frontiers in Aerospace Engineering

TYPE Original Research
PUBLISHED 14 September 2026
DOI 10.3389/fpace.2026.1808857

Check for updates

OPEN ACCESS

EDITED BY
Zhaodan Kong,
University of California, Davis,
United States

REVIEWED BY
Qingli Liu,
The Hong Kong Polytechnic University,
Hong Kong SAR, China
Ziquan Deng,
University of California, Davis,
United States

*CORRESPONDENCE
Aida Sharif Rohani,
aida.sharifrohani@nasa.gov

RECEIVED 11 February 2026
REVISED 07 July 2026
ACCEPTED 16 July 2026
PUBLISHED 14 September 2026

CITATION
Sharif Rohani A, Clarke SS and Das A (2026) A large language model–based method for improving decision-making in aviation.
Front. Aerosp. Eng. 5:1808857.
doi: 10.3389/fpace.2026.1808857

COPYRIGHT
© 2026 Sharif Rohani, Clarke and Das. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

A large language model–based method for improving decision-making in aviation

Aida Sharif Rohani*, Stephen S. Clarke and Aditya Das

NASA Ames Research Center, Moffett Field, CA, United States

The rapid emergence of advanced air mobility technologies such as air taxis and delivery drones is transforming the aviation landscape by introducing new layers of complexity for decision-making in dynamic and crowded airspaces. To support this evolving environment, we propose a Large Language Model (LLM)-based method that improves decision-making in aviation operations by leveraging users' natural language queries alongside validated aviation expert models and real-time data sources. This method interprets user intent, selects appropriate expert tools such as models for trajectory prediction, runway configuration, and Traffic Management Initiative (TMI) forecasting, and provides contextualized responses using relevant aviation data sources. Unlike general purpose LLM applications, the proposed system grounds its outputs in aviation-validated tools and data sources, addressing the risk of hallucination in a domain where inaccurate information can have serious operational consequences. Evaluated on 1,000 domain-specific aviation queries across nine tools using two models, GPT-4o and GPT-4o-mini, the system achieves tool selection accuracy ranging from 86% to 100% across all tools, with results independently validated on a held-out test set. This work illustrates the potential of LLMs as decision support tools in aviation when carefully grounded in validated domain-specific models and datasets.

KEYWORDS
large language models, aviation decision support, tool calling, air traffic management, prompt engineering, advanced air mobility, LLM agents

1 Introduction

As aviation operations expand and become more complex with the rise of air taxis, delivery drones, and other emerging technologies (Lee et al., 2022), it will be essential to ensure these novel services are safely integrated within the existing operations of the National Airspace System (NAS). This evolution will present aviation decision-makers with unprecedented challenges as they manage increasingly crowded, and complex airspace environments. At the same time, aviation personnel may face growing information overload as they attempt to digest traffic data, weather updates, and NOTAMs in real time. These challenges are intensified by human factors such as fatigue, stress, and variability in expertise across shifts that make it difficult to maintain consistently high situational awareness (Johnson and William, 2006).

While AI holds promise for optimizing operations, and enhancing safety in complex airspace environments, current guidelines from the Federal Aviation Administration (FAA) stress the importance of incremental AI deployment with robust output verification before any operational use (Federal Aviation Administration, 2024).

Recent advances in Large Language Models (LLMs), such as the Generative Pre-trained Transformer (GPT) (Radford et al., 2019), Large Language Model Meta AI (LLaMA) (Taori et al., 2023; Touvron et al., 2023), and Text-to-Text Transfer Transformer (T5) series (Raffel

et al., 2020), have demonstrated exceptional capabilities in learning, reasoning, and understanding natural language. These strengths enable LLMs to select, and interact with external tools to address complex, domain-specific tasks.

LLMs can play an important role as centralized knowledge systems by retrieving and synthesizing information from FAA documents, safety protocols, regulations, various aviation models, and simulation software that capture operational and technical aspects of the airspace (Schlichting et al., 2025; Li et al., 2024; Wang et al., 2023). However, LLMs are also prone to hallucinations, and capable of producing outputs that may be plausible in language but are not grounded in factual truth (Xu and Kankanhalli, 2024; Banerjee and Singla, 2024). One study from Nature illustrates that as LLMs scale, their tendency to provide confident yet incorrect responses increases (Zhou et al., 2025). To mitigate these risks, one solution is to ground LLM outputs in validated aviation knowledge and tools, such as those used in ToolGen (Wang et al., 2025), rather than relying solely on statistical language patterns. Integrating LLMs with expert aviation models and tools developed through decades of rigorous research can enhance the transparency and robustness of future decision-support systems.

While combining LLMs with tools is an established paradigm, most existing work focuses on general purpose or single-domain tasks where tools are well-defined and broadly applicable. In contrast, the method proposed here targets a domain-specific use case where the tools themselves are aviation-validated models and data sources, and the challenge is getting the LLM to reliably invoke them for common operational tasks such as retrieving weather observations, finding routes, or predicting trajectory parameters. This is non-trivial because aviation tools operate under strict constraints, where certain models only support specific aircraft types or airports, and the system must recognize when a query falls outside those constraints rather than calling a tool with invalid inputs. Queries from aviation professionals are often underspecified, using domain shorthand that omits required parameters, which means the system must handle multi-turn clarification rather than assuming defaults. With 9 tools that overlap semantically, incorrect tool selection is both frequent and consequential given the operational nature of the domain.

Figure 1 illustrates the overview of our method. Upon receiving a user query, relevant inputs are extracted, and subsequently specific tools from the expert toolset (such as TMI forecasting, trajectory prediction, or information retrieval tools) are called to generate a response with references to the models and data sources used. This loop continues as long as the user has follow-up questions, and each interaction uses the updated context from previous queries. The contributions of this paper are as follows:

- A multi-tool LLM-based framework for aviation that grounds natural language queries in domain-specific validated models, real-time data sources, and simulation outputs, reducing dependence on general-purpose language outputs that lack accuracy guarantees.
- A methodology for integrating LLMs with operational aviation tools through structured prompt engineering and human-in-the-loop validation, where domain experts iteratively refine tool descriptions to achieve reliable tool selection across a diverse set of domain-specific tasks.
- A dataset of 1,000 manually drafted and verified aviation queries across nine tools and five question categories, including edge cases and wrong-tool temptations, designed to evaluate LLM-based tool selection in a realistic operational setting.
- An evaluation framework comparing two model sizes across multiple aviation tools, providing insight into the tradeoffs between model capability and constraint following in domain-specific LLM applications.

The remainder of the paper is organized as follows. Section 2 provides a brief background on relevant LLM techniques. Section 3 describes our LLM-based method, including the aviation tools and datasets used, and the performance metrics. In Section 4, we present the results and evaluate the performance. Finally, Section 5 discusses lessons learned and concludes the paper.

2 Background

This section reviews the key techniques that enable our LLM-based method. First, we describe LLM-based agents and explain how they interpret user queries and coordinate aviation tools and models. Next, we introduce retrieval-augmented generation (RAG), which grounds responses in verified aviation documents, datasets, and live operational data. Finally, we discuss prompt engineering methods used to ensure context-aware behavior in supporting decision support systems in aviation.

2.1 LLM-based agents

LLMs are advanced artificial intelligence systems designed to process and generate human language. At their core, they are transformer-based neural networks (Vaswani et al., 2017) trained on vast collections of text, ranging from books and articles to websites and technical documents. Through this large-scale training, LLMs learn patterns of language use, which enables them to not only recognize grammar and vocabulary but also capture context and reasoning. Because of these capabilities, LLMs can perform a wide variety of NLP tasks such as text generation, summarization, question answering, and information extraction. Their ability to understand context, extract relevant information, and produce coherent responses makes them especially powerful for applications that require the comprehension of complex textual inputs. In the context of the method, the LLM operates as an agent that can interpret user queries, select the most relevant tools, retrieve necessary inputs, and synthesize outputs for decision support. This behavior is characterized by several key components:

- Autonomous reasoning: The LLM agent can process information, make decisions, and execute tasks without continuous human intervention. This autonomy is crucial for real-time applications where timely responses are essential (Alanazi, 2025).
- Tool integration: The agent can interface with various domain-specific tools and databases, and provides accurate and contextually relevant information (Schlichting et al., 2025; Wang et al., 2025).

