatory text.

Semantic Fragmentation: A single, complete thought or regulatory rule is often arbitrarily split across two or more chunks due to fixed token limits. This fragmentation makes it impossible for the retrieval system to grasp the full context of a regulatory requirement. For instance, a regulation specifying conditions for equipment approval may be divided such that the requirements appear in one chunk while the approval criteria appear in another, preventing the vector search from understanding the complete regulatory intent.

Contextual Noise: A single chunk frequently contains multiple distinct, unrelated rules or topics when chunking follows rigid structural boundaries. This mixing of concepts confuses the vector search algorithm, as the embedding must simultaneously represent multiple semantic spaces. The

resulting vector representation becomes diluted, reducing retrieval precision when users query for specific regulatory requirements.

Inability to Handle Semi-Structured Data: Lists, multi-conditional rules, and table-like text present particular challenges for vector search systems. When such structured information is embedded directly, the semantic relationships encoded in the structure (such as hierarchical conditions or tabular comparisons) are lost. Natural language queries struggle to match against these embedded structures because the query and document representations occupy different semantic spaces despite containing relevant information.

2.4.4.4.2 Proposed solution: summary-enhanced chunking

We propose a preprocessing strategy that leverages LLMs to create conceptually dense summaries for embedding while preserving original chunks for final answer generation. This approach addresses the limitations of traditional chunking through a two-stage storage and retrieval mechanism.

Logic-Driven Preprocessing: Prior to embedding generation, we employ an LLM to analyze each original chunk and ensure it represents a logically complete and cohesive unit of thought at an appropriate length. The LLM identifies instances where chunks contain incomplete regulatory statements or span multiple unrelated topics. For chunks requiring adjustment, the LLM either combines fragments that belong together or separates unrelated content, ensuring each processed chunk contains exactly one complete regulatory concept or rule.

Embed the Essence: Instead of embedding the potentially noisy original chunk directly, we generate and embed a concise, conceptually dense summary. The LLM distills the original chunk into its essential regulatory meaning, removing procedural details, examples, and redundant phrasing while preserving all substantive requirements. This summary provides a much cleaner signal for vector search, as it represents a single, focused concept without the noise present in the original text.

Structure to Queryable Format: For chunks containing semi-structured data such as lists, multi-conditional rules, or tabular information, the summary transformation converts these structures into queryable natural language format. For instance, a table specifying different requirements based on aircraft type and operation class becomes a prose summary describing the conditions and their associated requirements. This transformation enables semantic similarity matching between natural language queries and structured regulatory content.

Dual Storage Architecture: The system maintains both the original chunk and its generated summary in the vector database. Critically, only the summary is embedded and indexed for vector search. When a query retrieves relevant summaries based on semantic similarity, the system returns the corresponding original chunks as context for the LLM's answer generation. This architecture ensures high retrieval precision (via clean summary embeddings) while maintaining answer fidelity (via complete original text).

2.4.4.4.3 Implementation example

Consider a regulatory text chunk from 14 CFR § 77.35:

Original Chunk:

§ 77.35 Extensions, terminations, revisions and corrections. (a) You may petition the FAA official that issued the Determination of No Hazard to Air Navigation to revise or reconsider the determination based on new facts or to extend the effective period of the determination, provided that: (1) Actual structural work of the proposed construction or alteration, such as the laying of a foundation, but not including excavation, has not been started; and (2) The petition is submitted at least 15 days before the expiration date of the Determination of No Hazard to Air Navigation.

Generated Summary (Stored in Vector Database):

"summary": "Petitioning for revisions or extensions to a Determination of No Hazard to Air Navigation"

"source_id": "§ 77.35"

Query Processing: When a user queries “How do I extend a no-hazard determination?”, the semantic search compares the query embedding against summary embeddings, retrieving the concise summary above. The system then provides the complete original chunk (with all conditions and details) to the LLM for answer generation, ensuring both high retrieval recall and comprehensive, accurate responses.

2.4.4.4.4 Comparative analysis

To demonstrate the effectiveness of summary chunking, we conducted a side-by-side comparison using a query about forward navigation light requirements.

Query:
"How do brightness requirements for forward navigation lights change with angle?"

2.4.4.4.4.1 Traditional chunking method results

The traditional approach retrieved three candidate chunks, each containing the full regulatory text from different sections (14 CFR § 25.1387, § 29.1387, and § 27.1387). These chunks contained identical or highly similar text describing position light requirements, along with references to unrelated subsections. The vector search struggled to differentiate between the candidates because their embeddings were similar due to shared boilerplate language.

LLM Response (Traditional Method): "I cannot provide an answer based on the provided information because it does not contain any relevant context related to U.S. Federal Aviation Regulations (CFR). The provided content appears to be unrelated to experimental certificates, aircraft certifications, or airworthiness regulations, which are typically the focus of my expertise as an expert assistant for CFR."

The LLM failed to answer because the retrieved chunks, while containing relevant information, were embedded with too much noise for effective semantic matching. The system effectively failed to retrieve the correct regulatory section.

2.4.4.4.4.2 Summary chunking method results

The summary-enhanced approach retrieved three candidates with clear, distinct summaries:

- **Candidate 1 (14 CFR § 25.1391):** Summary stated, "The table specifies the minimum intensity values for forward position lights at different dihedral angles." Original chunk preserved the complete intensity table with specific values.
- **Candidate 2 (14 CFR § 25.1391):** Summary stated, "This section outlines the minimum intensity requirements for forward and rear position lights." Original chunk included the complete regulatory text with the intensity table in the horizontal plane.
- **Candidate 3 (14 CFR § 25.1385):** The summary mentioned position light requirements but addressed installation-related requirements rather than the angle-dependent intensity table. Although retrieved, this candidate was less relevant and was not relied upon in the final answer.

LLM Response (Summary Method): "Based on the provided context, specifically (Source: 14 CFR § 25.1391), we can see that the intensity (in candles) of the forward red and green position lights changes with angle:

- For angles between 0° to 10°: Intensity is 40
- For angles between 10° to 20°: Intensity is 30
- For angles between 20° to 110°: Intensity remains constant at 5”

The summary chunking method successfully retrieved the correct section and enabled the LLM to parse and present the tabular data accurately. The clean summary embeddings matched the query intent effectively, while the preserved original text provided complete information for answer generation.

2.4.5 Completeness evaluation framework for regulatory RAG systems

2.4.5.1 Evaluation dataset construction

2.4.5.1.1 Building a comprehensive QA dataset for regulatory RAG evaluation

Effective evaluation of regulatory RAG systems requires a carefully constructed dataset that captures the full spectrum of question complexity and information structure patterns found in real-world aviation regulatory queries. Traditional question answering datasets often lack the domain-specific complexity and multi-dimensional information requirements characteristic of regulatory compliance questions. To address this gap, we developed a specialized evaluation dataset grounded in CFR Title 14, comprising 6,806 regulatory sections.

Our dataset construction methodology follows a systematic stratified sampling approach. We first applied regex-based text feature analysis to classify regulatory sections into distinct categories based on their structural and semantic characteristics. This classification enables us to ensure representative coverage across different question types that a regulatory RAG system must handle. From this classified corpus, we performed stratified sampling, selecting 50 regulatory sections with 10 samples per category to maintain balanced representation.

For each selected section, we employed LLM-assisted QA generation with explicit ground truth evidence extraction. This approach ensures that every generated question has verifiable answers anchored in specific regulatory text, enabling precise evaluation of retrieval quality. Each QA pair underwent manual quality review to validate question clarity, answer accuracy, and appropriate difficulty level. This rigorous construction process yields a dataset specifically designed to evaluate completeness, the central challenge in regulatory RAG systems where partial or incomplete answers can have serious compliance implications.

2.4.5.1.2 Enhancing ecological validity: integration of pilot examination questions

To complement our systematically generated questions and enhance the ecological validity of our evaluation, we integrated authentic regulatory questions drawn from real-world pilot knowledge examinations. These questions, sourced from the MITRE ALUE benchmark (Mangortey, Singh, Chen, & Sarkhel, 2025), an open-source aviation knowledge exam dataset, represent the types of regulatory queries that aviation professionals encounter in practice.

The original multiple-choice format (A/B/C options) was converted to open-ended question answer pairs. This conversion serves two critical purposes: First, it avoids parametric leakage where the RAG system might generate answers based on memorized training data rather than retrieved context. Second, open-ended formats enable more realistic evaluation of answer generation quality, as systems must synthesize complete responses rather than select from pre-defined options. This format better reflects real-world usage scenarios where users pose questions in natural language and expect comprehensive, synthesized answers.

2.4.5.1.3 Final dataset characteristics

The resulting evaluation dataset comprises 220 carefully curated question answer pairs spanning seven distinct question types, each presenting unique challenges for RAG systems. The distribution reflects both the structural diversity of regulatory text and the variety of information-seeking patterns in aviation compliance queries. Table 14 presents the complete dataset statistics.

Table 14. Dataset distribution of question types in the evaluation dataset

FACTUAL	62	28.2%
CONDITIONAL	36	16.4%
CONCEPTUAL	34	15.5%
MULTI-HOP	26	11.8%
CROSS-REFERENCE	22	10.0%
TABULAR	20	9.1%
PILOT EXAM	20	9.1%
TOTAL	220	100%

Each question type presents distinct evaluation challenges: FACTUAL questions test basic retrieval accuracy; CONDITIONAL questions require understanding regulatory prerequisites and

constraints; CONCEPTUAL questions evaluate the system's ability to connect high-level aviation concepts to the relevant regulatory provisions; MULTI-HOP questions demand reasoning across multiple regulatory sections; CROSS-REFERENCE questions evaluate the system's ability to resolve references between related regulations; TABULAR questions test whether the system can retrieve and synthesize structured or table-like regulatory information; and PILOT EXAM questions assess performance on practical, real-world aviation knowledge questions. The diversity of question types ensures comprehensive assessment of RAG system capabilities across the full spectrum of regulatory query complexity.

Figure 18 illustrates the structure of our QA pairs, showing how each question is linked to ground truth regulatory text evidence, cited subsections, and expected answer content. This structured format enables precise evaluation of both retrieval completeness (whether all relevant regulatory text was found) and answer completeness (whether all required information dimensions were addressed in the generated response).


```
 "section_id": "14-27-27.1401",
      "qa_pairs": [
        {
          "question_id": "14-27-27.1401_Q1",
          "question": "What is the required effective flash frequency range for a rotorcraft's
anticollision light system?",
          "answer": "According to 14 CFR § 27.1401(c), the anticollision light system must provide
an effective flash frequency of not less than 40, nor more than 100, cycles per minute. This
frequency is observed from a distance and applies to each sector of light. In overlap areas where
multiple light sources exist, flash frequencies may exceed 100 but must not exceed 180 cycles per
minute.",
          "question_type": "FACTUAL",
          "difficulty": "easy",
          "requires_table": false,
          "cited_subsections": ["(c)"],
          "ground_truth_evidence": "(c) Flashing characteristics. The arrangement of the system,
that \nis, the number of light sources, beam width, speed of rotation, and \nother characteristics,
must give an effective flash frequency of not \nless than 40, nor more than 100, cycles per minute.
The effective flash \nfrequency is the frequency at which the rotorcraft's complete \nanticollision
light system is observed from a distance, and applies to \neach sector of light including any
overlaps that exist when the system \nconsists of more than one light source. In overlaps, flash
frequencies \nmay exceed 100, but not 180, cycles per minute."
        },
```


Figure 18. Sample QA pairs format

2.4.5.2 Evaluation framework: addressing the completeness challenge

2.4.5.2.1 The completeness problem in regulatory RAG systems

Evaluating RAG systems for regulatory question answering presents fundamental challenges that distinguish this task from traditional natural language generation evaluation. While established metrics such as BLEU and ROUGE effectively measure text similarity in general domains, they fall short in capturing the critical dimensions of regulatory RAG performance: factual grounding,

information completeness, and absence of hallucinations. For safety-critical aviation regulations, partial or incomplete answers represent not merely suboptimal performance but potentially dangerous guidance that could lead to regulatory noncompliance.

