Iti-Validator: A Guardrail Framework for Validating and Correcting LLM-Generated Itineraries

Shravan Gadbail
International Institute of Information Technology
Hyderabad, India
shravan.gadbail@students.iiit.ac.in

Masumi Desai
International Institute of Information Technology
Hyderabad, India
masumi.desai@students.iiit.ac.in

Kamalakar Karlapalem
International Institute of Information Technology
Hyderabad, India
kamal@iiit.ac.in

Abstract

The rapid advancement of Large Language Models (LLMs) has enabled them to generate complex, multi-step plans and itineraries. However, these generated plans often lack temporal and spatial consistency, particularly in scenarios involving physical travel constraints. This research aims to study the temporal performance of different LLMs and presents a validation framework that evaluates and improves the temporal consistency of LLM-generated travel itineraries. The system employs multiple state-of-the-art LLMs to generate travel plans and validates them against real-world flight duration constraints using the AeroDataBox API. This work contributes to the understanding of LLM capabilities in handling complex temporal reasoning tasks like itinerary generation and provides a framework to rectify any temporal inconsistencies like overlapping journeys or unrealistic transit times in the itineraries generated by LLMs before the itinerary is given to the user. Our experiments reveal that while current LLMs frequently produce temporally inconsistent itineraries, these can be systematically and reliably corrected using our framework, enabling their practical deployment in large-scale travel planning.

CCS Concepts

• **Information systems** → **Spatial-temporal systems**; • **General and reference** → **Validation**; *Experimentation*; *Performance*; *Evaluation*; • **Computing methodologies** → **Planning and scheduling**; **Temporal reasoning**.

Keywords

Temporal Validation, Temporal Consistency, LLM Guardrail, LLM Validator, LLM-generated Itineraries, Flight API, Itinerary Correction

ACM Reference Format:
Shravan Gadbail, Masumi Desai, and Kamalakar Karlapalem. 2025. Iti-Validator: A Guardrail Framework for Validating and Correcting LLM-Generated Itineraries. In *The 33rd ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL '25), November 3–6, 2025, Minneapolis, MN, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3748636.3763206

SIGSPATIAL '25, Minneapolis, MN, USA

ACM ISBN 979-8-4007-2086-4/2025/11
https://doi.org/10.1145/3748636.3763206

1 Introduction

Recent advances in Large Language Models (LLMs) have enabled them to generate coherent and contextually relevant text across various domains, including travel planning. [1] A notable use case is automated itinerary generation, where LLMs produce multi-step travel plans from user prompts. While these plans may appear well-structured, they often overlook real-world constraints.

Temporal consistency—ensuring realistic transit times, and avoiding overlapping events—is a key but often ignored aspect. Without grounding in such constraints, LLMs can generate infeasible itineraries, such as scheduling simultaneous activities in different cities or suggesting impossible travel durations.

To address this limitation, we propose Iti-Validator, a guardrail framework designed to evaluate and correct LLM-generated travel itineraries. The framework leverages real-world travel data, specifically flight duration obtained from the AeroDataBox API—to verify the feasibility of generated plans. By validating and, if necessary, correcting these itineraries before presenting them to the user, Iti-Validator ensures that generated outputs not only read well but also adhere to realistic temporal constraints. While this work focuses on itineraries involving commercial flights, this scope is intentional: the commercial aviation sector is a trillion-dollar industry, and the ability to automatically generate and validate such itineraries has significant real-world applicability. Airlines, travel agencies, and itinerary planning platforms can directly integrate such a system to improve operational efficiency and customer satisfaction.
This work makes the following contributions:

- We conduct a systematic analysis of multiple state-of-the-art LLMs to assess their performance on temporal reasoning tasks within the domain of travel planning.
- We introduce a validation pipeline that integrates external knowledge sources (AeroDataBox for real-world travel feasibility) to detect and rectify temporal inconsistencies in generated itineraries.

2 Related Work

Recent work has explored the use of LLMs for travel itinerary generation, spatiotemporal reasoning, and path planning. **TripCraft** [2] presents a benchmark for LLM-based travel planning under realistic constraints such as transfer windows and transit durations. Similarly, **TravelPlanner** [8] introduces a planning challenge where

[1] This is the short version of our paper. The link to the long version with additional details will be available soon.

language agents must generate valid itineraries under multi-step logical and temporal constraints, highlighting how even state-of-the-art LLMs struggle without explicit constraint enforcement. **ItiNera** [7] integrates spatial clustering and optimization with LLMs to produce urban itineraries for real users, focusing on efficient routing within a city. **Geo-LLaMA** [5] tackles trajectory generation by injecting geospatial constraints into LLM outputs. These works primarily focus on optimizing or learning better itinerary generation within the LLM itself.