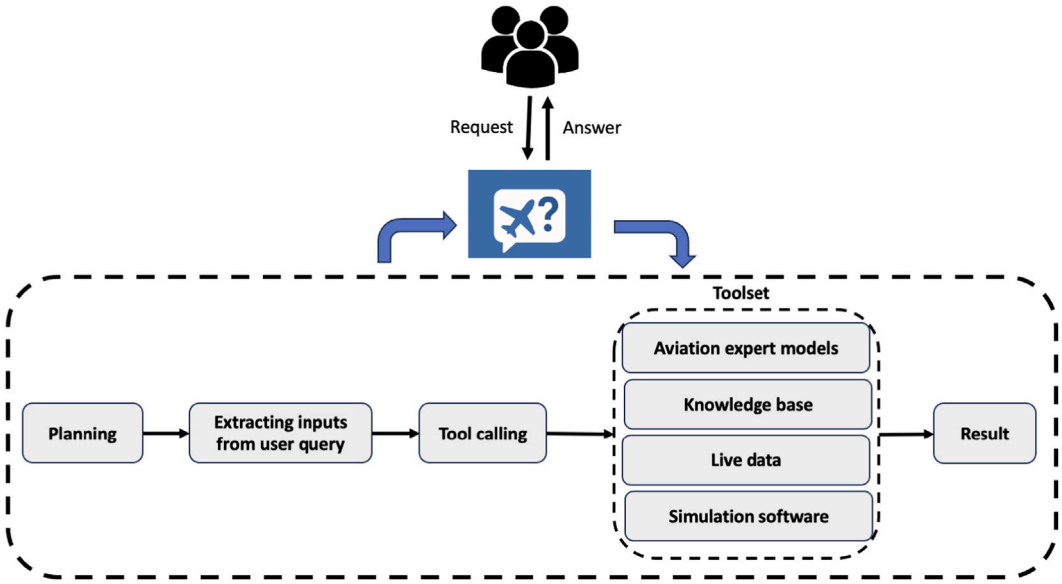

FIGURE 1
The overview of the LLM-based method for supporting decision-making in aviation.

- Contextual understanding: Leveraging its training on diverse datasets, the LLM agent can understand and adapt to the nuances of domain-specific terminology and scenarios to ensure that its outputs are appropriate for the domain (Lewis et al., 2020).

These capabilities enable LLM-based agents to act as an additional layer of support in complex, dynamic environments like aviation, where they can assist in tasks ranging from real-time data queries to various decision-making support scenarios.

Recent agent frameworks such as ReAct (Yao et al., 2023) and Toolformer (Schick et al., 2023) demonstrate that LLMs can effectively reason about and invoke external tools. ReAct interleaves reasoning traces with action steps to improve decision making in multi-step tasks, while Toolformer trains the model to decide when and how to call tools through self-supervised learning. Unlike these frameworks, our method does not require training or reasoning traces and relies entirely on natural language tool descriptions and prompt engineering to guide tool selection. The contribution of this work is not a new agent algorithm but rather the application and evaluation of LLM-based tool calling in a domain-specific aviation setting with validated operational tools, where the challenge lies in reliable tool selection under strict operational constraints rather than open-ended reasoning.

2.2 Retrieval-augmented generation (RAG)

Retrieval-Augmented Generation (RAG) was introduced by Lewis et al. (2020) to address the limitations of LLMs that depend only on internal knowledge. While such models often generate fluent text, they can produce factual errors when handling knowledge-intensive tasks. RAG combines two main components: a retriever, which searches external knowledge sources such as dense passage retrievers or vector databases, and a generator that produces final outputs conditioned on both the retrieved evidence and the model's internal knowledge. To improve accuracy and contextual relevance, RAG grounds responses in relevant documents. It also allows easier adaptation to specialized domains without retraining the base model and supports transparency by linking generated responses to source material. As a result, RAG has been applied in general NLP (Lewis et al., 2020), finance (Zhao et al., 2024), and aviation (Wang et al., 2023).

RAG typically involves processing an input query to formulate search terms, retrieving related documents from an external knowledge base, integrating these documents as additional context, and then generating a final response. The integration of retrieved context with internal model knowledge enhances output precision, especially when domain-specific accuracy is required.

2.3 Prompt engineering

Prompt engineering is a foundational technique in NLP that focuses on designing input prompts to guide language models toward producing the desired output (Sahoo et al., 2024). The effectiveness of prompt engineering lies in providing clear, structured, and context-rich instructions that align with the LLM's training data. Well-crafted prompts often include relevant context, example formats, and explicit instructions to help the model interpret the user's intent more accurately.

Prior research in NLP and generative AI has demonstrated that prompt quality significantly affects the relevance, coherence, and accuracy of model outputs (Chen and Langrene, 2023; Strobelt et al., 2022). Advanced prompting techniques include zero-shot prompting (Kojima et al., 2022), where the model is given only the task instruction without examples; few-shot prompting (Brown and Ryder, 2020), which provides a small number of examples to guide the model; and chain-of-thought prompting (Wei et al., 2022), where the model generates intermediate reasoning steps before producing a final answer.

3 Materials and methods

Our method integrates LLMs with a diverse set of domain-specific expert models to support complex, multi-step reasoning tasks in aviation. This integration enables LLMs to dynamically interpret user queries, retrieve relevant aviation data, and invoke the appropriate expert tools to generate validated and context-aware responses. By combining the language understanding and reasoning capabilities of LLMs with rigorously tested domain knowledge, the system provides reliable and interpretable support for air traffic management decisions. The core of the proposed method is an LLM that acts as a decision-making layer between air traffic end users and a library of validated expert models, datasets, documents, simulation software, APIs, and other tools. The system operates as follows:

1. Query interpretation: The method builds on LLM capabilities to process natural language queries. The central LLM identifies the user's intent, whether it's a data request, a general aviation question, an update on weather, routes, or runways, or the need to run a simulation. From there, key entities such as airport names, weather conditions, and TMIs are extracted to guide the next steps in processing.
2. Model selection: Based on the interpreted query, the most appropriate aviation tool or combination of tools are selected. When the user does not explicitly specify a tool, the central LLM is instructed to pick the correct tool(s) from its full tool list.
3. Model execution: Once the tool is selected, the type of input that the model from the tool list requires is determined. The user query is then mapped to the inputs which are being filled as completely as possible. Any missing input is automatically filled using default values or by invoking additional plugins to gather the necessary data. This ensures the model receives a complete and accurate set of inputs for execution.
4. Explanation generation: After model execution, the final answer, which could be a combination of numerical values and text, is generated. The final answer is further curated according to the user's initial query. For each tool, the LLM is instructed on how to present the results and thus provides a transparent explanation by referencing the data sources, models, and any assumptions made during processing.
5. Expert feedback loop: During a set of evaluations across various aviation scenarios, the LLM's behavior is assessed to ensure it performs as intended. Human domain experts review and verify the outputs. When incorrect models are selected or query interpretations are inaccurate, the issues are documented for further analysis. Based on this feedback, the model's inputs, outputs, and prompt design are refined to improve tool selection and information usage.

