Enabling On-Premises Large Language Models for Space Traffic Management

Enrique De Alba
EO Solutions
enrique.dealba@e-o.solutions

Abstract

Natural language processing systems leveraging on-premises large language models (LLMs) can translate natural language into structured JSON commands for Space Traffic Management (STM) systems. While cloud-based LLMs excel at this task, security constraints necessitate local deployment, requiring evaluation of smaller on-premises models. We demonstrate that resource-efficient 7B-parameter models can achieve high accuracy for STM command generation through a two-stage pipeline. Our pipeline first classifies objectives, then generates schemas. Empirically, we observe that initial classification accuracy strongly influences overall performance, with failures cascading to the generation stage. We demonstrate that quantization disproportionately increases structural errors compared to semantic errors across 405 objectives. The best quantized model (Falcon3-7B-GPTQ) shows a 3.45% accuracy drop, primarily from structural errors. Our findings highlight limitations in how model compression affects applications that require syntactic validity. More broadly, we explore the feasibility of LLM deployment in air-gapped environments while uncovering how quantization asymmetrically impacts structured output generation.

Our code is available at:
https://github.com/enrique-dealba/LLM-STM.

1 Introduction

The proliferation of satellites and the global democratization of launch services have transformed Earth orbit into a densely populated arena (The Aerospace Corporation, Space Safety Institute, 2024; United Nations Office for Outer Space Affairs, 2025). Existing Space Traffic Management (STM) tools require specialized expertise. In particular, hand-authoring simulator-specific JSON schemas creates a bottleneck that slows response times (Fletcher and Kadan, 2024; De Alba et al., 2024). Building on natural-language interfaces for collaborative telescope networks (Fletcher et al., 2021), we introduce a framework that translates operator intent from everyday language into machine-interpretable STM commands, potentially lowering entry barriers and enabling delegation to autonomous agents.

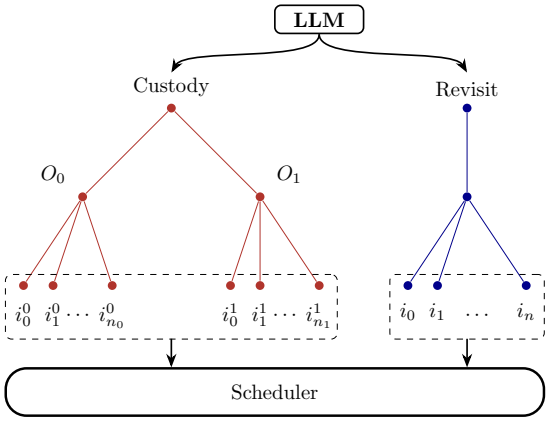

Figure 1: System architecture where the LLM interprets natural language to generate structured objectives (e.g., Custody, Revisit, etc.). Objectives undergo hierarchical decomposition, potentially via sub-objectives (O_k), producing sets of atomic intents (i_j^k). These intents collectively form the input specification for the scheduler.

We enable natural-language interaction for STM within secure operational contexts, using on-premises large language models (LLMs) to translate user objectives into structured commands that initiate the decomposition and scheduling workflow in Figure 1. Our contributions can be summarized as follows: (1) we show that 7B-parameter models achieve $>90\%$ accuracy for STM command generation through a two-stage pipeline, (2) we identify cascading failures where initial classification errors prevent access to strong schema-filling capabilities, and (3) we find that quantization disproportionately

Proceedings of Recent Advances in Natural Language Processing,pages 260–267
Varna, Sep 8–10, 2025
https://doi.org/10.26615/978-954-452-098-4-032

degrades structural schema comprehension compared to semantic understanding, impacting compressed model deployment in applications. While smaller open-source models present trade-offs following power-law scaling (Kaplan et al., 2020), they run on single GPUs, keeping information on-premises (Omnifact, 2025). We evaluate whether prompt engineering and multi-step parsing enable structured output generation.