In parallel, a line of research examines the spatial and temporal reasoning abilities of LLMs more generally. For spatial path planning, **LLM-A*** [6] combines LLM reasoning with the A* algorithm to ensure valid navigation routes. **PPNL** [1] introduces a benchmark for spatial-temporal reasoning in natural language, while **Zhang et al.** [9] mitigate spatial hallucinations using reinforcement learning-enhanced prompting. For temporal reasoning, **ToT** [4] evaluates LLMs on logical time arithmetic tasks, and **TCP** [3] benchmarks collaborative project planning with interdependent temporal constraints.

Despite these advances, prior work either enforces constraints during generation or uses learned models to improve output quality. **In contrast, our approach introduces a post-processing temporal validator that operates independently of the LLM.** Unlike methods such as *ItiNera* or *Geo-LLaMA*, our validator requires neither fine-tuning nor integration with the LLM's internal architecture, making it model-agnostic. To the best of our knowledge, this is the first work to systematically validate and correct LLM-generated travel itineraries using real-world flight data accessed via an API (AeroDataBox) as an external knowledge source.

3 Methodology: Iti-Validator Framework

The proposed system implements a multi-stage validation and correction framework for travel itinerary generation. The architecture consists of four main components: (1) an LLM-based itinerary generator, (2) the AeroDataBox API integration for real-world flight time retrieval, (3) a validation mechanism that checks itinerary consistency against temporal rules, and (4) a correction mechanism that adjusts the itinerary to fix any detected issues. Figure 1 presents an overview of the system.

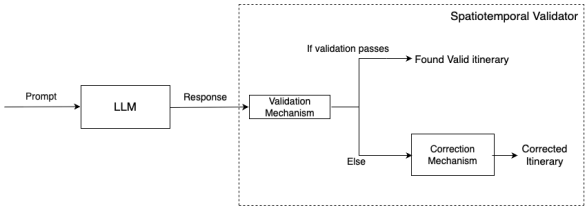

Figure 1: System architecture

3.1 Validation Rules

To evaluate an itinerary, the validator applies a set of rules which aim to account for real-world travel constraints:

- **No Overlap Rule:** A traveler cannot be in two places at the same time. The itinerary is invalid if any visit intervals overlap or if a departure time from one city precedes the arrival time to that same city.
- **Minimum Transit Time Rule:** Traveling between two distinct cities requires a minimum non-zero time. We assume all inter-city travel is by commercial aircraft (the fastest conventional mode for long-distance travel). For each pair of consecutive destinations, the validator fetches the minimum flight duration $t_{minimum}$ between those cities using the AeroDataBox API. If the itinerary allocates less travel time than $t_{minimum}$ between a departure and the next arrival, that segment is marked invalid.
- **Maximum Transit Time Rule:** An itinerary leg should not contain excessively more time than needed. We set an upper bound that travel time should not exceed $2 \times t_{minimum}$ (double the minimum flight duration, to accommodate layovers or realistic scheduling buffers). If an LLM allocates an extremely large gap for a flight (e.g., several days for a 10-hour flight), that segment is also flagged as invalid.
- **Minimum Stay Rule:** Each destination should have a minimum stay duration (we use two days in our experiments, unless specified otherwise). This ensures the itinerary is realistic for a multi-city tour, avoiding itineraries where a traveler lands and departs immediately with no time spent.

The validator parses the LLM's JSON itinerary (which includes location names and timestamped arrival/departure for each stop) and checks each leg against these rules. If all segments satisfy the rules, the itinerary is labeled *valid*. If any rule is violated, the itinerary is *invalid* and needs correction.

While our current implementation assumes international travel by commercial flights, the validation rules are modular and easily extendable. For instance, adding trains or road travel would involve incorporating corresponding minimum and maximum transit times from relevant datasets. Similarly, domain-specific rules (e.g., budget limits, visa constraints, specific dates) can be layered onto the framework without altering its core structure.