The central challenge we address is the "missing zone problem." Consider the layered structure of knowledge in a RAG system shown in Figure 19.

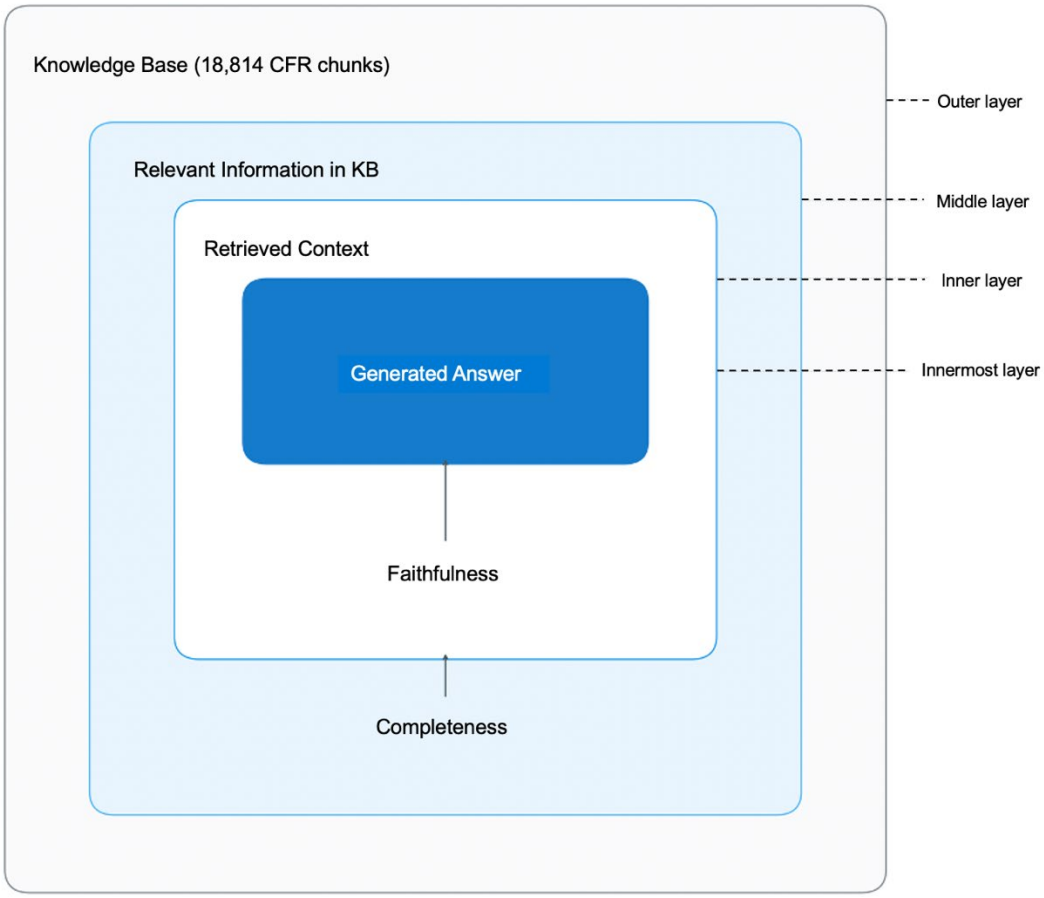

Figure 19. Layered structure of knowledge in a RAG system

Information gaps remain undetected because the system does not know what it does not know. Prior work has identified two distinct missing zones in RAG systems (Chan, et al., 2024): Zone 1 (Incomplete Retrieval) occurs when relevant regulatory information exists in the knowledge base but is not retrieved into the context provided to the LLM, whereas Zone 2 (Incomplete Generation) occurs when the necessary information has been successfully retrieved and is present in the retrieved context, but the generation model fails to incorporate it into the final answer. Traditional evaluation metrics fail to distinguish between these failure modes, hindering diagnostic analysis and system improvement.

Consider a pilot asking, "What are the VFR minimums for Class B airspace?" A complete answer must address visibility requirements, cloud clearance rules, and potential day/night operational differences. A system achieving perfect faithfulness scores (all stated facts are accurate) could still deliver an incomplete answer if it retrieves only visibility requirements while missing cloud clearance rules, or if it retrieves all information but generates an answer addressing only some dimensions. Existing metrics would fail to detect either form of incompleteness, potentially leaving users with dangerously partial regulatory guidance.

2.4.5.2.2 Foundation: RAGAS framework and LLM-as-a-Judge methodology

Our evaluation framework builds upon established methodologies from RAGAS, a specialized framework designed for RAG system evaluation (Es, James, Espinosa-Anke, & Schockaert, 2024). RAGAS introduced several foundational principles that inform our approach: component-level decomposition that separates retrieval quality from generation quality, the LLM-as-a-Judge paradigm that uses LLMs to provide nuanced evaluation without extensive human annotation (Zheng, et al., 2023), and normalized scalar metrics in the [0,1] range that enable intuitive interpretation and cross-system comparison.

However, while RAGAS provides comprehensive evaluation through faithfulness and answer relevancy metrics, it does not explicitly address completeness in the presence of incomplete retrieval. RAGAS faithfulness measures if generated statements are supported by retrieved context, and answer relevancy assesses if responses address the posed question. Neither metric evaluates whether all required information dimensions have been retrieved and addressed. This gap is particularly critical for regulatory domains where comprehensive coverage of all applicable requirements, conditions, and exceptions is mandatory.

2.4.5.3 Intelligent question decomposition for completeness assessment

2.4.5.3.1 Core principle: dimensional analysis of information requirements

Our framework introduces a novel approach to completeness evaluation grounded in the recognition that complex regulatory questions inherently contain multiple information dimensions. Unlike simple factual queries that admit single-valued answers, regulatory questions typically require synthesis across multiple requirements, conditions, or specifications. For instance, “VFR weather minimums” encompasses visibility thresholds, cloud clearance distances (in three spatial dimensions: below, above, and horizontal), and potentially time-of-day variations. Each dimension represents an atomic information requirement that must be addressed for a complete answer.

Our key insight, inspired by query decomposition methods in recent coverage analysis work (Ammann, Golde, & Akbik, 2025; Chan, et al., 2024), is that completeness can be precisely quantified through dimensional decomposition. We decompose each complex question into atomic sub-questions: one per information dimension, where each sub-question targets a single verifiable fact. Completeness then becomes: *(number of dimensions addressed) / (total required dimensions)*. This formulation transforms the ambiguous concept of "completeness" into a measurable metric based on dimensional coverage.

2.4.5.3.2 Automated dimensional detection through cascading priority rules

To automate dimensional analysis, we developed a cascading priority system that applies pattern recognition and domain knowledge to detect information dimensions. The system processes questions through six priority levels, with earlier priorities taking precedence when multiple patterns match:

Priority 1 – Enumeration Detection: The system identifies explicit quantifiers in questions. When a question asks, "What are the TWO conditions..." or "Which THREE instruments...", the enumeration detector extracts the count and generates the specified number of sub-questions. This handles cases where regulatory text explicitly enumerates requirements.

Priority 1.5 – Conditional Pattern Recognition: This priority detects questions involving conditional logic or dual dimensions. For example, when a question mentions "BOTH overlapping AND non-overlapping sectors," the system recognizes two distinct dimensional contexts and generates separate sub-questions for each. This pattern commonly appears in regulations specifying different requirements for different operational scenarios.

Priority 2 – Comparison Analysis: When questions involve comparative language ("Compare Class A versus Class B" or "Difference between X and Y"), the system identifies the entities being compared and generates parallel sub-questions for each. This ensures comprehensive coverage when regulations specify different requirements for different categories.

Priority 3 – Range Detection: Questions asking about ranges ("What are the minimum and maximum values?") trigger generation of two sub-questions: one for the lower bound and one for the upper bound. This pattern frequently appears in regulatory specifications of acceptable parameter ranges.

Priority 4 – Conjunction with Domain Knowledge: When questions contain conjunctions (e.g. "and"), the system applies hard-coded aviation domain knowledge to interpret the dimensions

involved. For instance, a query asking about “visibility and cloud-clearance requirements under VFR minimums" contains a conjunction and triggers knowledge that VFR minimums involve both visibility requirements and cloud clearance rules (which themselves decompose into below/above/horizontal dimensions), typically yielding 2–4 sub-questions depending on question specificity.

Priority 5 – Known Multi-Dimensional Concepts: The system maintains a knowledge base of inherently multi-dimensional regulatory concepts. For example, "cloud clearance" automatically maps to three spatial dimensions (below clouds, above clouds, horizontal from clouds) as specified in 14 CFR § 91.155. This ensures comprehensive evaluation even when questions use compact terminology that masks underlying dimensional complexity.

2.4.5.3.3 LLM-Powered sub-question generation and refinement

Once the dimensional analysis identifies the number and nature of information dimensions, an LLM generates specific, well-formed sub-questions for each dimension. The LLM receives the detected dimensional structure along with the original question and produces atomic sub-questions that are: (1) independently answerable, (2) non-overlapping, and (3) collectively exhaustive of the original question's information requirements.

Generated sub-questions undergo three-stage post-processing to ensure quality. First, duplicate detection removes semantically identical sub-questions that may arise from overlapping patterns. Second, generic question filtering eliminates vague queries like "Are there any other requirements?" or "What else should be considered?" that fail to target specific factual dimensions. Third, meta-question filtering removes questions about the question itself ("Is X different from Y?" or "Are there exceptions to this rule?") that evaluate regulatory structure rather than retrieving specific requirements. This refinement ensures that each remaining sub-question represents a concrete, answerable atomic fact.

2.4.5.4 LLM-as-a-Judge evaluation system

2.4.5.4.1 Design principles from evaluation literature

Our judge system implements best practices from recent work on LLM-based evaluation (Zheng, et al., 2023). We employ binary judgment (YES/NO decisions) rather than Likert scales, as research shows binary classifications produce more reliable inter-judge agreement. Each judgment includes chain-of-thought reasoning where the LLM first articulates its analysis before rendering a decision, improving judgment accuracy and enabling post-hoc verification of reasoning quality. We set the temperature to 0.0 for deterministic consistency across repeated

evaluations of the same content. The system incorporates few-shot examples, demonstrating correct judgment patterns, which calibrates the LLM judge and improves accuracy on edge cases.

Critically, our judge emphasizes semantic equivalence over exact word matching. Regulatory answers may be phrased in multiple equivalent ways: "three statute miles visibility" and "visibility of 3 SM" convey identical information despite different surface forms. The LLM judge evaluates if the semantic content of an answer addresses a sub-question, not whether specific keywords appear.

2.4.5.4.2 Specialized reasoning mechanisms for regulatory content

To handle regulatory text nuances, we implemented two specialized reasoning capabilities that extend beyond standard LLM-as-a-Judge approaches:

Range Reasoning: When a sub-question asks, "What is the intensity at 15 degrees?" and the retrieved context states "10°-20°: 30 candles," the judge applies mathematical reasoning to recognize that 15 falls within the specified range [10, 20], and therefore the answer is present (30 candles). Without this capability, the judge would require exact string matching and fail to recognize range-based answers, a common pattern in regulatory specifications of graduated requirements.

Conditional Reasoning: When a sub-question asks, "Under what conditions can X operate?" and the context states "X requires: A, B, and C", the judge recognizes that the conditions are explicitly listed despite different phrasing. The judge understands that a list of requirements semantically answers a question about conditions, even when the exact phrase "under the conditions" does not appear. This prevents false negatives on semantically equivalent but syntactically different answer formulations.