For the development of our research prototype, we used MADI (Modular AI Development Interface), a software platform developed under NASA's Convergent Aeronautics Solutions (CAS) project. MADI is designed to streamline the development of LLM-powered applications by providing an infrastructure for building and deploying AI systems within the NASA ecosystem. Our central LLM for this prototype was GPT-4o-2024-08-06, which serves as the core reasoning engine for processing queries, generating responses, and interfacing with aviation tools. MADI comes with pre-built support for user interfaces (e.g., web-based chat), a robust API for handling requests and model coordination, a local SQL database for storing query history or logs, vector search for document retrieval (RAG), and secure authentication mechanisms. Such features enable the method to act as a framework with a modular architecture consisting of three core components: a user interaction layer, a centralized API, and a suite of expert model and data plugins. This design enables us to integrate multiple aviation-specific capabilities while maintaining flexibility and scalability. This setup allows us to focus on research questions around LLM-based decision support, while relying on MADI's infrastructure to handle backend communication and system integration.

The LLM-based method integrates aviation-domain tools and data sources through a structured prompt-engineering strategy. Each domain-specific tool, such as those for runway configuration, trajectory prediction, or route selection, is paired with tailored prompts that instruct the central LLM on when and how to invoke the appropriate model. For example, if a user asks, "What is the best runway configuration for JFK Airport right now?", the LLM first determines the type of tool required, in this case, the runway configuration model, and then identifies the necessary inputs, such as current and forecasted weather conditions. This triggers a two-step tool invocation: first, the FAA API is called to retrieve and preprocess the relevant weather data; second, the processed inputs are passed to NASA's in-house runway configuration expert model. The model's prediction is then returned and synthesized into a clear, natural-language response for the user.

This approach ensures that the LLM does not rely on assumptions or hallucinations, but instead fetches the right data, invoke validated models, and contextualize the outputs for the user. In other words, such framework prevents black-box AI behavior by grounding responses in domain-vetted models and human feedback. To continually improve alignment with domain expectations, we incorporate a human-in-the-loop feedback loop, where aviation experts review system responses during the setup of new tools. If an incorrect tool is triggered or the input is misinterpreted, the prompts are revised to refine future behavior.

3.1 Aviation expert tool set

In this section, we describe the aviation expert models, documents, simulation software, and APIs integrated with our central LLM. Each tool, or plugin, as we sometimes refer to them interchangeably, was chosen for its validated performance in addressing key aviation tasks such as runway configuration, trajectory prediction, and information extraction from unstructured communication.

- Trajectory prediction (`tool: trajectory prediction`): For trajectory forecasting, we employed a machine learning model that estimates key aircraft performance model (APM) parameters, such as drag coefficients and aircraft weight, using historical flight records (Rohani et al., 2023; Puranik and Kalyanam, 2022). The model uses an ODE fitting approach to generate flight-specific APM labels and trains machine learning

models to predict these parameters for unseen flights. It was shown that this model improves the accuracy of trajectory predictions in the absence of proprietary aircraft data.

- Runway configuration prediction (`tool: runway configuration`): To support runway configuration decision-making, we integrated a rule-based model that leverages airport layout, operational constraints, and current weather conditions to predict the optimal runway configuration. The model helps anticipate changes in runway usage due to wind, or other ATM airport constraints such as runway closures.
- Air traffic management's information extraction (`tool: entity recognition`): We adopted a DeBERTa-based NLP pipeline from recent research that extracts vital ATM decisions and updates from FAA telecon transcripts (Rohani et al., 2025). The system combines intent classification and slot filling to identify structured information from unstructured dialogue, and achieves F1-scores close to human-level annotation. This model enables automated detection and classification of TMIs.
- ATCSCC operations plans (`tool: advisory`): This tool gets access to the most recent ATCSCC operations plan advisory which is hosted online at https://www.fly.faa.gov/adv/adv_spt.jsp. The operation plans are documents, updated about every 2 hours, which contain general information about the status of the NAS. They contain terminal and en route constraints through TMIs as well as general status or constraints to terminal areas across the country. This plugin lets the central LLM have access to this information which can be used for further prompts or inputs into other plugins.
- Aviation routes (`tool: routes`): The plugin is an aviation route API that identifies the best available routes based on the origin and destination airport by using FAA identifiers. The API supports parameters such as airport names, route type (e.g., High, Low, Special Directional, TEC), aircraft type, altitude constraints, and ARTCC (Air Route Traffic Control Center) identifiers for both departure and arrival.
- Aviation weather (`tool: weather`): This plugin takes in a query for date and time to access a public API hosted at https://aviationweather.gov/ for current weather conditions and forecasts. It has access to TAFs, METARS, aircraft reports, AirSIGMETS and GAIRMETs. The base LLM is able to parse weather information from the API, and use that in responses to the user and as inputs to other plugins.
- NASA live operational data services (`tool: live data`): NASA has developed cloud-based digital information services to support the transformation of the National Airspace System (NAS) by enabling data-driven airspace operations for both traditional and emerging aviation. These services provide a secure and scalable ecosystem for accessing and sharing real-time and historical air traffic data, and supporting improved situational awareness and operational decision-making (DIP, 2025). The proposed method interfaces with NASA's live operational data services through authenticated APIs, enabling secure access to high-rate flight data streams. This integration empowers our method by providing thousands of flight data updates per second for real-time analysis and evaluation.
- FAA manual information retrieval (`tool: faa manuals`): This plugin enables RAG on a collected dataset of publicly available FAA documents. Some notable documents include the Aeronautical Information Manual (AIM) and other important orders, notices, and handbooks provided by the FAA to the aviation community. There is a plethora of information stored within these documents which our method can quickly parse through and provide answers to prompts by the user. Our central LLM has access to these documents both through local file access and retrieval techniques, such as RAG, which was explained in the Background Section.
- BlueSky flight controller (`tool: BlueSky`): As part of our system's ability to simulate or test aviation decision-making processes, we integrated the BlueSky ATC Simulator which is an open-source platform for air traffic simulation (Leeuwen and Ellerbroek, 2016). BlueSky provides an accessible API that enables direct manipulation of aircraft state and behavior. It allows users to send flight commands to individual aircraft in real time and offers access to core flight parameters and logic, which makes it an ideal environment for reproducible and customizable experimentation.

3.2 Dataset

To evaluate the decision-making and tool-integration capabilities of our LLM-based method, we created a task-specific dataset of 1000 instances, with more than 75 different question-and-answer pairs per tool, all prepared by domain experts. The questions span the full functional range of each tool, and captures a wide variety of use cases, input scenarios, and expected outputs. The answers reflect both the capabilities and limitations of each tool. For example, the `trajectory prediction` tool supports only a limited subset of aircraft types and airports. The dataset includes queries both within the tool's operational scope and explicit edge cases outside it. In these edge cases, our LLM is expected to reason about the limitations and either refrain from providing a response or indicate that the data is unavailable. This setup evaluates our LLM's ability to interpret user intent, select the appropriate tool(s), and handle boundary conditions intelligently. Ground truth tool labels and input parameters were established by the same domain experts who drafted the questions. Rather than independent labeling followed by agreement measurement, questions were constructed collaboratively with consensus on the expected system behavior for each query. In cases of disagreement the more conservative interpretation was adopted.

The dataset spans five question categories designed to evaluate distinct failure modes. Single tool queries (59.8%) provide complete information and test basic tool invocation. Missing parameter queries (17.6%) omit one or more required inputs to test whether the system correctly requests clarification rather than invoking the tool with incomplete information. Edge case queries (10.7%) include out-of-scope inputs, fictional scenarios, and impossible conditions to test rejection behavior. Wrong tool queries (7.3%) are phrased to resemble a different tool and test disambiguation between semantically overlapping tools. Finally, no tool queries (4.6%) are general knowledge questions that do not require any tool invocation and test whether the system avoids unnecessary tool calls. The distribution reflects the operational reality that most aviation queries are well-formed, while a meaningful proportion involve ambiguity, missing information, or attempts to invoke the wrong tool.