3.2 Correction Mechanism

For each segment that fails validation, our framework automatically adjusts the schedule to fix the issue:

- If a segment's allocated travel time is too short (violating the Minimum Transit Time Rule), we push the arrival time at the destination forward. Specifically, for a segment where city A's departure time is t^A_{dep} and city B's arrival time is t^B_{arr}, if $t^B_{\text{arr}} - t^A_{\text{dep}} < t_{min}(A, B)$, we set

$$t^B_{\text{arr}} := t^A_{\text{dep}} + t_{min}(A, B)\,,$$

  effectively delaying the visit to city B to ensure enough transit time.
- If a segment's travel time is excessively long ($t^B_{\text{arr}} - t^A_{\text{dep}} > 2 \times t_{min}$), we pull the arrival time earlier to a reasonable window. We cap it at $2 \times t_{min}$ after departure.

- If an overlap is detected (city B's arrival precedes city A's departure), we shift the subsequent schedule forward in time until no overlaps remain.
- If a city's stay is shorter than the minimum, we extend the departure time from that city to meet the 2-day threshold.

After applying these modifications to all problematic segments, the itinerary is checked again to ensure all constraints are satisfied. In practice, a single pass of adjustments resolves all conflicts in our test cases.

We also explored an alternative correction approach: having the LLM itself regenerate the itinerary after being told what issues to fix. The validator can produce a corrective prompt listing each problem (e.g., "Flight from X to Y is too short, needs at least N hours; extend stay in Z to 2 days") and ask the LLM to produce a new itinerary. *We tried this iterative re-prompting (up to three attempts) for the three LLMs on 4-city itineraries. However, this approach proved unreliable—many itineraries remained invalid even after multiple retries, as the LLMs would fix some issues but introduce or leave others.* Given these inconsistent results, we discarded the self-correction approach in favor of deterministic rule-based adjustments, which guarantee a valid itinerary whenever one is feasible.

3.3 Implementation Details

LLM Itinerary Generation: We experiment with three LLMs to generate itineraries, allowing us to compare their baseline performance and illustrate the need for validation. The models are: (1) OpenAI's **GPT-4o-mini**, (2) Google's **Gemini-2.0** (flash-thinking experimental preview), and (3) Meta's **LLaMA2-7B** (7-billion-parameter open-source model). Each model is prompted to produce a multi-city trip itinerary in JSON format, listing each destination with arrival and departure timestamps (UTC, 24-hour format). We explicitly instruct a minimum two-day stay per city in the prompt to enforce the stay rule at generation time, though the models do not always comply.

AeroDataBox API Integration: We use the AeroDataBox service to obtain realistic flight durations between city pairs. Given two cities (identified by major airport codes), the API returns typical flight times. The returned duration is parsed (hours and minutes) and converted to seconds for comparison. We add a fixed buffer (4 hours) to each flight duration as the minimum required travel time to account for airport logistics. If the API fails or no data is available for a route, we retry the API call, upto a maximum of 3 times. All API queries are cached to avoid redundant requests when validating multiple itineraries with common segments. In cases where the API fails repeatedly or returns null data, our framework proceeds only with non-null results to ensure that all validations are based on reliable flight duration information. This approach maintains the robustness of the system even when certain route data is unavailable.

The overall process is: ① LLM generates itinerary ② validator checks each segment via AeroDataBox and the rules ③ if any segment violates a rule, adjust travel times ④ output the corrected itinerary (or report it as valid if no changes needed). If an itinerary had format-related issues (e.g., JSON-formatting, date-time formatting), the framework would retry the generation of the itinerary by re-prompting, upto a maximum of three retries, the LLM by specifying the exact issue along with the base prompt. [2]

4 Evaluation Setup and Metrics

We evaluated the LLMs' raw itinerary outputs using multiple metrics. Each LLM was tasked with generating itineraries for trips with a varying number of destinations (we tested 4, 5, and 6 destination trips, which equates to 3, 4, and 5 flights per itinerary respectively). The cities were chosen to ensure both short and long international flights, covering a wide range of cases, with examples including routes across North America, Asia, Europe, and the Middle East.

For each combination of model and number of cities, we collected multiple itineraries (100 per combination). We then applied the validator to each itinerary. To quantify performance, we use:

- **% Invalid Itineraries (raw):** The percentage of LLM-generated itineraries that contained one or more temporal inconsistencies before any correction. This indicates how often the LLM failed to produce a fully feasible plan.
- **% Invalid Segments (raw):** Out of all individual travel segments (flight legs) in the itineraries, the percentage that violated constraints (either too short or too long). This measures how frequently an LLM misjudges a single leg's timing.
- **Avg Issues per Itinerary:** The average number of issues (invalid segments) detected per itinerary. An itinerary with multiple problematic legs would increase this count.