These specialized mechanisms are essential for regulatory evaluation. Without range reasoning, the system would incorrectly mark as incomplete any answer that provides a range specification when asked about a specific value within that range. Without conditional reasoning, the system would fail to recognize explicitly enumerated requirements as satisfying "under what conditions" questions. Both would lead to systematic underestimation of retrieval and answer completeness in regulatory contexts.

2.4.5.5 LLM-as-a-Judge evaluation system

2.4.5.5.1 Metric definitions and calculations

Our framework evaluates RAG systems through four complementary metrics that collectively diagnose retrieval quality, generation quality, overall completeness, and faithfulness. Table 15 summarizes what each metric captures.

Table 15. Four complementary metrics for completeness evaluation

Metric	Definition
Retrieval Completeness (Zone 1)	Proportion of sub-questions whose answers are present in retrieved context. Measures whether the retrieval component found all necessary regulatory text.
Answer Completeness (Zone 2)	Among sub-questions answerable from retrieved context, proportion actually addressed in the generated answer. Measures generation efficiency.
Overall Completeness	Proportion of all sub-questions ultimately answered in the final response. Represents end-to-end completeness from the user perspective.
Faithfulness (Answer Precision)	Proportion of claims in the generated answer that are supported by retrieved evidence. Detects hallucinations and unsupported extrapolations.

Formally, the metrics are calculated as follows:

Metric 1 – Retrieval Completeness: *RC = (sub-questions in context) / (total sub-questions)*. This metric isolates retrieval performance by measuring what proportion of required information dimensions were successfully retrieved, independent of if the generation component used them.

Metric 2 – Answer Completeness: *AC = (sub-questions in answer) / (sub-questions in context)*. This metric evaluates generation efficiency by measuring what proportion of available information (present in retrieved context) was actually incorporated into the answer. Note that the denominator is context count, not total count, this specifically targets Zone 2 (generation incompleteness).

Metric 3 – Overall Completeness: *OC = (sub-questions in answer) / (total sub-questions)*. This represents the end-to-end user experience: what proportion of their information needs were ultimately satisfied. Mathematically, $OC = RC \times AC$, reflecting the compound nature of completeness across both pipeline stages.

Metric 4 – Faithfulness: *F = (supported claims) / (total claims)*. Following the FActScore methodology (Min, et al., 2023), we decompose generated answers into atomic factual claims

and verify each against retrieved context. This detects hallucinations, where the system fabricates information not present in regulatory text.

2.4.5.5.2 Diagnostic power: distinguishing failure modes

The separation of Metrics 1 and 2 enables precise diagnosis of system failure modes. Consider two scenarios, both achieving 67% Overall Completeness:

Scenario A: *RC = 67%, AC = 100%, OC = 67%.* The retrieval component failed to find all relevant regulatory text (Zone 1 problem), but the generation component successfully used all available information. Solution: Improve retrieval (expand search, better query formulation, add graph traversal).

Scenario B: *RC = 100%, AC = 67%, OC = 67%.* Retrieval was perfect, but the generation component omitted information present in context (Zone 2 problem). Solution: Improve generation (better prompting, explicit instructions to cover all context, increase context window).

Traditional metrics would report identical overall scores for both scenarios, masking the fundamentally different underlying problems and leading to inappropriate remediation strategies. Our decomposed metrics enable targeted system improvement.

2.4.5.6 Demonstration: Complete Evaluation Workflow

Figure 20 illustrates the complete evaluation workflow on a real regulatory query: "What are the VFR minimums for Class B airspace?" This example demonstrates how our framework decomposes a seemingly simple question into multiple information dimensions, evaluates each dimension independently, and aggregates results into diagnostic metrics.

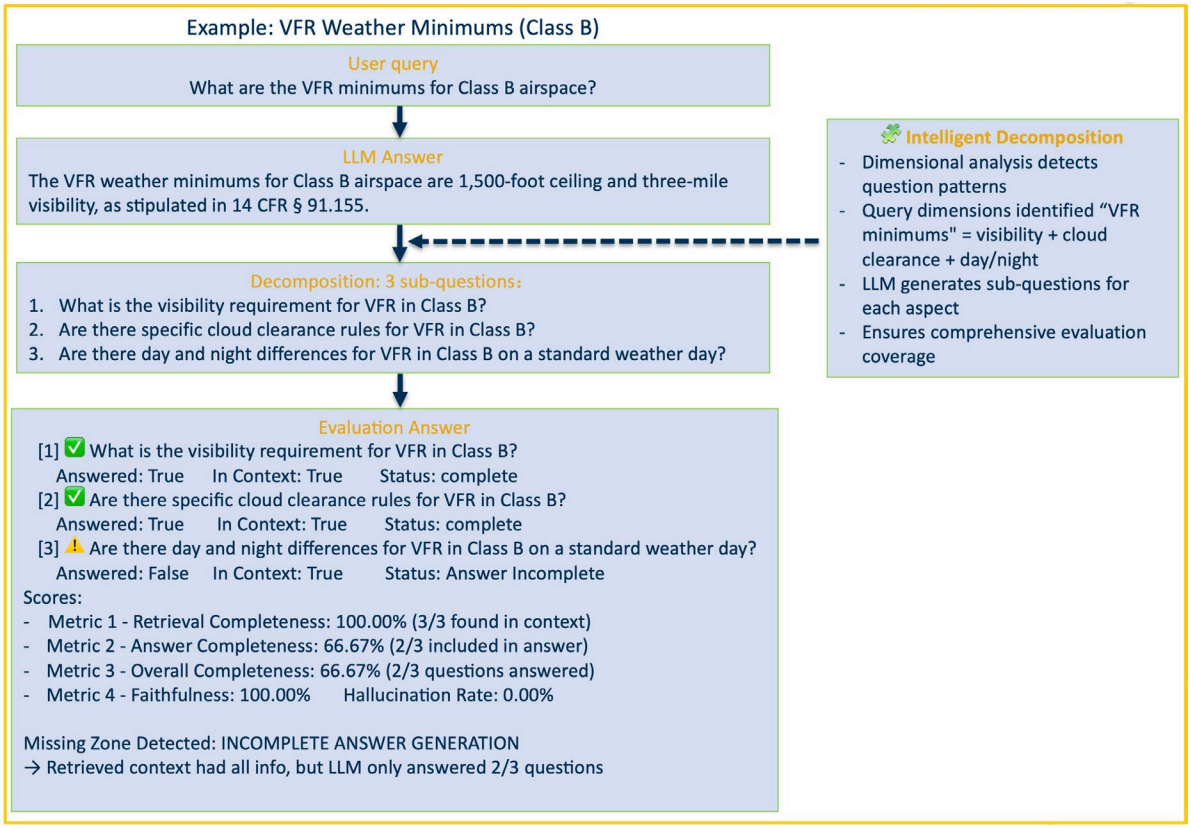

Figure 20. Complete Evaluation Workflow for VFR Weather Minimums Query

Question Decomposition Phase: The dimensional analysis identifies that "VFR minimums" encompasses three distinct information dimensions: (1) visibility requirements, (2) cloud clearance rules, and (3) potential day/night operational differences. Domain knowledge priority rules trigger on "VFR minimums" to activate this decomposition. The LLM generates three specific sub-questions targeting each dimension.

RAG System Response: The RAG system retrieves context from 14 CFR § 91.155 and generates: "The VFR weather minimums for Class B airspace are 1,500-foot ceiling and three-mile visibility, as stipulated in 14 CFR § 91.155." This answer addresses visibility and cloud clearance but omits day/night considerations.

LLM-as-a-Judge Evaluation: The judge evaluates each sub-question:

(1) Visibility requirement → Found in context: YES, Found in answer: YES, Status: Complete.
(2) Cloud clearance rules → Found in context: YES, Found in answer: YES, Status: Complete.
(3) Day/night differences → Found in context: YES, Found in answer: NO, Status: Answer Incomplete.

Metrics Calculation: Retrieval Completeness = 3/3 = 100.0% (all information was retrieved). Answer Completeness = 2/3 = 66.7% (only 2 of 3 available pieces were used). Overall Completeness = 2/3 = 66.7% (user received 2 of 3 required pieces). Faithfulness = 100.0% (everything stated was accurate, no hallucinations).

Diagnostic Insight: The system exhibits a Zone 2 (incomplete generation) problem. Retrieval was perfect, but the answer generation component failed to address all dimensions present in the retrieved context. This incomplete answer could mislead pilots about whether Class B VFR minimums vary between day and night operations – a potentially safety-critical omission. Traditional keyword-based or BLEU-style metrics would miss this incompleteness, since the answer contains accurate, relevant information. Only dimensional decomposition reveals the missing information dimension.

2.4.6 Summary of findings

This section presented a comprehensive RAG system for CFR regulatory question answering that addresses the limitations of traditional keyword-based search through semantic understanding and flexible retrieval. We demonstrated a hybrid architecture combining direct citation lookup for explicit references with semantic vector search for natural language queries, achieving effective retrieval across diverse query types, from simple fact lookups to complex conceptual questions. Our novel summary-enhanced chunking strategy significantly improved retrieval precision by embedding conceptually dense summaries while preserving original text for answer generation, successfully handling semi-structured regulatory content such as tables and multi-conditional rules. We developed a completeness-focused evaluation framework that addresses the "missing zone problem" through intelligent question decomposition and LLM-as-a-Judge methodology. The framework employs priority-based dimensional analysis to automatically decompose complex regulatory questions into atomic sub-questions, enabling precise measurement of information coverage. We implemented four complementary metrics (Retrieval Completeness, Answer Completeness, Overall Completeness, and Faithfulness) that separately evaluate retrieval quality and generation quality, providing granular diagnostic capability to pinpoint system failure modes. The framework was validated on a carefully constructed 220-question dataset spanning seven question types, demonstrating its ability to detect incomplete answers that traditional metrics would miss.

Future work should leverage this evaluation framework to systematically compare KG-based RAG against pure semantic search RAG, testing the hypothesis that graph traversal improves cross-reference resolution and multi-hop reasoning. Using Retrieval Completeness (Metric 1) as the primary measure to isolate retrieval quality, question-type stratification (CROSS-

REFERENCE, MULTI-HOP, FACTUAL) will identify where KG-RAG provides the most value and quantify performance advantages for specific use cases. Our dimensional decomposition approach enables fair, comprehensive comparison by ensuring both systems are evaluated against identical information requirements, providing evidence-based insights for KG-RAG investment decisions in regulatory question answering contexts.

2.5 Extending the Daedalus multi-agent RAG system: migration to local infrastructure and integration of MCP

The Daedalus multi-agent RAG system, developed by the FAA team for regulatory question answering, originally relied on OpenAI's API for both language model inference and embedding generation. To enable independent research and development without external API dependencies, we undertook a comprehensive modification effort comprising two complementary contributions. First, we migrated the entire system to local Ollama infrastructure, replacing OpenAI models with open-source alternatives (Llama 3.1 for inference, nomic-embed-text for embeddings) while maintaining comparable performance. Second, we integrated the MCP to establish a modular, extensible architecture, demonstrated through containerization of the VectorSearch component using FastMCP and Docker. This section presents the technical implementation of each extension, demonstrates their practical benefits through concrete examples, and discusses how the MCP-based architecture positions the system for future integration of KGs and multi-source data.