The dataset covers all airports supported by the runway configuration tool (DEN, CLT, DFW), all aircraft types supported by the trajectory prediction tool (A319, A320, B737, AC_B738), all four supported destinations (DEN, DFW, LAX, MSP), and all five weather data sources available through the aviation weather tool (METARs, TAFs, etc.). Questions were distributed across a range of airports, carriers, altitudes, and operational scenarios to ensure the evaluation reflects realistic query diversity rather than a narrow set of conditions.

3.3 Metrics

To evaluate our method's performance, including its ability to select appropriate tools, interpret queries, and generate accurate outputs, we use accuracy, recall, precision, and F1 score as metrics. For each query y, let $T(y)$ denote the ground-truth set of tools, and $\hat{T}(y)$ be the predicted set of tools selected by the model. We define the Correct Selection Rate (CSR) as the percentage of queries where the predicted tool set exactly matches the ground-truth set:

$$\text{CSR} = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}1, \text{ if } \hat{T}(y_i) = T(y_i)\\ 0, \text{ otherwise}\end{cases} \quad (1)$$

where N is the total number of queries.

To assess the correctness of input parameters used in tool calls, whether extracted directly from the user query or obtained through prior tool invocations, we employ the Input Selection Accuracy (ISA) metric which measures the proportion of queries for which the predicted input parameter set exactly matches the ground truth:

$$\text{ISA} = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}1, \text{ if } \hat{I}_i = I_i\\ 0, \text{ otherwise}\end{cases} \quad (2)$$

where I_i is the ground-truth input set for the i-th query, and $\hat{I}_i$ is the predicted input set.

3.4 Human-in-the-loop feedback process

The human-in-the-loop feedback process involved two aviation domain experts with backgrounds in air traffic management and operations. For each new tool, experts reviewed system outputs against a set of representative queries and flagged cases where the wrong tool was selected or where inputs were misinterpreted. In the early stages of the process, particularly when the first tools were being introduced, more than 20 revision cycles were needed before the system reached acceptable performance. As patterns emerged for what prompt language worked well, the number of revisions required for each subsequent tool decreased considerably. In cases where the two experts disagreed on the expected system behavior, the more conservative interpretation was adopted. Additionally, a held-out test set of 50 questions was drafted independently by an aviation professional not involved in the prompt tuning process.

4 Results

Our LLM-based method's performance was assessed using the curated dataset of tool-specific questions described earlier. Each query was submitted to the system, and its responses were assessed manually by domain experts using the metrics defined in the previous section. The evaluation was conducted using two models: GPT-4o, a large and highly capable model, and GPT-4o-mini, a smaller and more efficient alternative. Comparing the two models provides insight into whether a lighter model can achieve comparable performance in a domain-specific tool calling setting, which has practical implications for deployment cost and latency. Table 3 presents the evaluation results for each tool across GPT-4o and GPT-4o-mini. GPT-4o achieves an average CSR of 95.2% and ISA of 96.1%, while GPT-4o-mini achieves 91.5% and 93.7% respectively across all nine tools. CSR is consistently high across most tools, with several reaching 100%, which indicates that the method reliably identifies the appropriate tool under clearly defined input conditions. ISA is also near perfect for most tools which reflects the system's ability to correctly extract input parameters once the right tool is selected.

The weather tool shows the lowest CSR for both models (86.3% for GPT-4o and 85.5% for GPT-4o-mini), primarily due to false positives where general knowledge questions about weather concepts such as definitions and procedures triggered a tool call rather than a direct response. Fictional or impossible scenarios also contributed to this pattern, as both models called the tool rather than declining the query. The runway configuration tool shows the largest performance gap between the two models (94.7% vs. 78.9%), as GPT-4o more consistently declined queries for airports outside the supported set while GPT-4o-mini was more likely to invoke the tool regardless of the airport constraint. ISA for both models is around 91% on correct tool calls, which indicates that input extraction is reliable once the correct tool is selected. The FAA Manuals tool reports CSR of 94.7% for GPT-4o and 93.6% for GPT-4o-mini. Live data and advisory tools both perform strongly, with CSR above 90% for both models. Named entity recognition achieves 100% CSR and ISA for both models, which is consistent with the clear and unambiguous nature of its invocation condition. Overall, the results confirm that the method maintains strong performance across a diverse set of aviation tools and query types.

Across all nine tools, GPT-4o consistently outperforms GPT-4o-mini on CSR, however the size of the gap varies considerably depending on the nature of the tool. For tools with strict operational constraints, such as runway configuration which only supports three specific airports, the gap is largest at 15.8 percentage points. Similarly, trajectory prediction and BlueSky show gaps of 4.6 and 5.4 points respectively, both tools that require specific aircraft types or command formats. This pattern suggests that GPT-4o is more reliable at recognizing and respecting operational boundaries, declining out-of-scope queries rather than attempting to invoke the tool with invalid inputs. In contrast, for tools with fewer constraints such as weather, routes, live data, and FAA manuals, the gap between the two models is consistently small, ranging from 0.8 to 1.5 percentage points, indicating that GPT-4o-mini performs comparably when tools do not require strict scope enforcement. This finding has practical implications for deployment: GPT-4o-mini may be a suitable alternative for open-ended tool calling tasks, while GPT-4o is preferable when tools have hard operational constraints that the system must reliably enforce.

To assess whether the reported performance generalizes beyond the development data, a held-out test set of 50 questions was drafted

TABLE 1 Dataset distribution across question categories for each tool.

Tool	#Samples	Single tool	Missing params	Edge case	Wrong tool	No tool
trajectory prediction	87	49	21	13	4	0
runway configuration	76	48	25	0	2	1
entity recognition	87	64	15	7	1	0
advisory	118	110	2	2	4	0
routes	145	113	23	7	2	0
weather	120	37	27	24	18	14
live data	142	43	33	28	22	16
faa manuals	94	94	0	0	0	0
blueSky	131	40	30	26	20	15
Total	**1000**	**598**	**176**	**107**	**73**	**46**

TABLE 2 Prompt instructions, and input/output requirements for various tools.

Tool	Prompt instruction	Input	Output
trajectory prediction	This model was designed and trained to find the optimal aviation performance model (APM). Fill out the missing fields based on the following parameters description when creating an input into the model. Show the assumptions from the second part of the JSON that the tool made	Aircraft, distance, pressure, temperature, month, destination	APM parameters (CD0, kappa)
runway configuration	This model was designed to predict the runway configurations at a particular airport provided the weather forecast. Please follow each of the input parameters when creating an input into the model	Airport, hour, scheduled arrival/departure, wind direction	Runway
entity recognition	This tool is used to find the named entities from a text that the user has	Passage	Named entities
advisory	This tool will search and get the most recent ATCSCC advisory from the NAS status website. The advisory contains important air traffic management information such as terminal and en route constraints, runway closures, and other important ATM information	Day, month, year	Advisory from a specific day and time
routes	This aviation route API can find the best route based on the origin and destination of the airport. You will get a set of arrays; from those select the route or routes and send it back to the user	Destination, origin, route, hours1-3, area, altitude, aircraft, flow, seq, ARTCC	Route
weather	Get the current and forecast weather in a given location, this tool can be used if the user needs the weather info as an input to another tool	Location	Weather condition
live data	This plugin fetches live data from the NASA data streaming service to answer a user question about NOTAMs or airspace. Only use it if the user asks for live flight data and nothing else	Flight number, destination, origin, airline	Location, flight info
faa manuals	Search in FAA's publicly available manuals, books, and orders pertaining to aviation	Query	Chunks of text extracted from manuals
blueSky	BlueSky is a simulation tool for air traffic management. If the user prompts you with a controller action, take the necessary steps to recreate that action with BlueSky commands. If there are multiple actions, make multiple tool calls. If the user specifies a direct command, pass along the command as is	Command, and flight's unique identifier	command(s) executed in BlueSky software

TABLE 3 The LLM-based method's performance (CSR and ISA) for each plugin across GPT-4o and GPT-4o-mini.