2 Related Work

Recent work suggests that LLM hidden states encode truthfulness signals that can be extracted to detect errors (Orgad et al., 2025), with correct generations exhibiting measurably sharper context activations than hallucinated outputs (Chen et al., 2024). The internal layers of models contain more information than their final outputs suggest. Structured outputs enhance reliability. By constraining model responses to predefined formats like JSON schemas, outputs become interpretable and verifiable. OpenAI uses constrained decoding that guarantees schema conformity (OpenAI, 2024), while open-source tools like Instructor leverage Pydantic (Colvin et al., 2024) to validate and correct LLM-generated JSON (Liu, 2024). Similar structured generation challenges arise in model-based systems engineering (Crabb and Jones, 2024). We implement structured specifications similar to intent-based approaches for autonomous sensor orchestration (Fletcher and Kadan, 2024). This builds upon established frameworks for situation awareness in autonomous agents (Dahn et al., 2018) and leverages concepts from observation planning systems (Morris et al., 2018). We employ structured generation for interpretability and error mitigation. On-premises deployment requires addressing memory constraints. Key-value (KV) caches bottleneck memory in LLM inference (Zhang et al., 2024). PagedAttention (Kwon et al., 2023) addresses this. We also use GPTQ quantization (Frantar et al., 2023) to reduce model weights. Our quantized Falcon3-7B uses 18GB of memory. This enables on-premises STM deployment of models up to 22B parameters.

3 STM Objective Specification

We translate natural-language objectives into machine-interpretable specifications for STM operations. The architecture comprises an LLM-driven parser and utilities for output post-processing and validation. Figure 2 shows an example input. Each STM objective in our test set contains 7–32 fields (mean: 15.9) with prompt lengths ranging from 70–367 tokens. This input is fed to the LLM along with a designed prompt that instructs the model to output a structured representation of the objective, treating prompt engineering as a form of specialized programming that transforms natural language to structured outputs (Beurer-Kellner et al., 2023). In our implementation, the structured format is a JSON schema that captures the essential parameters of the task (e.g., type of objective, targets, time window, required outputs, etc.). The LLM serves as the translation engine from unstructured language to a structured representation. We validate outputs against the predefined schema. This end-to-end flow, User NL Objective $\rightarrow$ LLM $\rightarrow$ JSON output $\rightarrow$ Validation, forms the core data pipeline. Valid JSON objectives are passed to the objective decomposition and scheduler modules depicted in Figure 1; invalid JSON triggers re-prompting or fixes.

Example Prompt

Track RSO target 28884 with the sensors SENSOR-01 and SENSOR-02 in TEST mode for a 36 hour plan, schedule four periodic revisits per hour, and use U markings. Please use RATE_TRACK_SIDEREAL tracking, set the priority to 3, and set patience to 30 mins. Start at **2025-03-13 15:30:00**, end at **2025-03-14 22:30:00**.

Figure 2: This prompt is an example of the test cases used in the evaluation, demonstrating specific field values (e.g., exact timestamps "2025-03-13 15:30:00", sensor names "SENSOR-01", etc.). This ensures unambiguous one-to-one correspondence between prompts and expected schemas, eliminating interpretation variability and enabling precise accuracy measurement. If the system fails to produce exactly what was requested, it indicates a clear error.

3.1 Model Deployment and Memory Constraints

Deploying models like Falcon3-7B and Mistral-7B on local hardware requires optimization for GPU memory constraints. To enhance efficiency, we implement GPTQ quantization, reducing model weights to 4-bit precision with minimal quality impact (Xiong et al., 2024), enabling our quantized Falcon3-7B to process longer STM objec-

JSON Schema Instructions

```
The output should be formatted as a JSON
instance that conforms to the JSON schema
below.
As an example, for the schema
{"properties":
  {"foo":
    {"title": "Foo",
    "description": "a list of strings",
    "type": "array",
    "items": {"type": "string"}
    }
  }, "required": ["foo"]
}
the object {"foo": ["bar", "baz"]} is a
well-formatted instance of the schema.
The object {"properties": {"foo": ["bar",
"baz"]}} is not well-formatted.
Here is the output schema: ...
```

Figure 3: This instruction snippet is the default prompting strategy we use for structured outputs, providing explicit formatting guidelines that include JSON schema examples and clarifying valid versus invalid formatting.

tives while occupying only 18GB VRAM. The PagedAttention algorithm partitions the KV cache into fixed-size blocks of size B, enabling non-contiguous memory allocation. For each query token i, attention is computed block-wise as:

$$A_{ij} = \frac{\exp(q_i^\top K_j/\sqrt{d})}{\sum_{t=1}^{\lceil i/B \rceil} \exp(q_i^\top K_t/\sqrt{d})}, \; o_i = \sum_{j=1}^{\lceil i/B \rceil} A_{ij} V_j$$

where K_j and V_j represent blocks of key and value vectors, and A_{ij} denotes attention scores between query i and block j.