2.5.1 Introduction to the FAA Daedalus system

The Daedalus system is a multi-agent RAG framework developed by the FAA for regulatory question answering. The system implements a Chain of Responsibility design pattern with three specialized agents orchestrated through a central coordinator (Pydantic AI, 2025):

Agent Architecture:

- **QueryAnalyzer**: Classifies query intent (lookup, explanation, compliance, comparison), extracts search terms, and determines optimal search strategy (semantic, direct, or hybrid)
- **SearchAgent**: Executes hybrid retrieval, combining semantic search with keyword-based direct lookup across 18,814 regulatory text chunks
- **ResponseSynthesisAgent**: Synthesizes coherent answers with proper CFR citations, caveats for ambiguous information, and relevant contextual details

Figure 21 illustrates this three-agent chain, showing the sequential flow from query classification through retrieval to response generation. The system leverages the Pydantic-AI framework (Pydantic AI, 2025) for agent implementation, which provides type-safe validation, structured output handling, and seamless integration with various LLM providers. Pydantic-AI brings the "FastAPI feeling" (Pydantic AI, 2025) to agent development through decorator-based tool definitions, automatic parameter validation, and comprehensive observability support.

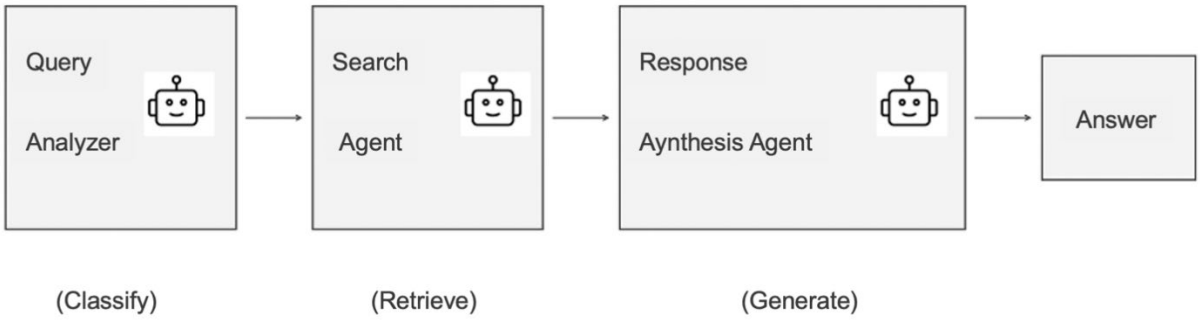

Figure 21. Daedalus Basic Agent Chain

2.5.2 Migration to local Ollama infrastructure

2.5.2.1 Motivation

The original Daedalus implementation, as shared by the FAA, relied on OpenAI's API for both language model inference and embedding generation. The system used text-embedding-ada-002 for generating 1536-dimensional embeddings and generative pre-trained transformer (GPT) models for agent reasoning (the original configuration specifies openai:gpt-5-nano as the default LLM model).

For our research and development work, we needed to run the system without access to OpenAI API credentials. Additionally, migrating to a fully local infrastructure offered several advantages for our use case:

- **Independent Development**: Ability to experiment and iterate without API access requirements
- **Data Sovereignty**: All regulatory queries and processing remain on local infrastructure
- **Reproducibility**: Consistent behavior independent of external service changes

Therefore, we migrated the entire system to local Ollama infrastructure, demonstrating that open-source models can effectively support domain-specific RAG applications.

2.5.2.2 Implementation

We migrated the entire Daedalus stack to local Ollama infrastructure, which provides optimized local inference for open-source LLMs through llama.cpp's efficient implementation with quantization support (Ollama, 2025). The migration involved two key components:

Language Model Migration: All Pydantic-AI agents now use Llama 3.1 (8B parameter model) running on Ollama instead of GPT models. The transition was facilitated by Pydantic-AI's provider abstraction layer, which allows specification of alternative API endpoints while maintaining the same agent interface. By configuring the OpenAI-compatible provider to point to Ollama's local endpoint, we achieved seamless integration without modifying agent logic.

Embedding Model Migration: The system now uses nomic-embed-text for generating 768-dimensional embeddings instead of OpenAI's ada-002 (1536 dimensions) (Nussbaum, Morris, Duderstadt, & Mulyar, 2024). The migration required:

- Modifying embedding generation to call Ollama's **/api/embeddings** endpoint
- Recreating the vector database with 768-dimensional embeddings (Douze, et al., 2024)for all 18,814 regulatory chunks
- Updating SQLite schema from embedding FLOAT[1536] to embedding FLOAT[768]

Configuration Management: The system reads embedding and LLM configuration from environment variables, enabling easy provider switching. Key configuration parameters include the LLM model (Llama 3.1), embedding model (nomic-embed-text), embedding provider (ollama), and the Ollama base URL.

2.5.3 Model Context Protocol: architecture for extensibility

2.5.3.1 What is MCP?

MCP is an open standard introduced by Anthropic in November 2024 for connecting AI assistants to external data sources and tools (Anthropic, 2024). MCP addresses the "N×M integration problem" where N AI applications each require M custom connectors to access M data sources, creating unsustainable complexity (Anthropic, 2024; David, 2024).

MCP establishes a client-server architecture using JSON-RPC 2.0 for communication, deliberately reusing message-flow patterns from the Language Server Protocol (LSP) (David,

2024). The protocol defines several core primitives (Anthropic, 2024; Model Context Protocol Contributors, 2025):

- **Resources**: Structured data (documents, database records) that enrich model context
- **Tools**: Executable functions the model can invoke (database queries, API calls, searches)
- **Prompts**: Prepared instruction templates guiding model behavior
- **Roots**: Entry points into file systems or environments

MCP can be thought of as "USB-C for AI applications" (Anthropic, 2024) – just as USB-C provides a standardized connection for devices, MCP provides a standardized way to connect AI models to diverse data sources and tools.

MCP vs. Function Calling: While OpenAI's function calling (introduced 2023) and the ChatGPT plugin framework solve similar problems, they require vendor-specific connectors. MCP's key advantage is vendor neutrality and growing ecosystem support – OpenAI officially adopted MCP in March 2025, followed by Google DeepMind (David, 2024).

2.5.3.2 Why MCP for Daedalus?

Integrating MCP into the Daedalus architecture provides four strategic benefits:

- **Modularity** Traditional direct coupling between agents and data sources creates tight dependencies, as shown in Figure 22.

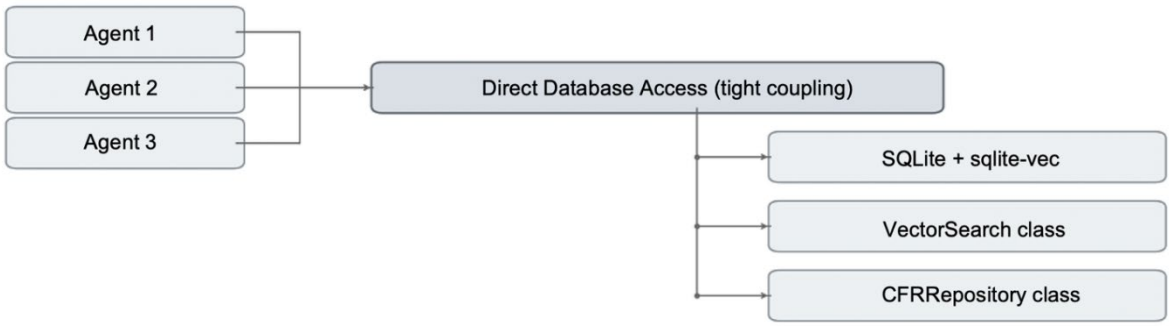

Figure 22. Before MCP

With MCP, agents access data through a standardized interface, enabling seamless backend swaps. This is illustrated in Figure 23.

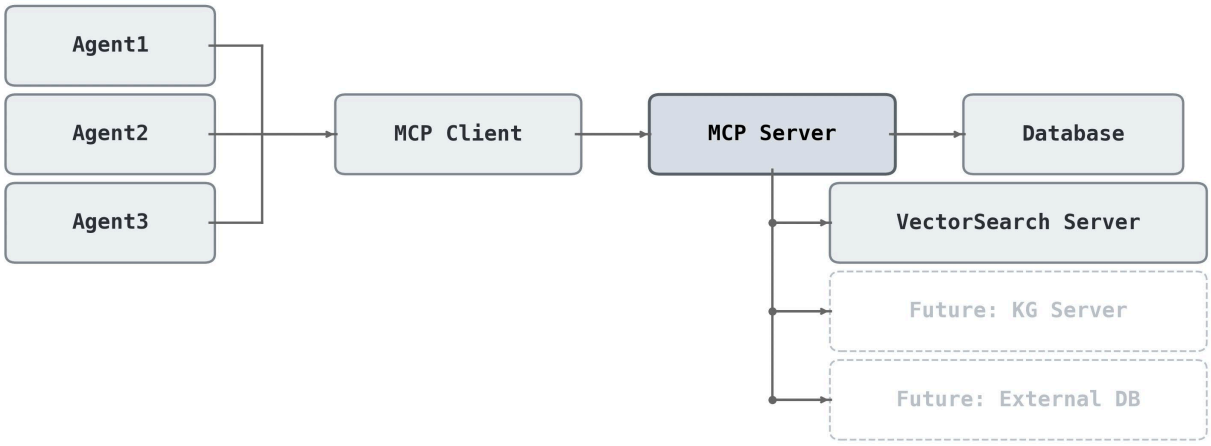

Figure 23. After MCP

The VectorSearch implementation can be swapped (SQLite → PostgreSQL → Pinecone) without modifying agent code, as long as the MCP interface remains consistent.

- **Extensibility** (Critical for Future Work) Our research roadmap includes integrating multiple external knowledge sources:
    - **Knowledge Graphs**: Entity relationships extracted from FAA regulations
    - **FAA Advisory Circulars**: Supplementary guidance documents
    - **ASRS/NTSB Databases**: Accident and incident reports

Each data source can be deployed as an independent MCP server. Agents automatically discover newly available resources and tools through MCP's discovery mechanism, requiring zero code changes in the agent layer (Anthropic, 2024; Model Context Protocol Contributors, 2025).

- **Security and Governance** MCP provides structured access control through explicit tool definitions (Anthropic, 2024; Model Context Protocol Contributors, 2025). Agents can only perform operations exposed by the MCP server, preventing:
    - Arbitrary database queries
    - Unauthorized data access
    - SQL injection vulnerabilities

This is particularly critical for government applications where access control and audit trails are mandatory.

- **Interoperability** The same regulatory data MCP server can be accessed by multiple AI applications beyond Daedalus:
    - Compliance checking tools
    - Training simulators
    - Automated report generation systems
    - Third-party analysis tools

This enables shared infrastructure across an organization without duplicating integration work (Anthropic, 2024).

2.5.3.3 Model Context Protocol in the broader ecosystem

As of early 2025, MCP has been rapidly adopted across the AI industry (David, 2024):

- **LLM Providers**: OpenAI, Google DeepMind, Anthropic
- **Development Tools**: Zed, Replit, Codeium, Sourcegraph, Cursor
- **Enterprise Systems**: Pre-built MCP servers available for Google Drive, Slack, GitHub, Postgres, and others (Anthropic, 2024; Model Context Protocol Contributors, 2025)

This growing ecosystem validates MCP as a de facto standard for AI-data integration, ensuring long-term compatibility and community support.

2.5.4 Model Context Protocol practice: containerized VectorSearch

2.5.4.1 Motivation

To validate MCP's practical benefits and establish patterns for future knowledge source integration, we implemented VectorSearch as a containerized MCP server. This serves three purposes:

- **Proof of Concept**: Demonstrate MCP integration in a real production component
- **Infrastructure Template**: Establish patterns applicable to KG and external database servers
- **Immediate Benefit**: Isolate VectorSearch dependencies and enable cross-platform deployment

2.5.4.2 Design decision: bypass Pydantic-AI

A critical architectural choice was to implement the MCP server outside the Pydantic-AI agent framework. While Pydantic-AI provides excellent abstraction for LLM-based agents, the VectorSearch component is a pure data retrieval service requiring no LLM reasoning. Integrating it through MCP provides:

- **Direct access**: VectorSearch.search() called directly without agent orchestration overhead
- **Simplified deployment**: Container runs standalone without Pydantic-AI dependencies
- **Clear separation**: Data layer decoupled from agent reasoning layer
- **Performance**: No LLM inference in the retrieval path

This design exemplifies MCP's flexibility – not all system components need to be LLM agents.