Tool	CSR (%)		ISA (%)		#Samples
	GPT-4o	GPT-4o-mini	GPT-4o	GPT-4o-mini	
trajectory prediction	97.7	93.1	92.1	86.6	87
runway configuration	94.7	78.9	91.1	91.3	76
entity recognition	100	100	100	100	87
advisory	99.2	96.6	100	100	118
routes	98.6	97.2	96.9	94.7	145
weather	86.3	85.5	86.6	77.5	120
live data	91.6	90.1	100	100	142
faa manuals	94.7	93.6	100	100	94
blueSky	93.9	88.5	98.6	93.0	131
average	**95.2**	**91.5**	**96.1**	**93.7**	**1000**

TABLE 4 System performance on the independent held-out test set of 50 questions drafted by a non-author aviation professional.

Tool	Correct	CSR (%)
advisory	12/12	100
weather	16/17	94
live data	6/6	100
entity recognition	1/1	100
faa manuals	1/1	100
routes	5/5	100
runway configuration	3/4	75
trajectory prediction	3/4	75
Total	**47/50**	**94**

independently by an aviation professional not involved in the prompt tuning process. The system achieved 94% CSR on this independent set using GPT-4o, which is consistent with the results reported on the main dataset. These results suggest that the prompt engineering process did not overfit to the development data and that the system generalizes well to unseen queries from independent evaluators.

To assess the reliability of the reported results, 95% Wilson score confidence intervals (Wilson, 1927) were computed for each tool and reported in Table 5. The intervals confirm that performance differences across tools are statistically meaningful. For example, the weather tool CSR of 86.3% has a confidence interval of [79.4%–91.6%], which does not overlap with the advisory interval of [95.4%–99.9%], confirming that the gap between these two tools reflects a genuine difference in task difficulty rather than sampling variability. For tools with small gaps between the two models, such as weather (86.3% vs. 85.5%) and live data (91.6% vs. 90.1%), the confidence intervals overlap substantially, indicating that the difference between models is not statistically meaningful for these tools. The only tool where the model gap is supported by non-overlapping confidence intervals is runway configuration (GPT-4o: [87.2%–97.9%] vs. GPT-4o-mini: [68.5%–86.6%]), consistent with the finding that GPT-4o more reliably enforces the operational constraints of this tool.

Table 6 presents a structured taxonomy of failure modes observed during evaluation along with frequency estimates, severity rankings, and mitigation strategies. Wrong tool selection was the most operationally significant failure mode given its direct impact on response quality, but remained rare at 1.0% of tool-requiring queries. The most frequent failure was invoking a tool when the query should have been declined, occurring in 15.0% of no-tool and edge-case queries, primarily in the weather tool where general knowledge questions about meteorological concepts incorrectly triggered tool calls. Input extraction errors occurred in 6.2% of correct tool calls for tools with extractable parameters, most commonly due to station string mismatches in weather queries and ambiguous time references in runway configuration queries. The FAA Manuals tool showed a corpus coverage gap rate of 29.8%, where questions fell outside the indexed document collection, distinct from the 94.7% tool selection accuracy reported in Table 3. Failure modes related to stale data, cascading errors, and simulator command misinterpretation were not directly measurable in this evaluation and are acknowledged as important considerations for operational deployment.

4.1 Baseline evaluation

To contextualize the results of the tool-grounded system, we evaluated a baseline GPT-4o without tool access on a representative sample of questions from each tool category. The purpose of this evaluation is not to provide a direct numerical comparison, as the nature of each tool makes such a comparison either trivial or inapplicable, but rather to illustrate why tool grounding is necessary for each category of aviation query.

- `trajectory prediction`: Baseline GPT-4o consistently returned general aviation performance guidance including

TABLE 5 95% Wilson score confidence intervals for CSR across all tools and both models.

Tool	GPT-4o		GPT-4o-mini	
	CSR (%)	95% CI	CSR (%)	95% CI
`trajectory prediction`	97.7	[92.0–99.4]	93.1	[85.8–96.8]
`runway configuration`	94.7	[87.2–97.9]	78.9	[68.5–86.6]
`entity recognition`	100	[95.8–100]	100	[95.8–100]
`advisory`	99.2	[95.4–99.9]	96.6	[91.6–98.7]
`routes`	98.6	[95.1–99.6]	97.2	[93.1–98.9]
`weather`	86.3	[79.4–91.6]	85.5	[78.5–91.0]
`live data`	91.6	[85.8–95.1]	90.1	[84.1–94.0]
`faa manuals`	94.7	[88.1–97.7]	93.6	[86.8–97.0]
`blueSky`	93.9	[88.4–96.9]	88.5	[82.0–92.9]

TABLE 6 Error taxonomy for observed failure modes with severity rankings and mitigation strategies.

Failure mode	Frequency	Severity	Mitigation
Wrong tool selected	Low (1.0%)	High	Human prompt refinement
Tool called when it should be declined	Moderate (15.0%)	Medium	Scope constraints in tool description
Wrong inputs extracted	Low (6.2%)	Medium	Required parameter validation
FAA manuals corpus coverage gap	Moderate (29.8%)	Medium	Corpus expansion
Stale data returned	N/A	High	Real-time API cache invalidation
Cascading error in multi-tool query	Rare	High	Output validation between tool calls
Simulator command misinterpretation	Rare	High	BlueSky command schema validation

cruise altitude, cost index, and fuel estimates, but did not produce the CD0 and Kappa parameters required by the downstream trajectory prediction model. Since these parameters are outputs of a validated machine learning model trained on historical flight data, a plain LLM cannot produce equivalent results regardless of its general aviation knowledge.

- `runway configuration`: Baseline GPT-4o produced plausible runway recommendations based on wind alignment and general airport knowledge. However these recommendations were not grounded in the validated rule-based model and in several cases contradicted the model output. Since our tool only supports three specific airports and uses operational constraints not available to a general LLM, the baseline produces unverifiable outputs.
- `entity recognition`: The entity recognition tool was trained specifically to detect ATM operational concepts such as Traffic Management Initiatives, flow constraints, and facility identifiers from FAA telecon transcripts. Baseline GPT-4o performed a generic named entity recognition task instead, extracting airports, airlines, and times using standard NLP categories. The two outputs address fundamentally different information extraction objectives and are not directly comparable. The validated NLP model produces taxonomy-compatible structured output required by the downstream pipeline while baseline GPT-4o does not.
- `advisory`: All advisory queries require real-time access to the ATCSCC operations plan hosted at the FAA NAS status website, which is updated approximately every 2 hours. GPT-4o has no access to this data source and declined all queries, correctly acknowledging that it cannot provide current NAS status information.
- `routes`: GPT-4o either declined route queries citing the dynamic nature of preferred routes or produced general descriptions of typical ARTCC handoff sequences and common routing patterns. Neither response type contains actual FAA preferred route strings, which are retrieved directly from the FAA routes API and change based on current traffic management initiatives and weather.
- `weather`: All weather queries require real-time access to aviationweather.gov for current METARs, and TAFs. Baseline GPT-4o declined all queries and directed users to external sources.
- `live data`: To evaluate this tool, the same flight data used by the system was provided to GPT-4o as raw CSV context, giving

FIGURE 2
Use case 1: User requests the optimal runway configuration and current weather conditions at a specific airport. Our method first calls the `weather` tool to retrieve weather reports, then uses the `runway configuration` tool to determine the best runway configuration based on wind and operational factors.

both systems access to identical data. GPT-4o received the equivalent data as unstructured CSV and attempted to answer the same questions through natural language reasoning over the raw text. The results revealed a clear performance gap on queries requiring structured filtering and aggregation. Our system correctly answered questions such as counting flights above FL350, and identifying flights by wake turbulence category, and filtering arrivals by destination airport, while baseline GPT-4o consistently failed on these query types, by stating it could not categorize flights by altitude level or flight type from the unstructured data. For simple lookup queries both systems performed comparably when the requested flight was present in the dataset.