Since our validator corrects any detected issue, the post-validation itineraries were all consistent. Thus, our evaluation focuses on the severity of issues in the raw output and how much correction was needed.

5 Results and Analysis

LLM Model	Cities	Invalid Itin.	Invalid Seg.	Avg Issues/Itn.
Gemini-2.0†	4	**48**%	**21.00**%	**0.63**
GPT-4o-mini	4	97%	78.00%	2.34
LLaMA2-7B	4	100%	92.33%	2.77
Gemini-2.0†	5	**68**%	**27.75**%	**1.11**
GPT-4o-mini	5	100%	82.00%	3.28
LLaMA2-7B	5	100%	93.69%	3.75
Gemini-2.0†	6	**76**%	**29.60**%	**1.48**
GPT-4o-mini	6	99%	78.00%	3.90
LLaMA2-7B	6	100%	91.60%	4.58

Table 1: Temporal inconsistencies in raw LLM-generated itineraries, before applying our validator. "Invalid Itin." is the percentage of itineraries with any temporal inconsistencies. "Invalid Seg." is the percentage of flight segments that violate constraints. "Avg Issues/Itn." is the average number of issues (segments or stays) per itinerary.((†) Refers to the Gemini-2.0 variant: gemini-2.0-flash-thinking-exp-01-21.)

Table 1 summarizes the consistency of raw itineraries across models and destination counts. Our deterministic, rule-based validator corrected 100% of invalid itineraries. Each LLM was tested on 100 itineraries with 4, 5, and 6 destinations.

[2]The code and detailed experiment results are available at the following link: https://github.com/MasumiD/Spatiotemporal-Validator-for-LLMs.

We find that all models show growing temporal inconsistency as itinerary complexity increases. **LLaMA2-7B** performed the worst, with 100% invalid itineraries across all settings and over 90% of segments violating constraints. Its average issues peaked at 4.58 per itinerary for 6 cities. Since the number of flight segments equals one less than destinations, a 6-city trip contains 5 segments.

GPT-4o-mini performed slightly better but still produced 97–100% invalid itineraries, with up to 3.90 issues per 6-city plan. **Gemini-2.0 (Flash)** fared best, with only 48% invalid itineraries for 4 cities and 1.48 issues on average for 6 cities. However, its performance also degraded with more destinations, showing no model is inherently grounded in travel constraints.
Distinct error modes emerged:

- **Underestimating Travel Time:** GPT-4o-mini and Gemini often allocated insufficient time, e.g., giving only 8 hours for the Tokyo–Dubai leg (about 3 hours too short). This explains why invalid segment percentages were high but not 100%.
- **Overestimating Travel Time:** LLaMA2-7B consistently overshot, sometimes adding multi-day gaps. For example, it scheduled a 5-day Paris–Tokyo flight, violating maximum transit rules. Hence, all itineraries were infeasible.
- **Minimum Stay Violations:** Despite prompts for a 2-day stay, some itineraries had shorter durations (e.g., Gemini scheduled 1.7 days in Dubai). Less frequent but still problematic.

After validation, **all errors were eliminated**, dropping "Invalid Itin." percentages to 0%. The validator fixed all detected timing conflicts, while minor format issues (e.g., missing arrival fields) were resolved with feedback-based regeneration.
We also found LLMs lack reliable knowledge of flight networks. Even when prompted for realistic routes, models suggested nonexistent flights (e.g., Surat–Doha direct). This shows they are not grounded in actual flight data.
These findings highlight the importance of external validation for ensuring feasibility in LLM-generated itineraries.

6 Conclusion and Future Work

We introduced **Iti-Validator**, a guardrail framework to verify and correct travel itineraries generated by Large Language Models (LLMs). By cross-checking LLM outputs with real-world flight durations and enforcing simple temporal rules, our system transforms often-impractical AI-generated travel plans into feasible itineraries. Through experiments on itineraries from multiple LLMs, we demonstrated that temporal inconsistencies are frequent in raw outputs, and that our validator is effective in detecting and eliminating these issues. This post-processing approach significantly enhances the reliability of LLM-based travel planning without requiring any changes to the LLMs themselves.