2.5.4.3 Implementation

Architecture: the basic architecture of how we connect Daedalus to the containerized database via MCP is shown in Figure 21.

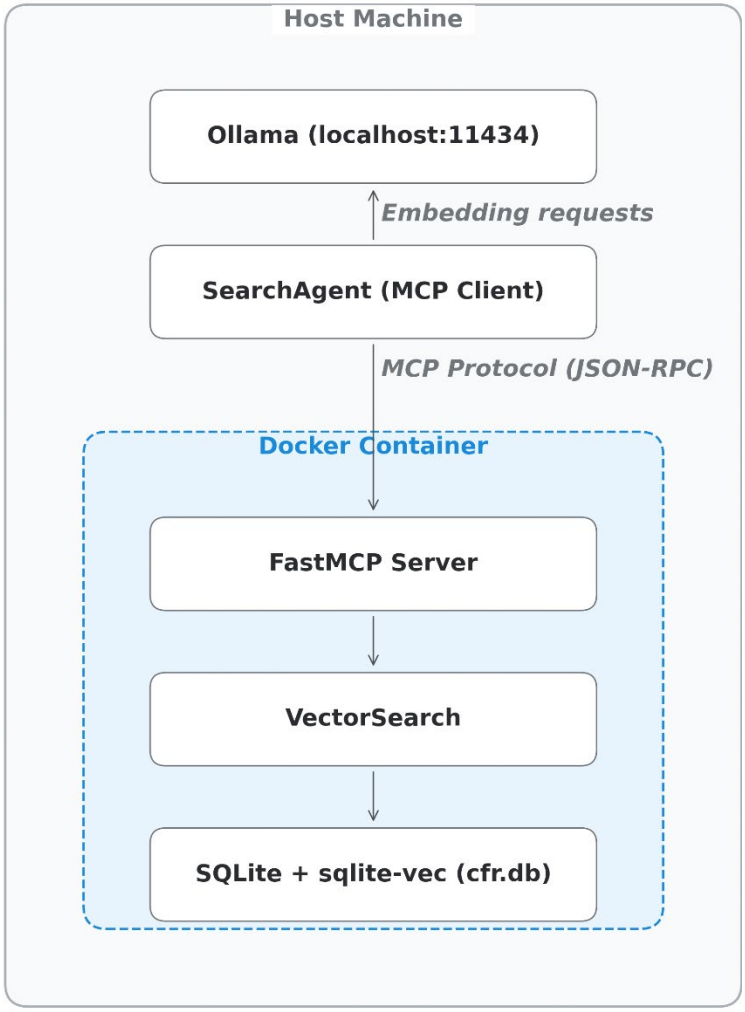

Figure 24. VectorSearch Container Architecture

Component 1: FastMCP Server: We use FastMCP (Lowin, 2025; Prefect, 2025), a high-level Python framework that simplifies MCP server development. FastMCP pioneered Python MCP development and was incorporated into the official MCP SDK in 2024 (Prefect, 2025; Model Context Protocol Contributors, 2025). FastMCP 2.0 provides production-ready features, including enterprise authentication, deployment tools, and testing utilities (Prefect, 2025).

FastMCP's decorator-based API simplifies tool definition significantly. For example, the VectorSearch functionality is exposed as an MCP tool using a simple decorator pattern (`@mcp.tool`). The tool function accepts a query string and top-k parameter, directly invokes the `VectorSearch.search()` method without Pydantic-AI overhead, and returns results in a structured dictionary format containing the regulatory text, section numbers, and similarity scores.

FastMCP automatically handles JSON-RPC protocol details, validates parameters using Pydantic schemas, provides error handling and retry logic, and exposes tool discovery endpoints for MCP clients (Lowin, 2025; Prefect, 2025).

Component 2: Docker Container: The Docker container encapsulates a complete VectorSearch environment including Python 3.11 runtime, FastMCP framework and dependencies, sqlite-vec compiled from source (required for ARM64 compatibility) (Garcia, 2025), and the CFR regulatory database (cfr.db, 298 MB, managed via Git LFS). The containerization strategy involves building sqlite-vec from source to support both ARM64 and x86_64 architectures, installing FastMCP and dependencies, copying the VectorSearch implementation and regulatory database, and exposing the MCP server endpoint.

Component 3: MCP Client Integration: The SearchAgent integrates with the containerized VectorSearch through the MCP. During initialization, the agent creates an MCP client configured to communicate with a Docker container named *daedalus-vector*. The client uses Docker's exec mechanism to invoke the MCP server inside the container. For each search operation, the agent asynchronously calls the *search_cfr* tool through the MCP protocol, passing the user's query and requesting the top-100 most relevant results. This architecture decouples the agent from the underlying database implementation, enabling modularity and independent scaling. The algorithm below presents a pseudocode for this integration.

Algorithm 1 SearchAgent MCP Integration

```
procedure INITIALIZE( )
    client ← MCPClient(command = "docker", args = [...])
end procedure
procedure SEARCH(query)
    async with client as connection:
        result ← connection.call_tool("search_cfr", {query, top_k=100})
    return parse_results(result)
end procedure
```

2.5.4.4 Technical highlights

- **Cross-Platform Support:** The container supports both ARM64 (Apple Silicon) and x86_64 architectures through multi-stage Docker builds and source compilation of sqlite-vec, which lacks pre-built ARM64 binaries.
- **Ollama Integration:** Embedding generation remains on the host via `host.docker.internal:11434`, avoiding the overhead of packaging Ollama models in the container.

- **Read-Only Database:** The cfr.db file is mounted read-only (`data:/app/data:ro`), preventing accidental modifications and supporting multiple container instances sharing the same database.
- **Template for Future Work**: The FastMCP + Docker pattern established here directly applies to KG and external database servers.
- **Easy Scaling**: Additional vector database instances can be deployed for load balancing.
- **Technology Flexibility**: Migrating to Pinecone/Weaviate/Qdrant requires only changing the MCP server implementation, not client code.

2.5.5 Summary of findings

This work demonstrates two complementary changes to the Daedalus multi-agent RAG system. First, we achieved complete data sovereignty through local infrastructure migration, replacing OpenAI API dependencies with Ollama while maintaining comparable performance. Second, our adoption of MCP establishes a modular, extensible architecture validated through VectorSearch containerization, preparing the system for knowledge graph and multi-source integration.

Together, these changes transform Daedalus from a demonstration system into a production-ready platform suitable for government and enterprise deployment. The MCP-based architecture positions the system for future expansion while maintaining security, modularity, and clear architectural boundaries. Our work validates MCP as a practical solution for AI system extensibility and provides concrete patterns that can be adapted for other RAG applications requiring multi-source integration and rigorous quality evaluation.

2.5.5.1 Future work

While our MCP framework establishes the technical foundation for KG integration, a more comprehensive investigation is needed to determine when KG augmentation provides measurable benefits versus when it represents unnecessary complexity. Leveraging our completeness evaluation framework and MCP's modular architecture, future work should implement an evaluation agent that systematically compares KG-RAG against pure semantic search RAG across diverse query types. Using Retrieval Completeness as the primary measure to isolate retrieval quality, question-type stratification will identify where KG provides the most value and quantify performance advantages for specific regulatory use cases. The MCP-based evaluation agent can be deployed as an independent service, enabling repeatable, scalable comparative analysis to provide evidence-based guidance on when the added infrastructure complexity of KG-RAG is justified by measurable gains in answer completeness.

3 Software tool documentation

3.1 Retrieval-augmented generation Completeness Evaluation Framework

3.1.1 Overview

The RAG Completeness Evaluation Framework (rag-evaluator) is a comprehensive software tool for assessing RAG systems. It addresses two critical missing zones in traditional RAG evaluation: incomplete retrieval and incomplete answer generation. The framework uses LLM-as-a-Judge methodology with intelligent question decomposition to measure four key metrics: Retrieval Completeness, Answer Completeness, Overall Completeness, and Faithfulness.

Key Features:

- **Intelligent Question Decomposition:** Automatically decomposes complex questions into atomic sub-questions using dimensional analysis (enumeration, conditional, range, comparison detection)
- **Four-Metric Evaluation:** Comprehensive assessment of both retrieval and generation quality
- **Docker-Ready Deployment:** Simple one-command deployment
- **Easy Integration:** Clean Python API for seamless RAG system integration
- **220 Pre-built Questions:** Curated test bank from FAA regulations (14 CFR)

3.1.2 System architecture

The framework follows a modular architecture organized into the following directory structure:

src/

- evaluation_api.py: Main evaluation API
- question_decomposer.py: Intelligent question decomposition
- completeness_judge.py: LLM-based completeness judge

data/

- evaluation_questions.json: 220 FAA CFR test questions

QuestionSelection/

- generate_test_questions.py: Test question generator

examples/

- simple_evaluation.py: Complete usage example

docker-compose.yml: Docker deployment configuration

Dockerfile: Container build specification

requirements.txt: Python dependencies

3.1.3 Installation and deployment

Prerequisites:

- Ollama (running on your host machine): Visit https://ollama.ai for installation instructions (macOS/Linux/Windows). Start the Ollama service using "ollama serve" and pull the evaluation model with "ollama pull Llama3.1:latest".
- Docker: Ensure Docker Desktop or Docker Engine is installed and running.

Installation Steps:

1. Clone the repository and navigate to the project directory.
2. Start the evaluation container: docker-compose up -d
3. Generate test questions (e.g., 20 random questions): docker exec -it rag-evaluator python QuestionSelection/generate_test_questions.py 20

4. Run the evaluation example: docker exec -it rag-evaluator python examples/simple_evaluation.py
5. View the detailed report: cat outputs/evaluation_report_*.txt

3.1.4 Usage

3.1.4.1 Basic API usage

To evaluate your RAG system, use the following Python code:

```
from src import evaluate_rag

# Prepare your RAG outputs
results = evaluate_rag(
    question_ids=["14-29-29.1401_Q3", "14-91-91.155_Q1"],
    retrieved_contexts={
        "14-29-29.1401_Q3": "your retrieved context",
        "14-91-91.155_Q1": "your retrieved context"
    },
    generated_answers={
        "14-29-29.1401_Q3": "your generated answer",
        "14-91-91.155_Q1": "your generated answer"
    },
    output_dir='./outputs'
)

# View metrics
print(f"Overall Completeness: {results['summary']['avg_overall_completeness']:.1%}")
print(f"Faithfulness: {results['summary']['avg_faithfulness']:.1%}")
```

3.1.4.2 Integration with your RAG system

To integrate the evaluation framework with your own RAG system, modify the examples/simple_evaluation.py file by replacing the my_rag_system function:

```
def my_rag_system(question):
    # Replace with your RAG implementation
    context = your_retriever.retrieve(question)
    answer = your_generator.generate(question, context)
    return context, answer
```

3.1.5 Evaluation metrics

Metric 1 - Retrieval Completeness: Percentage of sub-questions answered by retrieved context. Measures whether all required information was successfully retrieved.

Metric 2 - Answer Completeness: Percentage of sub-questions answered by the generated answer. Measures how well the generation component used available information.

Metric 3 - Overall Completeness: Combined metric calculated as Retrieval Completeness × Answer Completeness. Represents end-to-end question satisfaction.

Metric 4 - Faithfulness: Percentage of answer content supported by retrieved context. Detects hallucinations and unsupported claims.