- `faa manuals`: This is the only tool where a meaningful content comparison is possible, as static regulatory knowledge is well represented in baseline GPT-4o training data. Baseline GPT-4o answered 95% of regulatory questions correctly compared to 74% for the RAG-grounded system. However GPT-4o produced one factual error, by incorrectly identifying the hijacking transponder code, which illustrates the hallucination risk in an operational context. Our system returned no-information-found responses when content was absent from its corpus rather than generating plausible but unverifiable answers, which is a safer behavior for a domain-specific decision support system.
- `BlueSky`: BlueSky queries require direct API access to the simulation environment to execute aircraft commands. Baseline GPT-4o has no connection to the BlueSky simulator and cannot issue or verify commands, making a content comparison inapplicable.

4.2 Use cases

The instances in which our method can be used for improving decision-making are demonstrated through a set of distinct scenarios, where the LLM-based method leverages either a single tool or a combination of tools from its repository to provide answers. Each figure highlights which tool was selected at each stage of the conversation. Figures 2–6 illustrate examples of different scenarios in which a user employs this method to assist in decision-making. Figure 2 illustrates a scenario in which the user asks two related questions: first about the current weather, and then about the optimal runway configuration at a specific airport. Our method initially calls the `weather` tool to retrieve the relevant weather information. The central LLM retains this weather information in memory and later leverages it when invoking the `runway configuration` tool to determine the best runway configuration based on wind and operational factors.

In another scenario, shown in Figure 3, the user requests the latest FAA advisory impacting Boston and the best route from MCO to BOS given current constraints. Our method first calls the `advisory` tool to retrieve the latest advisory information. The central LLM retains this advisory in memory and subsequently uses it when invoking the `routes` tool, to provide suggestions for the optimal flight paths. In another scenario, shown in Figure 4, a user requests custom APM parameters from an expert tool to support accurate trajectory prediction for an A319 arrival at DFW. Although the user did not explicitly ask about the weather, this method correctly calls the `weather` tool first to retrieve relevant weather inputs. It then invokes the `trajectory prediction` model to generate the APM parameters. Our method clearly lists all input

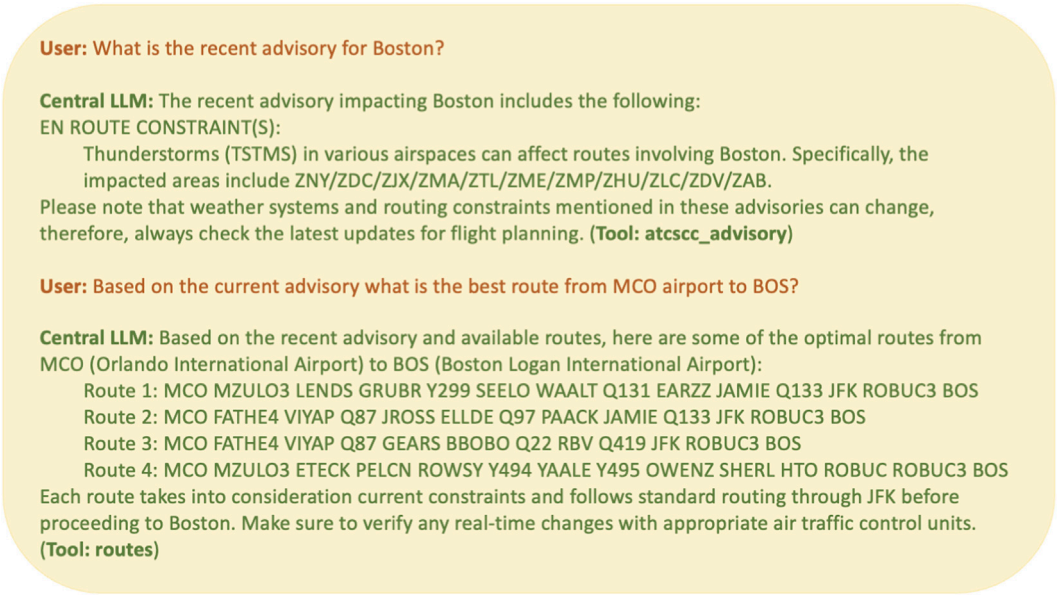

FIGURE 3
Use Case 2: User requests the latest FAA advisory impacting Boston and the best route from MCO to BOS based on current constraints. Our method first calls the `advisory` tool to obtain the latest advisories, then uses the `routes` to suggest optimal flight paths considering real-time impacts.

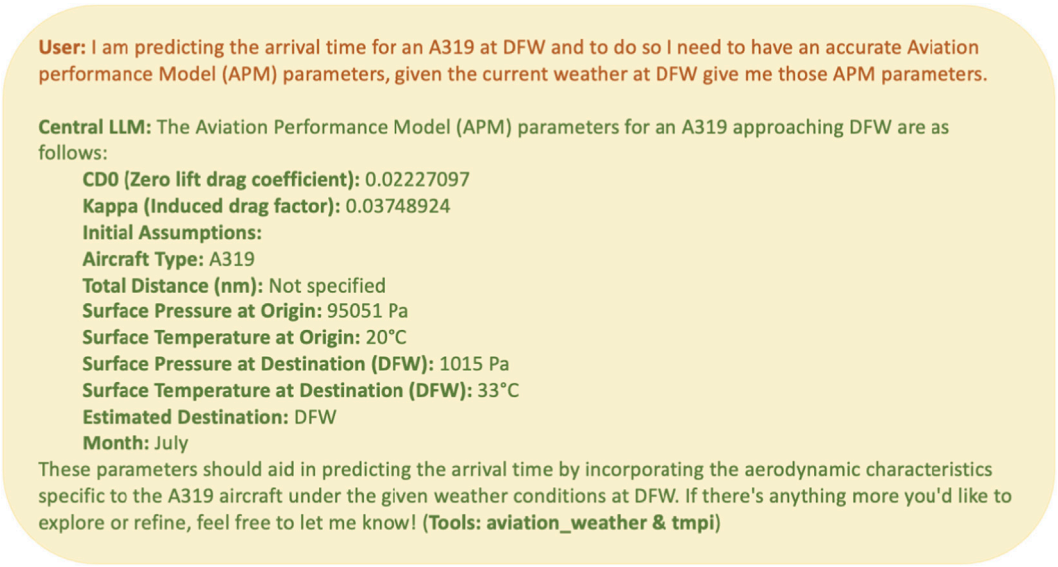

FIGURE 4
Use Case 3: A user requests custom Aviation Performance Model (APM) parameters from an expert tool to support accurate trajectory prediction for an A319 arrival at DFW. Our method first invokes the `weather` tool to retrieve the necessary weather inputs, followed by a call to the `trajectory prediction` model to generate the APM parameters.

assumptions derived from the provided information, such as aircraft type, destination airport, and current and forecasted weather, while leaving unspecified options, like total distance (which requires the origin airport), empty.

Figure 5 presents a case where a user requests the entities listed in a traffic management advisory by providing its raw text. Our method uses the `entity recognition` model to extract and provide the available entities from the advisory text and displays both the entity types and their corresponding values from the text. Finally, Figure 6 demonstrates a scenario where the user asks for the total number of United Airlines (UAL) flights currently airborne. Our method first uses the `entity recognition` tool to identify that "UA" refers to United Airlines.