3.2 Structured Output Pipeline

A distinguishing feature of our system is the structured output pipeline that ensures the model's response adheres to a specified JSON schema. Rather than prompting the LLM in an unconstrained way, we embed instructions in the prompt that explicitly require a JSON-formatted response with certain fields. As shown in Figure 3, we provide explicit instructions for JSON formatting. This improves format compliance and reasoning quality on diverse tasks (Kurt, 2024). However, we do not rely on the model itself to guarantee validity. After the LLM generates its output, we pass the result to a JSON schema validator (implemented via Pydantic models in Python) which checks that all required fields are present and of the correct type. This is analogous to the OpenAI structured output approach (OpenAI, 2024) which enforces compliance with a schema at generation time. Since our models do not natively support constrained decoding, we implement a robust post-generation validation framework similar to approaches that have shown high schema compliance in recent benchmarks (GuidanceAI, 2023). Our approach utilizes a two-stage parsing mechanism that implements principles from grammar-constrained decoding without requiring model fine-tuning (Geng et al., 2023). First, we extract and normalize the JSON from the model's response, employing regex-based techniques to identify JSON blocks within markdown-formatted text. Second, we validate the extracted JSON against a predefined schema using Pydantic models. For handling malformed outputs, we use a partial JSON parser that can recover from common structural errors. This parser incrementally processes each character in the response, maintaining a stack of expected closing delimiters, and can handle incomplete JSON objects, unclosed quotes, and missing brackets, similar to finite-state machine-based decoding techniques that enforce strict adherence to predefined formats (Willard and Louf, 2023). When validation fails, our system utilizes a structured exception handling mechanism that preserves both the original output and the specific validation error details.

4 Experimental Evaluation

We evaluate our on-premises STM objective parser focusing on two objectives: (1) measuring the accuracy of on-premises LLMs in translating natural-language STM objectives into structured JSON outputs, and (2) determining the impact of model architecture, size, and memory optimizations on performance and interpretability. Our experiments use an A100 80GB GPU.

4.1 Experimental Setup

Our two-stage pipeline first identifies the objective type from natural language, then generates a detailed JSON output with all required fields populated based on the identified type. Our benchmark suite comprises 405 diverse objective prompts, designed to span nine distinct STM objective categories (e.g., space object tracking, sensor orchestration, search). Each prompt in this set defines specific, unambiguous field values to ensure precise accuracy measurement. We evaluate nine large language models selected to cover a wide

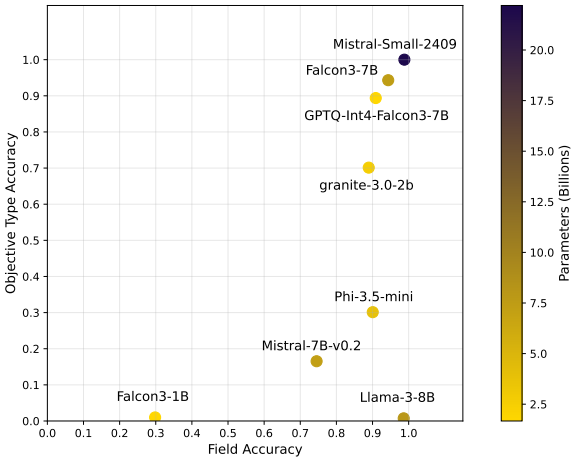

Figure 4: LLMs evaluated on 405 STM tasks from natural language to JSON. While larger models perform best overall, 7B-parameter models achieve competitive accuracies. Notably, Llama-3-8B and Mistral-7B show high field accuracy despite poor objective classification, highlighting the importance of accurate initial classification in the two-stage pipeline.