3.1.6 Example output

The evaluation framework produces structured output as follows:

```
======================================================================
Question: What are the permissible flash frequency ranges for
both overlapping and non-overlapping sectors?
Sub-questions Generated: 2
  [1] What are the ranges for overlapping sectors?
  [2] What are the ranges for non-overlapping sectors?
Evaluation Results:
  Retrieval Completeness: 100.00%
  Answer Completeness:    100.00%
  Overall Completeness:   100.00%
  Faithfulness:           100.00%

======================================================================
```

3.1.7 System configuration

The Ollama API endpoint is automatically configured via environment variables. When running inside Docker, the system uses http://host.docker.internal:11434/api/generate to access the host Ollama service. For local development outside Docker, it defaults to

http://localhost:11434/api/generate. The system auto-detects the environment and uses the appropriate endpoint.

3.1.8 How the framework works

3.1.8.1 Question decomposition

The framework uses dimensional analysis to intelligently break down complex questions:

- Enumeration: "What are the two conditions?" generates 2 sub-questions
- Conditional: "Under what conditions can X operate?" preserves all conditions
- Range: "What are the minimum and maximum values?" generates 2 sub-questions
- Comparison: "Compare Class A vs Class B" generates 2 sub-questions per entity

3.1.8.2 LLM-as-a-Judge evaluation

For each sub-question, the judge performs the following checks:

- Does the retrieved context contain the answer?
- Does the generated answer contain the answer?
- Provides reasoning for each judgment

3.1.8.3 Metric calculation

Retrieval Completeness = (Sub-questions in context) / (Total sub-questions)

Answer Completeness = (Sub-questions in answer) / (Sub-questions in context)

Overall Completeness = (Sub-questions in answer) / (Total sub-questions)

Faithfulness = (Answer claims supported by context) / (Total answer claims)

3.1.9 System requirements

Host System: Ollama with Llama3.1:latest model installed

Docker: 4GB RAM recommended

Storage: 20GB+ for Docker images, models, and logs

3.1.10 Troubleshooting

3.1.10.1 Ollama connection issues

To verify Ollama is running on the host:

curl http://localhost:11434/api/tags

To test connectivity from within the Docker container:

docker exec -it rag-evaluator curl http://host.docker.internal:11434/api/tags

3.1.11 Important notes

LLM-as-a-Judge Stability: Evaluation quality depends on LLM performance. For production use, consider using more capable models (e.g., GPT-4, Claude) for more consistent judgments, in place of the default Llama3.1:latest model.

Question Bank: The framework includes 220 test questions covering seven types: FACTUAL, CONDITIONAL, CONCEPTUAL, MULTI-HOP, CROSS-REFERENCE, TABULAR, and PILOT EXAM.

Docker Setup: Container uses Python 3.10 with all dependencies pre-installed.

3.2 Container for KG-based QA

To make the KG easy to run and share across teams, we maintain a containerized deployment that bundles the Neo4j graph database together with a lightweight QA interface. The container approach supports repeatable setup, consistent dependencies, and a clear separation between the data layer (the CFR KG) and the QA layer (the API and LLM integration).

Key Features

The container package contains three key components:

- **Neo4j KG (Title 14 CFR**) loaded from our RDF Turtle export. The latest KG export for demos and development is title_14_Chapter1_with_text_with_refs.ttl, which includes the regulation hierarchy, node text and cross-references.
- **Initialization script** (neo4j-init.cypher) that loads the TTL into Neo4j at startup.
- **FASTAPI service** (app.py) that connects to Neo4j and exposes a /query endpoint for question and answering workflows.

The repository also includes a simple client (llm_with_faa_kg.py) that demonstrates LLM-assisted querying of the graph using LangChain's GraphCypherQAChain, with a local LLM served via Ollama.

3.2.1 Quick workflow

The basic workflow is:

1. Build the image.

   ```
   docker build -t faa-cfr-kg .
   ```

2. Run the container.

   ```
   docker run -p 7474:7474 -p 7687:7687 -p 8081:8081 faa-cfr-kg
   ```

   This exposes:

- Neo4j Browser at http://localhost:7474 (Login with neo4j / supertest)
- API endpoint at http://localhost:8081/query

3. Run an LLM client locally.

   If using Ollama, start a model locally and run the provided client:

   ```
   ollama run Llama3

   python llm_with_faa_kg.py
   ```

When a question is asked, the client uses GraphCypherQAChain to generate Cypher from the natural language query, execute it against Neo4j, and summarize results.

4 Future work

This section outlines several promising avenues for advancing Regulation-KG, building upon the foundation detailed throughout this document. These directions include enhancing cross-referencing mechanisms within the KG, integrating additional regulatory guidance such as Advisory Circulars, and developing time-aware features to better track regulation changes. With

these improvements, Regulation-KG can become a more robust and comprehensive tool for regulatory analysis and other down-stream uses.

4.1 Detailed cross-referencing

The current JSON schema and the resulting KG capture cross-references primarily at the section level by detecting the § (silcrow) symbol and applying regular-expression rules. Additional work was done to extend this capability by using an LLM-based extraction method that identified more complex, context-dependent cross-references and incorporated them into the JSON. The goal of this effort was to produce a more detailed cross-reference layer that extends beyond section-to-section links and can be used to build a more richly connected KG.

While the LLM-based method demonstrated strong accuracy in detecting and extracting complex cross-references from regulatory context, practical challenges emerged when attempting to operationalize these outputs. Specifically, extracted references must map cleanly to existing node identifiers in the JSON to be usable for graph linking. In many cases, the extracted text is correct in meaning but does not exactly match the identifier format already present in the schema, which prevents reliable, automated resolution.

Future work will focus on improving the end-to-end extraction and resolution mechanism so that cross-references can be rewritten into forms that exactly match existing node identifiers. One direction is a semi-semantic matching layer on top of the tested LLM extraction method. In this approach, each extracted reference is compared against the set of valid identifiers in the JSON using both schema constraints and local context from the regulation text. Once the best-matching identifier is selected, the reference is rewritten in the JSON using that exact identifier format. This enhanced, detailed cross-referencing is expected to increase KG connectivity and improve accuracy and traceability in downstream RAG workflows. It will also enable more consistent traversal and provenance across linked regulatory nodes.

4.2 Advisory Circular integration

The Advisory Circular integration work so far has focused on building a lightweight, reproducible document-ingestion pipeline that can attach AC guidance to the existing Regulation-KG without changing the CFR hierarchy. We implemented a PDF extractor that pulls core cover-page metadata (e.g., docId, title/subject, issuedOn, initiatedBy) and scans the full document text to extract a normalized list of referenced CFR/FAR sections. Those references are then used to generate an append-only RDF/Turtle patch that creates an AC node and connects it to the relevant CFR Section/Subsection/Subsubsection nodes using the existing cross-reference relationship (refersTo). This provides a proof-of-concept “document layer” that makes regulatory

content more navigable and investigatory: users can start from a regulation and discover associated guidance documents or start from an AC and see all the regulatory provisions it touches, enabling richer context for compliance interpretation and safety barrier reasoning.

Future work will focus on moving from a patch-only workflow into a robust, scalable integration strategy across many ACs and other guidance documents. Key next steps include improving citation resolution (e.g., reliably linking paragraph-level references like 91.409(f)(4) to the most specific node available), adding provenance/evidence (page numbers, snippets, extraction method, confidence) so each AC-to-CFR link is auditable, and modeling internal AC structure (chapters/sections/paragraphs) to support fine-grained retrieval and show the exact guidance text relevant to the CFR clause. At scale, the same pipeline can be extended to additional document types (e.g., Orders, SAFOs, ADs) and paired with a hybrid retrieval index for full-text search, enabling Regulation-KG to function as a more complete oversight and investigation tool that links requirements to practical guidance, implementation expectations, and traceable supporting evidence.

4.3 Study regulation changes over time

Studying regulation changes over time is essential, because regulatory text is not static. Requirements evolve through clarifications, new constraints, exceptions, and reorganizations that can materially change compliance responsibilities. A time-aware view supports accurate interpretation by answering not only what a rule says, but also what changed, when changes became effective, and how the controlling language differed across periods. This traceability is especially important for incident and accident investigations, as well as retrospective analyses, where the correct answer depends on the regulatory state on a specific historical date rather than the latest version.

A practical implementation path for future work is interval-based versioning within the existing JSON hierarchy. Each regulation node can keep the same identifier and children while adding a “versions” field that stores multiple text variants along with “effective from” and “effective to” dates. During retrieval, an “as-of” date can be used to select the version that was in force at that time, so the system grounds answers in the appropriate regulatory language without needing separate full copies of the hierarchy. This keeps the structure consistent while making time an explicit part of how regulatory text is stored and accessed.

An alternative implementation path is a temporal KG-RAG design that keeps canonical regulation nodes stable while attaching time-stamped version nodes that hold the text for different periods. In this design, retrieval can bring back not only the relevant passage but also

surrounding context that was valid at the same time, such as linked definitions or cross-referenced requirements. This approach also provides a clearer way to handle structural edits, such as renumbering or moved paragraphs, by connecting older and newer versions through explicit relationships.

Integrating temporal regulation modeling can strengthen both correctness and interpretability of KG-RAG outputs. Time-scoped retrieval would reduce the risk of grounding answers in the wrong regulatory era, enable "what changed between dates" analyses, and improve traceability by linking responses to the controlling version of the text.

4.4 Incorporation of more data formats

Incorporating additional data formats beyond plain text is important, because regulatory meaning is often distributed across multiple representational forms. Many requirements are partially encoded in tables, figures, diagrams, charts, or structured appendices rather than fully described in narrative text. When these elements are excluded, retrieval systems may rely on incomplete textual summaries, leading to imprecise grounding or partial answers. A multi-format representation supports richer reasoning by allowing the system to interpret regulatory intent from both descriptive language and structured visual artifacts.

A practical implementation path for future work is to augment the existing KG with auxiliary nodes that store optical character recognition (OCR)-extracted tables and figures (Shi, Bai, & Yao, 2017). Using document layout analysis and OCR, structured content such as compliance tables, parameter limits, or tabulated thresholds can be extracted and linked to their corresponding regulation nodes. These auxiliary nodes would preserve both the textual transcription and structural metadata (e.g., row/column headers, caption context), enabling the system to retrieve structured constraints directly rather than relying on surrounding paragraph descriptions.

Another design direction is to explicitly tag regulation nodes by content type, such as text, table, figure, or appendix. During retrieval, the KG-RAG pipeline can selectively include or exclude specific content types depending on the query intent. For example, quantitative queries may prioritize table nodes, while definitional queries may prioritize narrative text. This content-aware filtering mechanism reduces retrieval noise and allows answer generation to be grounded in the most semantically appropriate evidence.

Integrating multi-format regulatory evidence can strengthen both answer precision and interpretability. By reducing reliance on incomplete textual descriptions and directly grounding

responses in structured artifacts, the system can provide more accurate constraint extraction, clearer citation traceability, and improved support for quantitative compliance reasoning.

4.5 Enhancing KG approach for question answering

Currently, the KG captures the CFR hierarchy and the explicit cross-references between sections. A clear next step is to enrich the graph with additional structure that reflects how people read and apply the regulations. The main goal is to help the QA system retrieve the most relevant rule text and supporting pieces that often determine the correct interpretation, such as definitions and carve-outs. This enrichment can make responses more accurate and easier to defend during review.