It then leverages its internal plugin to generate the correct SQL query, tailored to the live datasets and their specific column names. The query is executed on an internal PostgreSQL database that

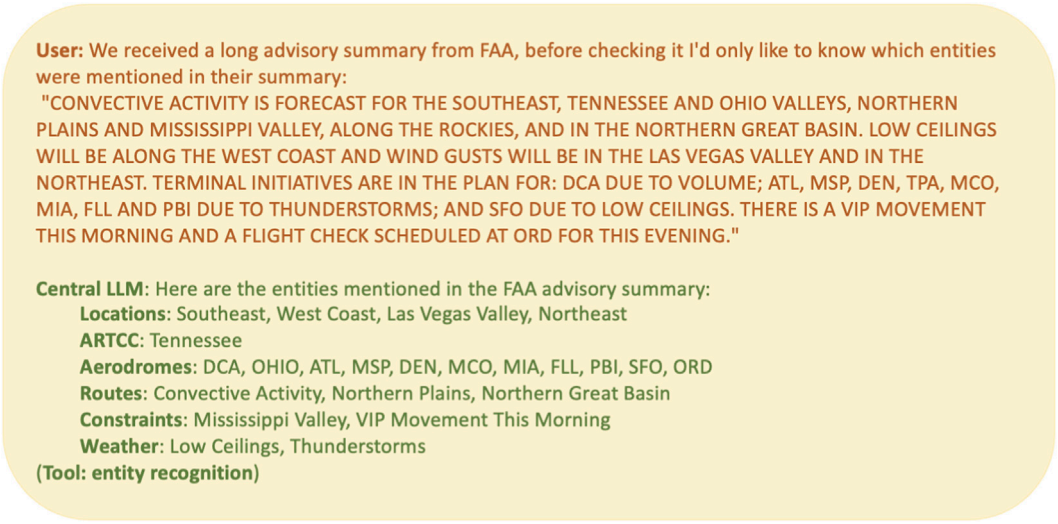

FIGURE 5
Use Case 4: A user requests for the entities listed in a traffic management advisory. The method picks `entity recognition` model to provide the available entities in the text.

refreshes every few seconds with live fuser data, and the results, including counts and sample records, are presented. In a subsequent question, the user requests to visualize one of these flights (UAL1646) in the BlueSky simulator. Our method instantiates the flight using all available live data associated with UAL1646 and runs it in the simulation, including issuing a direct-to-KDMS and descent-to-FL350 command. The BlueSky environment displaying UAL1646 is shown in the bottom picture.

5 Discussion

In this work, we developed a LLM-based method, designed for aviation by leveraging LLMs to answer domain-specific queries with verifiable evidence. In domain-specific fields such as aviation, the reliability and traceability of AI-generated outputs are essential. Rather than relying solely on generative responses, our method grounds its answers in outputs from trusted data sources and expert-approved tools to provide transparency and justification for each response. It integrates nine distinct tools, which entails both external APIs, datasets, simulation software, and models developed by domain experts in aviation domain. These tools span a diverse set of aviation-related functions, including air traffic management updates, weather data access, trajectory prediction parameters, runway configuration estimation, and real-time flight and route information. As demonstrated in our evaluation, our method achieves high performance across multiple metrics, including tool selection accuracy, and input fidelity. The results suggest that the system is capable of identifying the appropriate tools and returning accurate, supported responses across a broad spectrum of aviation query types. A key finding from the two-model comparison was that GPT-4o consistently outperforms GPT-4o-mini, with the performance gap being largest for tools that enforce strict operational constraints such as supported airports or aircraft types. For open-ended tools the two models perform comparably, suggesting that a smaller model may be sufficient depending on the deployment context.

The results demonstrate encouraging performance; however, more extensive evaluation is necessary to confirm these findings across broader datasets and use cases. The current evaluation dataset contains only 1000 examples. Future work should expand the dataset and introduce a dataset complexity metric to better characterize our method's performance under more challenging conditions. Another limitation is that certain tools are restricted to specific airports, aircraft types, or regions, and the system cannot extrapolate beyond those constraints. This restriction also contributed to some of the observed performance differences between the two models, as GPT-4o was more reliable at recognizing and declining out-of-scope queries while GPT-4o-mini was more likely to invoke the tool regardless.

Additionally, while a plain LLM baseline was not applicable for tools that rely on domain-specific models, we evaluated GPT-4o without tool access as a baseline across all nine tools. For tools requiring real-time data or domain-specific models the baseline produces trivially incorrect results, while for tools where meaningful comparison is possible such as FAA Manuals and live data, the tool-grounded system outperforms the plain LLM on structured queries. A rule-based or embedding-based tool selection baseline remains a direction for future evaluation. Beyond tool selection accuracy, CSR and ISA do not capture end-to-end correctness of the final user-facing answer. Validating whether a returned runway configuration is operationally valid given current winds and procedures, or whether a suggested route accounts for active traffic management initiatives, requires operational deployment with expert review of each tool output. The use cases presented in Figures 2–6 provide qualitative evidence of end-to-end correctness for representative scenarios, but a systematic end-to-end evaluation is acknowledged as an important direction for future work.

Moreover, our method was tested exclusively using GPT-based LLMs. Future work should investigate the system's

User: How many UA flights are in the sky?

Central LLM: There are currently 172 United Airlines (UA) flights in the sky. If you have any more questions or need further information, feel free to ask!
(**Tool: live data, entity recognition**, query: ":"SELECT COUNT(*) FROM flights WHERE carrier = 'UAL'"})

User: Show 5 of them with their altitude latitude and weight category.

Central LLM: Here are 5 United Airlines (UA) flights along with their altitude, latitude, and weight category:

- **Flight UAL1646**: Altitude: 350 ft, Latitude: 29.99454, Wake Turbulence Category: LARGE
- **Flight UAL2353**: Altitude: 3806.25 ft, Latitude: 29.88672, Wake Turbulence Category: LARGE
- **Flight UAL8189**: Altitude: 3050 ft, Latitude: 29.93405, Wake Turbulence Category: LARGE

(**Tool: BlueSky**)

User: Instantiate UAL1646 aircraft with its callsign, positions, heading, and speed on BlueSky.

Central LLM: UAL1646 has been instantiated in BlueSky (**Tool: BlueSky**).

User: UAL1646 proceed direct KDMS, descend and maintain FL350.

Central LLM: UAL1646 has been instructed to proceed direct KDMS and is descending to FL350 (**Tool: BlueSky**).

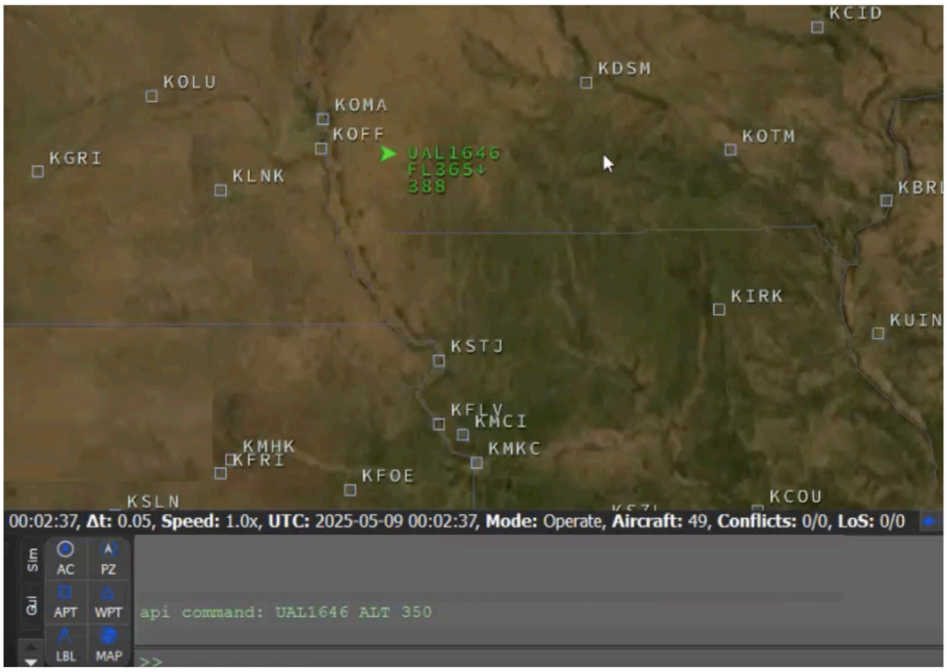

FIGURE 6
Use Case 5: The user asks for the number of United Airlines (UAL) flights currently airborne. The method calls `entity recognition` to extract entities from the request, queries the `live data` fuser to retrieve the count and sample records, displays five flights with altitude/latitude/wake-turbulence, instantiates flight UAL1646 in the `BlueSky` simulator, and issues a direct-to-KDMS and descent-to-FL350 command. The `BlueSky` simulation environment with UAL1646 is shown in the bottom picture.