range of parameter scales and architectures. The evaluated models include: Mistral-Small-Instruct-2409 (22.2B) (Mistral AI Team, 2024); Falcon3-7B (7.46B) and its 4-bit quantized version Falcon3-7B-GPTQ; Falcon3-1B (1.67B) (Team, 2024); Phi-3.5-mini-instruct (3.82B) (Abdin et al., 2024); Llama-3-8B-Instruct (8.03B) (AI@Meta, 2024); Mistral-7B-Instruct-v0.2 (7.24B) (Jiang et al., 2023); and Granite-2B (2.63B) (Granite Team, 2024).[1] The quantized Falcon3-7B-GPTQ has a memory footprint comparable to a 1.7B-parameter model. We also tested an additional smaller model, Qwen-2.5-0.5B (0.494B) (Yang et al., 2024), which achieved 0% on both objective type and field accuracy metrics with our current prompting strategy. This result establishes a practical lower bound for model capability on this complex task, and we have omitted Qwen-2.5-0.5B from comparative plots. All evaluations for the nine primary models utilize identical, carefully constructed prompts (see Figure 3), a consistent temperature setting of 0.2 to encourage deterministic outputs, and uniform validation criteria to ensure fair and rigorous comparison.

4.2 Evaluation Metrics and Processing

We employ two primary quantitative metrics to assess performance across the 405 test cases: (1) Objective Type Accuracy: This measures the efficacy of the first pipeline stage. It is defined as the percentage of test cases where the LLM correctly identifies the true STM objective type from a set of nine predefined objective types. (2) Field Accuracy: This evaluates the second pipeline stage. It is calculated as the percentage of correctly populated fields within the generated JSON objects, contingent upon the JSON being validated against the ground truth or correctly identified objective schema. Correct population considers both the field's value and its data type. To ensure a fair assessment of Field Accuracy, we implement custom normalizers for common variations that do not alter semantic meaning. These include normalization for datetime string formats (e.g., ISO 8601 vs. other representations), numerical comparisons with a defined tolerance, and order-invariant comparison for lists of items. Furthermore, we report slot presence F1 scores, derived from precision and recall of expected fields, to offer a nuanced view of schema population completeness. When JSON responses fail Pydantic validation, our system either re-prompts or applies rule-based fixes for common structural errors (mismatched brackets, unclosed quotes). A partial JSON parser recovers data from malformed outputs by maintaining a stack of expected closing delimiters, enabling extraction from partially correct responses.

[1]For brevity, "Instruct" suffixes are often omitted. Thus, "Mistral-Small-2409" refers to "Mistral-Small-Instruct-2409".

Model	Obj Type Acc. (%)	Field Acc. (%)
Mistral-Small-2409	100.00	98.81
Falcon3-7B	94.32	94.34
Falcon3-7B-GPTQ	89.38	90.89
Granite-2B	70.12	88.95
Llama-3-8B	0.74	98.63
Mistral-7B-v0.2	16.54	74.52
Phi-3.5-mini	30.12	90.06
Falcon3-1B	0.99	29.82
Qwen-2.5-0.5B[†]	0.00	0.00

Table 1: Objective Type Accuracy and Field Accuracy (%) across 405 STM objectives for on-premises LLMs (see Fig. 4). Field Accuracy is computed conditional on correct objective type and schema-valid JSON. [†]Qwen-2.5-0.5B yielded negligible accuracy and was omitted from comparative plots.

4.3 Results and Analysis

Table 1 summarizes performance across 405 STM objectives (see also Fig. 4). Mistral-Small-2409 attains 100.00% Objective Type Accuracy and 98.81% Field Accuracy. Falcon3-7B reaches 94.32% Objective Type Accuracy and 94.34% Field Accuracy, approaching Mistral-Small-2409