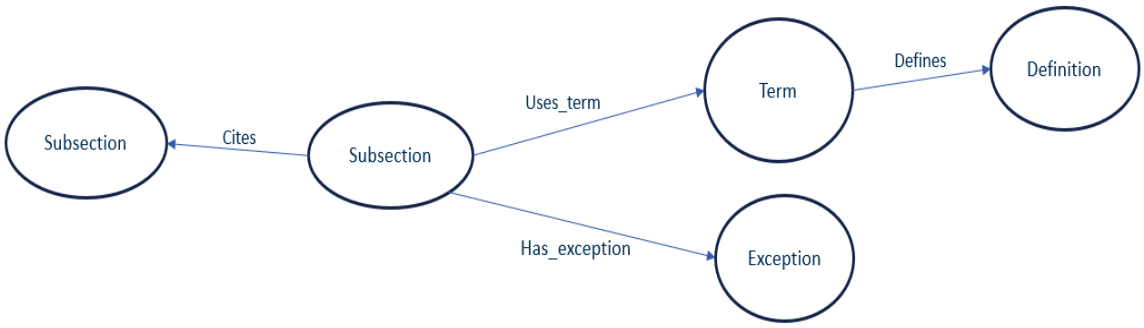

Figure 25. Enriched KG

Link Terms to Official CFR Definitions: One of the most common failure modes in regulation QA is treating a defined term as if it has everyday meaning. To reduce this, Term nodes and Definition nodes are added and connected with edges such as USES_TERM from a subsection or paragraph to a term and DEFINES or DEFINED_IN from the term to its definition text. At query time, once a governing subsection is selected, the system can automatically pull the definitions for the terms used in that subsection and include them in the evidence set passed to the language model. This encourages definition-grounded answers and reduces errors that come from plain language interpretation.

Represent Exceptions as First-Class Nodes: Another common issue is that an answer cites the main rule but misses a key exception, limitation, or condition. These carve-outs appear as "except," "unless," "does not apply," or "provided that." To handle this more reliably, exceptions can be represented explicitly as Exception nodes and connected to governing text with edges such as HAS_EXCEPTION or EXCEPTION_TO. With this structure, the retrieval step can pull exceptions deterministically whenever a subsection is used as evidence, instead of hoping that

semantic search happens to retrieve the carve-out text. This directly reduces correct but incomplete answers.

In summary, enriching the KG with term definition links and explicit exception structures is a practical way to improve QA performance without changing the overall retrieval and synthesis workflow. It makes the system more meaning-aware, reduces missed carve-outs, and supports citations that are more complete and easier to audit.

5 Conclusions

This report presented a multi-faceted approach to advancing risk-informed decision-making in aviation by combining advanced NLP, structured regulatory representation, and multi-agent RAG techniques. Through these efforts, the project addressed challenges in extracting meaningful insights from narrative safety data and navigating the complex structure of regulatory text.

A major contribution of the project was the development of two new datasets designed to enhance the ability to extract safety-relevant facts from ASRS and NTSB incident and accident reports. The first dataset is a question answering database containing 244 unique questions-answer pairs designed to enable precise supervision for LLM fine-tuning. The work detailed in this report demonstrates that the dataset's size, diversity, and quality-controlled annotations make it well suited for training models to identify events, contributing factors, and operational outcomes directly from narrative safety reports, thereby supporting more robust, data-driven safety assessments.

The second dataset consists of technically accurate synthetic narratives intended to improve model performance in multilabel classification of NTSB sequences of events. Prior work has shown that severe class imbalance across event categories limits the effectiveness of classification models, and that conventional synthetic-data methods often produce reports with unrealistic aircraft-engine characteristics or operational parameters. To address these limitations, this project employed a fine-tuned Mistral-7B model informed by a domain-specific KG derived from FAA aircraft data to generate synthetic reports. Analysis of the resulting data indicates that our approach eliminates hallucinated aircraft specifications, achieving an ASC score of 1.000 while maintaining lexical diversity comparable to authentic narratives. Moreover, augmenting underrepresented event categories led to substantial improvements in downstream classification performance, increasing the macro F1-score from 0.76 to 0.83. This demonstrated the value of knowledge-constrained generation in safety-critical aviation applications.

Parallel efforts focused on improving the accessibility and interpretability of regulatory information through the development of a regulation-structure-aware KG aligned with Title 14 of the CFR. The KG captured hierarchical relationships, preserved canonical identifiers, and surfaced cross-references as explicit, traversable links, enabling more accurate retrieval and explanation of regulatory content. The work reported here details the step-by-step process for generating a Regulation-KG. Future efforts will be directed towards extending cross-references to capture paragraph-level citations, citation ranges, chained and conditioned references, and incorporation by reference to external standards. Then, the expanded KG will be incorporated into a KG-RAG framework, and an assessment of KG-RAG reduction in hallucinations for regulations queries will be performed.

The project's technical developments were also integrated into the Daedalus multi-agent RAG system. This integration involved migrating the Daedalus codebase to local infrastructure, enabling multi-source retrieval through the newly developed KG, and implementing a novel completeness-evaluation agent capable of distinguishing between retrieval-based and generation-based incompleteness. This agent provides actionable diagnostics that improve both model performance and system reliability. Additional work is needed to assess when KG augmentation provides meaningful benefits and when it introduces unnecessary complexity.

Together, the contributions of this project significantly advance the datasets, methods, and tools available to support risk-informed decision-making in aviation safety. The work establishes a stronger technical basis for extracting actionable insights from narrative reports, interpreting regulatory text in a structured and reliable manner, and integrating these capabilities into scalable AI systems.

6 References

Ammann, P., Golde, J., & Akbik, A. (2025). *Question Decomposition for Retrieval-Augmented Generation.*

Anthropic. (2024, 11 25). *Introducing the Model Context Protocol*. Retrieved from https://www.anthropic.com/news/model-context-protocol

Anthropic. (2024). *Model Context Protocol: An Open Protocol for Seamless Integration Between LLM Applications and External Data Sources*. Retrieved 1 10, 2025, from https://modelcontextprotocol.io

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., . . . al., e. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS 2020)*, (pp. 1877–1901). Retrieved from https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

Buda, M., Maki, A., & Mazurowski, M. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks, 106*, 249–259. doi:10.1016/j.neunet.2018.07.011

Celikyilmaz, A., Clark, E., & Gao, J. (2020). *Evaluation of Text Generation: A Survey.* Retrieved from https://arxiv.org/abs/2006.14799

Chan, C.-M., Xu, C., Yuan, R., Luo, H., Xue, W., Guo, Y., & Fu, J. (2024). *RQ-RAG: Learning to Refine Queries for Retrieval Augmented Generation.*

Chandra, C., Jing, X., Bendarkar, M., Sawant, K., Elias, L., Kirby, M., & Mavris, D. (2023). Aviation-BERT: A Preliminary Aviation-Specific Natural Language Model. *AIAA Aviation 2023 Forum.* doi:10.2514/6.2023-3436

CrewAI. (2024). *CrewAI: Framework for Orchestrating Role-Playing, Autonomous AI Agents*. Retrieved 1 10, 2025, from https://www.crewai.com

Cyganiak, R., Wood, D., & Lanthaler, M. (2014). *RDF 1.1 Concepts and Abstract Syntax.* Retrieved from https://www.w3.org/TR/rdf11-concepts/

David, E. (2024, 11 25). *Anthropic releases Model Context Protocol to standardize AI-data integration*. Retrieved from https://venturebeat.com/

Dhingra, B., Faruqui, M., Parikh, A., Chang, M.-W., & Cohen, W. W. (2019). Handling Divergent Reference Texts when Evaluating Table-to-Text Generation. *ACL 2019*, (pp. 4884–4895). doi:10.18653/v1/P19-1483

Docker Inc. (2024). *Docker: Accelerated Container Application Development*. Retrieved 1 10, 2025, from https://www.docker.com

Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., . . . Jégou, H. (2024). *The Faiss library.* Retrieved from https://arxiv.org/abs/2401.08281

Dziri, N., Milton, S., Yu, M., Zaiane, O., & Reddy, S. (2022). On the Origin of Hallucinations in Conversational Models: Is it the Datasets or the Models? *NAACL 2022*, (pp. 5271–5285). doi:10.18653/v1/2022.naacl-main.387

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., . . . Larson, J. (2024). *From Local to Global: A Graph RAG Approach to Query-Focused Summarization.* Retrieved from https://arxiv.org/abs/2404.16130

Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2024). RAGAS: Automated Evaluation of Retrieval Augmented Generation. *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, (pp. 150–158). Retrieved from https://aclanthology.org/2024.eacl-demo.16/

et al. (2023). *Mistral 7B.* Retrieved from https://arxiv.org/abs/2310.06825

Federal Aviation Administration. (2024, October). Aircraft Characteristics Database. Retrieved from https://www.faa.gov/airports/engineering/aircraft_char_database

Federal Aviation Administration, Office of Aviation Medicine. (2001). *A Human Error Analysis of Commercial Aviation Accidents Using the Human Factors Analysis and Classification System (HFACS).* Retrieved from https://apps.dtic.mil/sti/tr/pdf/ADA387808.pdf

Francis, N., Green, A., Guagliardo, P., Libkin, L., Lindaaker, T., Marsault, V., . . . Taylor, A. (2018). Cypher: An Evolving Query Language for Property Graphs. *SIGMOD 2018*, (pp. 1433–1445). doi:10.1145/3183713.3190657

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., . . . Wang, H. (2023). *Retrieval-Augmented Generation for Large Language Models: A Survey.* Retrieved from https://arxiv.org/abs/2312.10997

Garcia, A. (2025). sqlite-vec: An extremely small, "fast enough" vector search SQLite extension that runs anywhere! (Version 0.1.7-alpha.2) [Computer software]. Retrieved from https://github.com/asg017/sqlite-vec

Gatt, A., & Krahmer, E. (2018). Survey of the state of the art in natural language generation: core tasks, applications and evaluation. *Journal of Artificial Intelligence Research, 61*, 65–170. doi:10.1613/jair.5477

Gehrmann, S., Adewumi, T., Aggarwal, K., Ammanamanchi, P. S., Aremu, A., Bosselut, A., . . . Zhou, J. (2021). The GEM Benchmark: Natural Language Generation, its Evaluation and Metrics. *GEM 2021 Workshop*, (pp. 96–120). doi:10.18653/v1/2021.gem-1.10

Gerganov, G. (2023). *llama.cpp - LLM inference in C/C++*. Retrieved 1 15, 2025, from https://github.com/ggml-org/llama.cpp

Goodrich, B., Rao, V., Liu, P. J., & Saleh, M. (2019). Assessing the Factual Accuracy of Generated Text. *KDD '19*, (pp. 166–175). doi:10.1145/3292500.3330955

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., . . . Vaughan, A. (2024). *The Llama 3 Herd of Models.* Retrieved from https://arxiv.org/abs/2407.21783

Gu, J., Jiang, X., Shi, Z., Tan, H., Zhai, X., Xu, C., . . . Guo, J. (2025). *A Survey on LLM-as-a-Judge.*

Guu, K., Lee, K., Tung, Z., Pasupat, P., & Chang, M.-W. (2020). *REALM: Retrieval-Augmented Language Model Pre-Training.* Retrieved from https://arxiv.org/abs/2002.08909

Hagberg, A., Schult, D., & Swart, P. (2008). Exploring Network Structure, Dynamics, and Function using NetworkX. *Proceedings of the 7th Python in Science Conference*, (pp. 11–15).