performance using a wider range of LLMs, including open-source models, and evaluate robustness across diverse model architectures and deployment conditions. A systematic ablation of prompt design choices, including the impact of required parameter definitions and when-to-call language on tool selection accuracy, is left for future work, as it would require re-running the full evaluation with systematically varied prompt components. Latency and token cost were not systematically measured in this prototype evaluation and a thorough analysis of response time and cost tradeoffs between model sizes is left for future work, as these are important considerations for operational deployment. Future research will also focus on incorporating multimodal data sources and models to include more complex task executions by involving parallel tool calling, and developing an autonomous agent that monitors live data feeds. Such a method will proactively issue alerts and provide real-time updates based on its integrated aviation expertise and tool capabilities.

Data availability statement

The datasets used in this study are subject to NASA data release procedures. Requests for access should be directed to the corresponding author.

Author contributions

AS: Conceptualization, Data curation, Formal Analysis, Methodology, Software, Validation, Writing – original draft, Writing – review and editing. SC: Software, Writing – review and editing. AD: Supervision, Writing – review and editing.

Funding

The author(s) declared that financial support was not received for this work and/or its publication.

Acknowledgments

The authors would like to thank the MADI team for their valuable contributions, and the NASA Aeronautics Research Institute (NARI) for their guidance in this research.

Conflict of interest

The author(s) declared that this work was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declared that generative AI was not used in the creation of this manuscript.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Alanazi, F. (2025). From llm reasoning to autonomous ai agents: a comprehensive review. *arXiv Preprint*.

Banerjee, A.Singla (2024). *Llms Will Always Hallucinate, and We Need to Live with This.arXiv:2409*. 05746. Available online at: https://arxiv.org/abs/2409.05746.

Brown, M., and Ryder, S. (2020). "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, 33. Curran Associates, Inc., 1877–1901.

Chen, Z., and Langrene (2023). *Unleashing the Potential of Prompt Engineering in Large Language Models: A Comprehensive Review.arXiv:2310*. 14735. Available online at: https://arxiv.org/abs/2310.14735.

DIP (2025). Digital information platform (DIP). *Tech. Rep.* NASA.

Federal Aviation Administration (2024). Roadmap for artificial intelligence safety assurance Version 1. U.S. Department of Transportation, Federal Aviation Administration. Available online at: https://www.faa.gov/media/82891.

Johnson, and William, B. (2006). The human factors guide for aviation maintenance and inspection. Federal Aviation Administration. Available online at: https://www.faa.gov/sites/faa.gov/files/about/initiatives/maintenance_hf/training_tools/HF_Guide.pdf.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. (2022). Large language models are zero-shot reasoners. Advances in Neural Information Processing Systems. *Curran Associates, Inc.* 35, 22199–22213.

Lee, P. U., Brasil, C. L., Bakowski, D. L., Gabriel, C., Evans, M., and Chartrand, R. (2022). "Identifying common coordination procedures across extensible traffic management (xtm) to integrate xtm operations into the national airspace system," in *AIAA AVIATION 2022 Forum* (American Institute of Aeronautics and Astronautics). doi: 10.2514/6.2022-3910

Leeuwen, V., and Ellerbroek, H. (2016). "Bluesky atc simulator project: an open data and open source approach," in *7th International Conference for Research on Air Transportation (ICRAT)* (Philadelphia, USA).

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Adv. Neural Inf. Process. Syst.* 33, 9459–9474.

Li, F., Feng, S., Yan, Y., Lee, C.-H., and Ong, Y. S. (2024). *Virtual co-pilot: Multimodal Large Language model-enabled quick-access Procedures for Single Pilot Operations.arXiv:2403*. 16645

Puranik, R. S.Kalyanam (2022). "An ode-fitting approach to estimate critical aircraft performance parameters for trajectory prediction," in *Proceedings of the 2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)*. doi: 10.1109/DASC55683.2022.9925880

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *Tech. Rep.* (OpenAI). Available online at: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* 21, 1–67.

Rohani, S., Puranik, R. S., and Kalyanam, (2023). "Machine learning approach for aircraft performance model parameter estimation for trajectory prediction applications," in *Proceedings of the 2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)* (IEEE). doi: 10.1109/DASC58285.2023.10308938

Rohani, S., Nielsen, D.-G.Kalyanam (2025). "Contextualizing air traffic management conversations using natural language understanding," in *AIAA Scitech Forum* (American Institute of Aeronautics and Astronautics), 2025–2151. doi: 10.2514/6.2025-2151

Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., and Chadha, A. (2024). A systematic survey of prompt engineering in large language models: techniques and applications. *arXiv Preprint*. 2402.07927. doi: 10.48550/arXiv.2402.07927

Schick, T., Dwivedi-Yu, J., Dessi, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., et al. (2023). Toolformer: language models can teach themselves to use tools. Advances in Neural Information Processing Systems. *Curran Associates, Inc.* 36, 68539–68551.

Schlichting, M. R., Rasmussen, V., Alazzeh, H., Liu, H., Jafari, K., Hardy, A. F., et al. (2025). *Leraat: Llm-Enabled real-time Aviation Advisory Tool.arXiv:2503*. 16477.

Strobelt, W., S., Webson, A., Sanh, V., Hoover, B., Beyer, J., Pfister, H., et al. (2022). Interactive and visual prompt engineering for ad-hoc task adaptation with large language models. *IEEE Trans. Vis. Comput. Graph.* 29, 1146–1156. doi: 10.1109/TVCG.2022.3209426

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., et al. (2023). "Stanford alpaca: an instruction-following LLaMA model,". Stanford University. Available online at: https://github.com/tatsulab/stanford_alpaca.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., et al. (2023). *Llama: Open and Efficient Foundation Language Models.arXiv:2302*. 13971

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). "Attention is all you need," in *Advances in Neural Information Processing Systems*, 30. Curran Associates, Inc.

Wang, L., Chou, J., Zhou, X., Tien, A., and Baumgartner, D. M. (2023). *Aviationgpt: A Large Language Model for the Aviation Domain.arXiv:2311*. 17686

Wang, R., Han, X., Ji, L., Wang, S., Baldwin, T., and Li, H. (2025). *Toolgen: Unified Tool Retrieval and Calling via Generation.arXiv:2305*. 08886

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., et al. (2022). *Chain-Of-Thought Prompting Elicits Reasoning in Large Language Models*. arXiv:2201.11903

Wilson, E. B. (1927). Probable inference, the law of succession, and statistical inference. *J. Am. Stat. Assoc.* 22, 209–212. doi: 10.2307/2276774

Xu, J.Kankanhalli (2024). *Hallucination is Inevitable: An Innate Limitation of Large Language Models.arXiv:2401*. 11817

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., et al. (2023). "React: synergizing reasoning and acting in language models," in *International Conference on Learning Representations*.

Zhao, Y., Singh, P., Bhathena, H., Ramos, B., Joshi, A., Gadiyaram, S., et al. (2024). "Optimizing llm-based retrieval augmented generation pipelines in the financial domain," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 6. Industry Track, 279–294.

Zhou, L., Schellaert, W., Martínez-Plumed, F., Moros-Daval, Y., and Ferri, C. (2025). Larger and more instructable language models become less reliable. *Nat.* 634, 61–68. doi: 10.1038/s41586-024-07930-y