with approximately one-third of the parameters. The GPTQ version of Falcon3-7B yields 89.38% Objective Type Accuracy and 90.89% Field Accuracy, a 3.45 percentage-point reduction in Field Accuracy relative to the full-precision model. Granite-2B obtains 70.12% Objective Type Accuracy and 88.95% Field Accuracy, indicating that model architecture and training methodology may outweigh raw parameter count. Llama-3-8B exhibits 98.63% Field Accuracy but 0.74% Objective Type Accuracy, highlighting the Stage 1 classification bottleneck. Mistral-7B-Instruct-v0.2 achieved 16.54% Objective Type Accuracy and 74.52% Field Accuracy. In comparison, Phi-3.5-mini achieved scores of 30.12% and 90.06%, respectively. At the lower bound, Falcon3-1B achieves 0.99% Objective Type Accuracy and 29.82% Field Accuracy. Since Qwen-2.5-0.5B yields negligible accuracy, we omitted it from the figures for clarity. Overall, accurate Stage 1 objective classification remains the primary bottleneck: strong schema-filling performance is inaccessible when the target schema is misidentified.

4.4 Field-Level Error Analysis

We conduct an error analysis categorizing failures into three classes: structural/omission errors (missing fields, null values), semantic/content errors (incorrect values, type mismatches), and complex/interaction errors (field swapping, cross-dependencies). Figure 5 shows how our two-stage pipeline creates cascading failures: when models misidentify the objective type in Stage 1, they populate fields for the wrong schema in Stage 2, resulting in low field coverage (recall). To isolate field-filling performance from objective classification accuracy, we introduce a coverage-scaled error metric:

$$\text{Scaled Error Count} = \frac{\text{Raw Error Count}}{\text{Recall}} = \frac{E_{\text{raw}}}{R}$$

where R represents the field-coverage recall (the fraction of expected fields the model attempts under the correct schema). This normalization estimates field-filling error rates independent of objective classification performance. Phi-3.5-mini exhibits low field-coverage recall ($R = 0.37$), indicating it attempts only 37% of the expected fields under the correct schema. (For reference, its Objective Type Accuracy is 30.12%.) At this coverage level, the scaled analysis projects approximately 770 total errors if all fields were attempted across

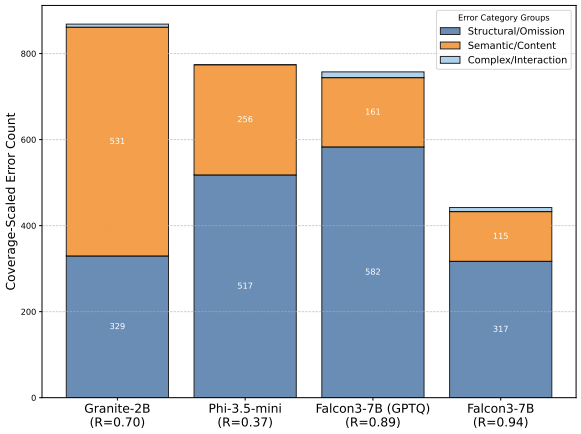

Figure 5: Coverage-scaled error analysis across four model architectures, showing the distribution of structural/omission, semantic/content, and complex/interaction errors normalized by field coverage (recall). Models with lower recall values (shown in parentheses) fill out fewer fields, and error counts are scaled by $1/R$ to estimate total errors if all fields were filled.

the 405 objectives. This projection reveals similar field-filling error rates between Phi-3.5-mini and Granite-2B (860 scaled errors), despite Granite-2B's higher field-coverage recall ($R = 0.67$). (Its Objective Type Accuracy is 70.12%.) The error distribution reveals architectural limitations in field population. Smaller models predominantly fail through structural omissions (582 scaled errors for Falcon3-7B-GPTQ), indicating schema comprehension deficits. Semantic errors remain relatively constant across model scales, suggesting that value-level understanding proves more robust than structural comprehension. Notably, quantized models show disproportionate increases in structural errors compared to their full-precision counterparts, indicating that compression techniques particularly impact schema understanding while preserving semantic capabilities. Figure 6 presents a detailed analysis across grouped objective fields, revealing distinct error profiles among models. Falcon3-1B demonstrates widespread structural failures with OMIT errors dominating all field categories. Mid-tier models like Granite-2B and Phi-3.5-mini exhibit more nuanced error patterns. While both struggle with structural errors in complex groups like "ID Lists" and "Timing Config," they increasingly show semantic errors for the fields they attempt to fill. Granite-2B encounters numerical errors in "Position & ROI" and arbitrary semantic errors in "Search Config," while Phi-3.5-mini exhibits semantic errors in "Request & Command."