Hashimoto, T. B., Zhang, H., & Liang, P. (2019). Unifying Human and Statistical Evaluation for Natural Language Generation. *NAACL 2019*, (pp. 1689–1701). doi:10.18653/v1/N19-1169

He, H., & Garcia, E. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering, 21*, 1263–1284. doi:10.1109/TKDE.2008.239

Hegde, H., Vendetti, J., Goutte-Gattat, D., Caufield, J. H., Graybeal, J. B., Harris, N. L., . . . Mungall, C. J. (2024). *A Change Language for Ontologies and Knowledge Graphs.* Retrieved from https://arxiv.org/abs/2409.13906

Holtzman, A., Buys, J., Du, L., Forbes, M., & Choi, Y. (2020). The Curious Case of Neural Text Degeneration. *ICLR 2020.* Retrieved from https://openreview.net/forum?id=rygGQyrFvH

Hu, E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., . . . Chen, W. (2022). LoRA: Low-Rank Adaptation of Large Language Models. *International Conference on Learning Representations (ICLR).* Retrieved from https://openreview.net/pdf?id=nZeVKeeFYf9

Huang, J., Abadi, D., & Ren, K. (2011). Scalable SPARQL Querying of Large RDF Graphs. *VLDB*, (pp. 1123–1134).

Hugging Face. (2023, December). *Mistral-7B-Instruct-v0.2*. Retrieved from https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2

Hugging Face. (n.d.). *Trainer - Hugging Face*. Retrieved from https://huggingface.co/docs/transformers/main_classes/trainer

Hunter, J. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering, 9*, 90–95. doi:10.1109/MCSE.2007.55

International Civil Aviation Organization (ICAO). (2010). *ICAO ADREP 2000 Taxonomy.* Retrieved from https://skybrary.aero/sites/default/files/bookshelf/1809.pdf

Jing, X., Bhanpato, J., Bendarkar, M. V., & Mavris, D. N. (2025). An Efficient Dual-Agent Framework for Generating and Evaluating Synthetic Aviation Safety Reports using Large Language Models. *AIAA AVIATION Forum and ASCEND 2025.*

Jing, X., Chennakesavan, A., Chandra, C., Bendarkar, M. V., Kirby, M., & Mavris, D. N. (2023). BERT for Aviation Text Classification. *AIAA AVIATION 2023 Forum.* doi:10.2514/6.2023-3438

Jing, X., Chennakesavan, A., Chandra, C., Bendarkar, M., Kirby, M., & Mavris, D. (2023). BERT for Aviation Text Classification. *AIAA AVIATION 2023 Forum*, (p. 3438). doi:10.2514/6.2023-3438

Jing, X., Sawant, K., Bendarkar, M., Elias, L., & Mavris, D. (2024). Expanding Aviation Knowledge Graph Using Deep Learning for Safety Analysis. *AIAA Aviation Forum and ASCEND 2024.* doi:10.2514/6.2024-4603

Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data, 7*, 535–547.

Joshi, K. P., & Saha, S. (2020). A semantically rich framework for knowledge representation of code of federal regulations. *ACM Journal Digital Government: Research and Practice*, 1-17.

JSON-RPC Working Group. (2013). *JSON-RPC 2.0 Specification.*

Keller, R. (2019). Building a Knowledge Graph for the Air Traffic Management Community. *The Web Conference (WWW '19)*, (pp. 1676–1680). doi:10.1145/3308560.3317706

Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., & Lewis, M. (2019). *Generalization through Memorization: Nearest Neighbor Language Models.* Retrieved from https://arxiv.org/abs/1911.00172

Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence, 5*, 221–232. doi:10.1007/s13748-016-0094-0

LangChain AI. (2024). *LangGraph: Building Language Agents as Graphs*. Retrieved 1 10, 2025, from https://langchain-ai.github.io/langgraph/

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., . . . Kiela, D. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.* doi:10.48550/arXiv.2005.11401

Li, J., Galley, M., Brockett, C., Gao, J., & Dolan, B. (2016). A Diversity-Promoting Objective Function for Neural Conversation Models. *NAACL 2016*, (pp. 110–119). doi:10.18653/v1/N16-1014

Li, J., Qian, L., & Taoxiong, L. (2024). Construction of Legal Knowledge Graph Based on Knowledge-Enhanced Large Language Models. *Information*, 1-17.

Logan, R., Liu, N. F., Peters, M. E., Gardner, M., & Singh, S. (2019). Barack's Wife Hillary: Using Knowledge Graphs for Fact-Aware Language Modeling. *ACL 2019*, (pp. 5962–5971). doi:10.18653/v1/P19-1598

Lowin, J. (2025). fastmcp: The fast, Pythonic way to build MCP servers and clients (Version 2.13.1) [Computer software]. Retrieved from https://github.com/jlowin/fastmcp

Mangortey, E., Singh, S., Chen, S., & Sarkhel, K. (2025). Aviation Language Understanding Evaluation (ALUE) – Large Language Model Benchmark with Aviation Datasets. *AIAA AVIATION Forum and ASCEND 2025.* AIAA.

Mangrulkar, S., Gugger, S., Debut, L., Belkada, Y., Paul, S., & Bossan, B. (2022). *PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods*. Retrieved from https://github.com/huggingface/peft

Meta. (2024). *Meta Llama 3.3 multilingual large language model*. Retrieved 1 15, 2025, from https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct

Min, S., Krishna, K., Lyu, X., Lewis, M., Yih, W.-t., Koh, P., . . . Hajishirzi, H. (2023). FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing* (pp. 12076–12100). Singapore: Association for Computational Linguistics.

Model Context Protocol Contributors. (2025). Model Context Protocol Python SDK (Version 1.22.0) [Computer software]. Retrieved from https://github.com/modelcontextprotocol/python-sdk

Model Context Protocol Contributors. (2025). *Model Context Protocol: An open protocol that enables seamless integration between LLM applications and external data sources and tools [Computer software and protocol specification]*. Retrieved from https://github.com/modelcontextprotocol

Monteiro, J., Sá, F., & Bernardino, J. (2023). Experimental Evaluation of Graph Databases: JanusGraph, Nebula Graph, Neo4j, and TigerGraph. *Applied Sciences, 13*, 5770. doi:10.3390/app13095770

NASA. (2021, 12). *ASRS Program Briefing.* Retrieved 1 15, 2025, from https://asrs.arc.nasa.gov/docs/ASRS_ProgramBriefing.pdf

NASA. (2025). *ASRS Database Online - Aviation Safety Reporting System*. Retrieved 1 15, 2025, from https://asrs.arc.nasa.gov/search/database.html

National Transportation Safety Board. (2025). *MDB Download Directory - NTSB.ADMS.DataTransfer.Web*. Retrieved 1 15, 2025, from https://data.ntsb.gov/avdata

National Transportation Safety Board. (2025). *NTSB Aviation Investigation Search*. Retrieved 1 15, 2025, from https://www.ntsb.gov/Pages/AviationQueryV2.aspx

Neo4j Labs. (2023). *APOC: Awesome Procedures on Cypher for Neo4j*. Retrieved 11 10, 2025, from https://neo4j.com/labs/apoc/

Neo4j Labs. (2023). *Neosemantics (n10s): RDF and Linked Data in Neo4j*. Retrieved 11 10, 2025, from https://neo4j.com/labs/neosemantics/

Noh, S., & Shortle, J. (2018). Barrier Analysis of an Aviation Safety Assessment Model. *INCOSE International Symposium, 28*, 616–627. doi:10.1002/j.2334-5837.2018.00504.x

Nussbaum, Z., Morris, J., Duderstadt, B., & Mulyar, A. (2024). *Nomic Embed: Training a Reproducible Long Context Text Embedder.* Retrieved from https://arxiv.org/abs/2402.01613

Office of the Federal Register (OFR). (2026, May 15). *eCFR: Title 14 of the CFR -- Aeronautics and Space*. Retrieved from Code of Federal Regulations: A point in time eCFR system: https://www.ecfr.gov/current/title-14

Ollama. (2024). *Ollama: Get up and running with large language models locally*. Retrieved 11 10, 2025, from https://ollama.com

Ollama. (2025). Ollama: Get up and running with large language models (Version 0.13.0) [Computer software]. Retrieved from https://github.com/ollama/ollama

Partnership for an Advanced Computing Environment (PACE). (2017). *PACE, Partnership for an Advanced Computing Environment*. Retrieved 5 25, 2025, from http://www.pace.gatech.edu

Prefect. (2025). *FastMCP: The fast, Pythonic way to build MCP servers and clients*. Retrieved from https://www.gofastmcp.com/

Pydantic. (2024). *Pydantic AI: Agent Framework Using Pydantic*. Retrieved 1 10, 2025, from https://ai.pydantic.dev

Pydantic AI. (2025). *Pydantic AI: GenAI Agent Framework, the Pydantic way*. Retrieved from https://ai.pydantic.dev/

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). *SQuAD: 100,000+ Questions for Machine Comprehension of Text.* Retrieved from https://arxiv.org/abs/1606.05250

Roelen, A., Verstraeten, J., Speijker, L., Bravo Muñoz, S., Heckmann, J., Save, L., & Longhurst, T. (2016). *Risk models and accident scenarios in the total aviation system.* Netherlands Aerospace Centre (NLR). Retrieved from https://reports.nlr.nl/server/api/core/bitstreams/97e2fd30-cf53-4dab-8a43-fc1c00294b9e/content

Samuylova, E. (2025, 7 23). *LLM-as-a-judge: a complete guide to using LLMs for evaluations*. Retrieved from https://www.evidentlyai.com/llm-guide/llm-as-a-judge

Shannon, C. (1948). A Mathematical Theory of Communication. *The Bell System Technical Journal, 27*, 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x

Shi, B., Bai, X., & Yao, C. (2017). An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Shuster, K., Poff, S., Chen, M., Kiela, D., & Weston, J. (2021). *Retrieval Augmentation Reduces Hallucination in Conversation.* doi:10.48550/arXiv.2104.07567

Su, W., Tang, Y., Wu, Z., Ai, Q., & Liu, Y. (2024). *DRAGIN: Dynamic Retrieval Augmented Generation based on the Real-time Information Needs of Large Language Models.* Retrieved from https://arxiv.org/html/2403.10081v1

Tanguy, L., Tulechki, N., Urieli, A., Hermann, E., & Raynal, C. (2016). Natural language processing for aviation safety reports: From classification to interactive analysis. *Computers in Industry, 78*, 80–95.

Templin, M. (1957). *Certain Language Skills in Children: Their Development and Interrelationships.* University of Minnesota Press. Retrieved from https://www.jstor.org/stable/10.5749/j.ctttv2st

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., . . . al., e. (2023). *Llama 2: Open Foundation and Fine-Tuned Chat Models.* Retrieved from https://arxiv.org/abs/2307.09288

van der Lee, C., Gatt, A., van Miltenburg, E., Wubben, S., & Krahmer, E. (2019). Best Practices for the Human Evaluation of Automatically Generated Text. *INLG 2019*, (pp. 355–368). doi:10.18653/v1/W19-8643

Wang, H., & Shi, Y. (2025). Knowledge Graph Combined with Retrieval-Augmented Generation for Enhancing LMs Reasoning: A Survey. *Academic Journal of Science and Technology*.

Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., . . . al., e. (2024). A Survey on Large Language Model Based Autonomous Agents. *Frontiers of Computer Science, 18*, 186345.

World Wide Web Consortium (W3C). (2013). *SPARQL 1.1 Query Language.* Retrieved from https://www.w3.org/TR/sparql11-query/

Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., . . . al., e. (2023). *The Rise and Potential of Large Language Model Based Agents: A Survey.* Retrieved from https://arxiv.org/abs/2309.07864

Yasunaga, M., Bosselut, A., Ren, H., Zhang, X., Manning, C. D., Liang, P., & Leskovec, J. (2022). *Deep Bidirectional Language – Knowledge Graph Pretraining.* doi:10.48550/arXiv.2210.09338

Zheng, L., Chiang, W., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., . . . Stoica, I. (2023). Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *Advances in Neural Information Processing Systems 36 (NeurIPS 2023).*

Zhong, L., Wu, J., Li, Q., Peng, H., & Wu, X. (2023). A comprehensive survey on automatic knowledge graph construction. *ACM Computing Surveys, 56*.