From an efficiency perspective, the guardrail operates within a few seconds per itinerary, dominated by API calls and correction logic. All corrections are deterministic and scale linearly with the number of itinerary legs. Since the AeroDataBox API calls are cached, repeated city pairs incur negligible overhead. Prompting costs for LLM regeneration (in cases of formatting errors) were minimal in our experiments, as most retries succeeded on the first attempt. These results suggest that Iti-Validator is both scalable and cost-effective for real-world deployment.

This work is a step toward safer and more trustworthy AI planning tools. In future work, we plan to extend the validator to handle a wider range of scenarios—such as multi-modal transport (integrating trains or driving legs) and more complex user preferences or constraints (e.g., maximum budget, specific layover cities). Another direction is to develop the validator as a more interactive assistant: it could provide explanations for its corrections or integrate with the LLM in a dialogue (beyond one-shot prompts) to iteratively refine itineraries with the user in the loop. Lastly, while our rule-based method outperformed autonomous LLM re-prompting for corrections, a hybrid approach could be explored: using learning to predict when a simple time-shift fix is insufficient and a larger itinerary restructuring is needed. We anticipate that our guardrail concept can be applied broadly to other domains where LLMs generate structured plans that must adhere to real-world constraints.

References

[1] Mohamed Aghzal, Erion Plaku, and Ziyu Yao. 2023. Can Large Language Models be Good Path Planners? A Benchmark and Investigation on Spatial-Temporal Reasoning. *arXiv preprint arXiv:2310.03249* (2023). https://arxiv.org/abs/2310.03249

[2] Soumyabrata Chaudhuri, Pranav Purkar, Ritwik Raghav, Shubhojit Mallick, Manish Gupta, Abhik Jana, and Shreya Ghosh. 2025. TripCraft: A Benchmark for Spatio-Temporally Fine-Grained Travel Planning. In *Proc. 63rd Annual Meeting of the Assoc. for Computational Linguistics (ACL 2025)*. –. https://arxiv.org/abs/2502.20508

[3] Zifeng Ding, Sikuan Yan, Zhangdie Yuan, Xianglong Hu, Fangru Lin, and Andreas Vlachos. 2025. TCP: A Benchmark for Temporal Constraint-Based Planning. *arXiv preprint arXiv:2505.19927* (2025). https://arxiv.org/abs/2505.19927

[4] Bahare Fatemi, Mehran Kazemi, Anton Tsitsulin, Karishma Malkan, Jinyeong Yim, John Palowitch, Sungyong Seo, Jonathan Halcrow, and Bryan Perozzi. 2024. Test of Time: A Benchmark for Evaluating LLMs on Temporal Reasoning. *arXiv preprint arXiv:2406.09170* (2024). https://arxiv.org/abs/2406.09170

[5] Siyu Li, Toan Tran, Haowen Lin, John Krumm, Cyrus Shahabi, Lingyi Zhao, Khurram Shafique, and Li Xiong. 2024. Geo-Llama: Leveraging LLMs for Human Mobility Trajectory Generation with Spatiotemporal Constraints. *arXiv preprint arXiv:2408.13918* (2024). https://arxiv.org/abs/2408.13918

[6] Silin Meng, Yiwei Wang, Cheng-Fu Yang, Nanyun Peng, and Kai-Wei Chang. 2024. LLM-A*: Large Language Model Enhanced Incremental Heuristic Search on Path Planning. In *Findings of the Assoc. for Computational Linguistics: EMNLP 2024*. 1087–1102. doi:10.18653/v1/2024.findings-emnlp.60

[7] Yihong Tang, Zhaokai Wang, Ao Qu, Yihao Yan, Zhaofeng Wu, Dingyi Zhuang, Jushi Kai, Kebing Hou, Xiaotong Guo, Jinhua Zhao, Zhan Zhao, and Wei Ma. 2024. ItiNera: Integrating Spatial Optimization with Large Language Models for Open-domain Urban Itinerary Planning. In *Proc. EMNLP 2024, Industry Track*. 1413–1432. doi:10.18653/v1/2024.emnlp-industry.104

[8] Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. TravelPlanner: A Benchmark for Real-World Planning with Language Agents. In *Proc. Int. Conf. on Machine Learning (ICML 2024)*. https://arxiv.org/abs/2402.01622

[9] Hongjie Zhang, Hourui Deng, Jie Ou, and Chaosheng Feng. 2025. Mitigating Spatial Hallucination in Large Language Models for Path Planning via Prompt Engineering. *Scientific Reports* 15 (2025), 8881. doi:10.1038/s41598-025-93601-5