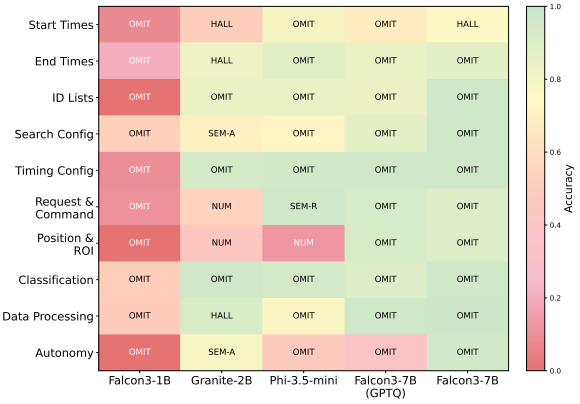

Figure 6: Average accuracy across grouped STM objective fields for LLMs evaluated on 405 test cases. Cell color indicates average accuracy (green=high, red=low) for the field group. Annotations specify the dominant error archetype (e.g., OMIT for omission, SEM-A for arbitrary semantic errors, HALL for hallucination, NUM for numerical errors) observed among the fields within each group that showed mistakes.

A notable pattern emerges in temporal field handling. Both Granite-2B and Falcon3-7B generate hallucinations in "Start Times," producing plausible but contextually incorrect values rather than omissions. This suggests systematic challenges in distinguishing hierarchically related temporal fields (objective-level versus intent-level times), representing a failure mode distinct from simple extraction errors. The more capable Falcon3-7B variants achieve higher accuracies across most field groups, yet quantization effects are evident. The GPTQ version shows increased OMIT errors compared to its unquantized counterpart, indicating that quantization degrades both structural comprehension and field population rates. These findings highlight the critical importance of accurate objective classification in our two-stage pipeline. Models like Falcon3-7B succeed by correctly identifying objective types 94% of the time, enabling their strong field-filling capabilities. In contrast, models with poor objective classification cascade these failures through the entire pipeline, making their field-filling abilities largely inaccessible. For STM deployment where complete and correct JSON objects are mandatory, the initial classification stage represents the primary bottleneck, as even models with strong field-filling capabilities cannot recover from objective misidentification.

5 Limitations

Our evaluation examines single generations per objective without analyzing consistency across multiple runs, which limits insights into output stability for deployment. The two-stage pipeline architecture creates cascading failure points where classification errors completely prevent schema generation, regardless of downstream capabilities. Furthermore, we evaluate only English prompts on our test cases with unambiguous specifications, whereas real operational scenarios often involve ambiguous or underspecified objectives requiring clarification. Our findings specifically address STM domain structured generation, and generalization to other safety-critical domains with different schema complexities remains unexplored. Additionally, we focus exclusively on accuracy metrics without examining latency variance or memory usage patterns under varying workloads, both of which are critical for operational deployment decisions.

6 Conclusion and Future Work

We find that 7B-parameter models can effectively generate structured STM commands on-premises, achieving over 90% accuracy through a two-stage pipeline. However, limitations emerge: cascading failures where classification errors prevent access to strong schema-filling capabilities, and quantization's disproportionate degradation of structural comprehension. Future research will explore methods to decouple these failure modes and preserve structural understanding under resource constraints. Collectively, these results provide the first systematic evaluation of on-premises LLMs for STM command generation, suggesting a path toward deployment in air-gapped environments.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, et al. 2024. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. *ArXiv*, abs/2404.14219.

AI@Meta. 2024. Llama 3 Model Card.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. 2023. Prompting Is Programming: A Query Language for Large Language Models. *Proc. ACM Program. Lang.*, 7(PLDI).

Shiqi Chen, Miao Xiong, Junteng Liu, Zhengxuan Wu, Teng Xiao, Siyang Gao, and Junxian He. 2024. In-Context Sharpness as Alerts: An Inner Representation Perspective for Hallucination Mitigation. *ArXiv*, abs/2403.01548.

Samuel Colvin, Eric Jolibois, Hasan Ramezani, Adrian Garcia Badaracco, Terrence Dorsey, David Montague, Serge Matveenko, Marcelo Trylesinski, Sydney Runkle, David Hewitt, and Alex Hall. 2024. Pydantic.

Erin Smith Crabb and Matthew T. Jones. 2024. Accelerating Model-Based Systems Engineering by Harnessing Generative AI. In *2024 19th Annual System of Systems Engineering Conference (SoSE)*, pages 110–115.

Nikolas Dahn, Stefan Fuchs, and Horst-Michael Gross. 2018. Situation Awareness for Autonomous Agents. In *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 666–671.

Enrique De Alba, Marco de Lannoy Kobayashi, and Alexander Cabello. 2024. Integrating LLMs With SatSim for Enhanced Satellite Tracking and Identification. In *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*.

Justin Fletcher, D Archambeault, J Schmidt, R Peterson, P Sydney, and S Hunt. 2021. The dynamic optical telescope system: Collaborative autonomous sensing for space domain awareness. *JDR&E*, 4:10–18.

Justin R. Fletcher and Jonathan E. Kadan. 2024. Autonomous interactive agents for global telescope network orchestration. In *Software and Cyberinfrastructure for Astronomy VIII*, volume 13101, page 131012U. International Society for Optics and Photonics, SPIE.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. OPTQ: Accurate Quantization for Generative Pre-trained Transformers. In *The Eleventh International Conference on Learning Representations*.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. 2023. Grammar-Constrained Decoding for Structured NLP Tasks without Finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10932–10952, Singapore. Association for Computational Linguistics.

IBM Granite Team. 2024. Granite 3.0 Language Models.

GuidanceAI. 2023. Guidance: A language model programming framework.

Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L'elio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *ArXiv*, abs/2310.06825.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *ArXiv*, abs/2001.08361.

Will Kurt. 2024. Coalescence: Making LLM Inference 5x Faster. Blog post on .TXT.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Jason Liu. 2024. Welcome to instructor - instructor. https://python.useinstructor.com/.

Mistral AI Team. 2024. Mistral-Small-Instruct-2409: A 22B Parameter Large Language Model. Hugging Face Model Repository. Accessed: 2025-02-25.

Brett M. Morris, Erik Tollerud, Brigitta Sipőcz, Christoph Deil, Stephanie T. Douglas, Jazmin Berlanga Medina, Karl Vyhmeister, Toby R. Smith, Stuart Littlefair, Adrian M. Price-Whelan, Wilfred T. Gee, and Eric Jeschke. 2018. astroplan: An Open Source Observation Planning Package in Python. *The Astronomical Journal*, 155(3):128.

Omnifact. 2025. The Case for Self-Hosting Large Language Models in Enterprise AI.

OpenAI. 2024. Introducing Structured Outputs in the API. https://openai.com/index/introducing-structured-outputs-in-the-api/.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. 2025. LLMs Know More Than They Show: On the Intrinsic Representation of LLM Hallucinations. In *The Thirteenth International Conference on Learning Representations*.

Falcon-LLM Team. 2024. The Falcon 3 Family of Open Models.

The Aerospace Corporation, Space Safety Institute. 2024. 2024 Space Safety Compendium: Collaborating for Sustainable Space Futures. Technical report, The Aerospace Corporation.

United Nations Office for Outer Space Affairs. 2025. Annual number of objects launched into space. Data processed by Our World in Data.

Brandon T. Willard and Rémi Louf. 2023. Efficient Guided Generation for Large Language Models. *ArXiv*, abs/2307.09702.

Yi Xiong, Hao Wu, Changxu Shao, Ziqing Wang, Rui Zhang, Yuhong Guo, Junping Zhao, Ke Zhang, and Zhenxuan Pan. 2024. LayerKV: Optimizing Large Language Model Serving with Layer-wise KV Cache Management.

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. 2024. Qwen2.5 Technical Report. *ArXiv*, abs/2412.15115.

Tianyi Zhang, Jonah Wonkyu Yi, Zhaozhuo Xu, and Anshumali Shrivastava. 2024. KV Cache is 1 Bit Per Channel: Efficient Large Language Model Inference with Coupled Quantization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.