

Markdown vs JSON for LLMs

Author: Kevin Anderson

Date: 03/14/2026

Topic: A Whitepaper on Readability, Reliability, Speed, and Structural Control in Human-Model Workflows

Abstract

As large language models move from experimentation into production systems, the choice of prompt and output format has become a practical engineering decision rather than a stylistic preference. Among the most common contenders, Markdown and JSON represent two different design philosophies. Markdown emphasizes readability, flexible hierarchy, and human collaboration. JSON emphasizes explicit structure, schema validation, and machine reliability. The question is no longer which format is "better" in the abstract, but which one is better for a given class of LLM workloads.

- Recent research shows that formatting itself can materially affect model performance: one 2024 study found that prompt template choice across plain text, Markdown, JSON, and YAML could significantly change results, including a reported case where **JSON improved accuracy by 42 percent over Markdown** on a code translation task.

This paper argues that Markdown is generally superior for human-LLM collaboration, long-form reasoning, documentation, retrieval synthesis, and iterative prompt development, while JSON is generally superior when exact structural fidelity, parser-safe outputs, or schema-constrained generation are non-negotiable. That position is supported by both vendor guidance and benchmark literature. OpenAI's Structured Outputs documentation explicitly recommends JSON Schema when developers need responses that reliably conform to a predefined structure, while recent structured-generation benchmarks such as JSONSchemaBench and LLMStructBench focus on the difficulty of generating valid, schema-compliant JSON at scale.

The central thesis is therefore conditional and operational: Markdown is the better default format for most mixed human-machine knowledge work because it offers strong readability with lightweight structure, but JSON is the better format for machine-enforced consistency, extraction pipelines, and constrained generation. In practice, the most effective systems often use both: Markdown as the authoring and reasoning layer, JSON as the final execution or interchange layer.

1. Introduction

The rise of LLMs has revived an old systems question in a new form: how should humans encode intent for machines? In traditional software, that question was answered by formal interfaces such as programming languages, data schemas, and APIs. In LLM systems, the answer is less rigid but no less consequential. Natural language is the primary interface, yet production applications increasingly require more than free-form prose. They require hierarchy, boundaries, constraints, and outputs that can be reviewed by people or consumed by software. That need is what brings formats like Markdown and JSON into direct comparison.

Markdown

Organizes information for **reading and writing**. Supports headings, lists, tables, quotes, and code fences — highly effective for prompts, reports, instructions, and documentation.

Optimized for the **human in the loop**.

JSON

Organizes information for **data exchange**. Supports key-value structures, arrays, typed primitives, and schema validation — ideal for APIs, extraction, tool calls, and automation pipelines.

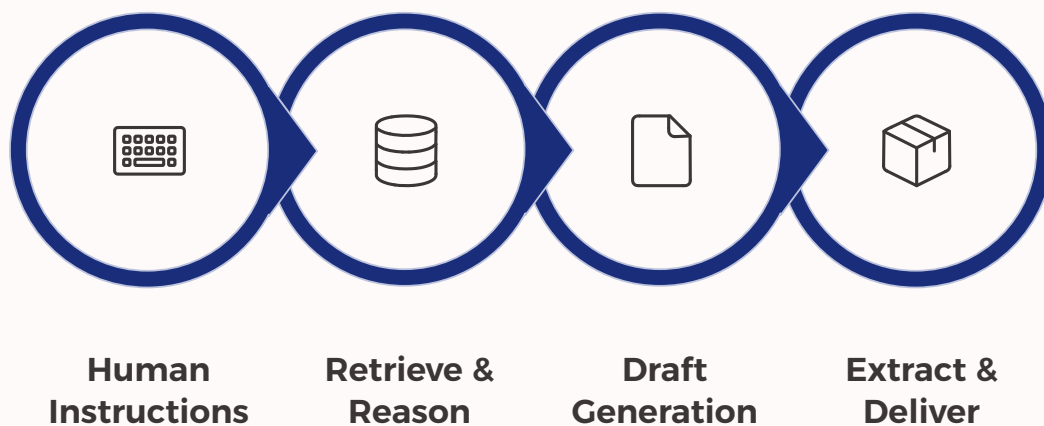
Optimized for the **machine in the loop**.

This paper examines that tradeoff in academic and practical terms. It asks four questions. First, how does format affect LLM behavior? Second, where does Markdown outperform JSON? Third, where does JSON clearly outperform Markdown? Fourth, what should organizations prioritize if their primary decision criteria are speed and quality?

The answer that emerges is not ideological. It is architectural. Markdown and JSON are not true substitutes in every context; they are interface layers with different strengths. The most mature LLM systems increasingly combine them rather than choosing one exclusively.

2. Why This Comparison Matters

The comparison between Markdown and JSON matters because modern LLM applications operate across both human-facing and machine-facing layers. A single workflow may begin with human-authored instructions, draw from retrieved documents, reason over long context, generate a draft, extract structured facts, and finally send the result into software systems. The format chosen at each stage can influence not only correctness, but latency, editability, review overhead, and downstream reliability.



The academic relevance is now clear as well. The 2024 prompt-formatting paper tested multiple human-readable templates and showed that formatting differences alone can lead to substantial variation in output quality across tasks. That finding is important because it means format is not merely presentational. It is part of the experimental condition. A system that benchmarks one prompt format and deploys another may unknowingly change its performance profile.

The industrial relevance is arguably even greater. OpenAI's Structured Outputs feature exists because many real applications need guaranteed schema adherence, not merely "mostly valid" output. JSON has become central in this layer because applications need to trust that required keys exist, enum values are valid, and nested objects obey predetermined structure. At the same time, most people do not naturally think, write, or review complex instructions in JSON. That tension explains why the Markdown-versus-JSON question has become so important.

3. Theoretical Framing: Two Different Interface Philosophies

Markdown and JSON can be understood as embodying two interface philosophies.

Markdown – Semantic Presentation Format

Markdown encodes hierarchy and emphasis in a way that remains easy for humans to read and modify. Its syntax is sparse. A heading can be created with a handful of characters. A list or code fence is similarly lightweight. This makes Markdown an efficient way to express structured thought without imposing heavy syntactic overhead.

In LLM contexts, that lightweight structure often helps separate task instructions, context, examples, and constraints while preserving a natural writing flow.

JSON – Formal Data Interchange Format

JSON is not primarily designed for narrative clarity, but for predictable serialization. Its strength lies in explicit field names, nesting, lists, and compatibility with schemas, parsers, validators, and downstream software.

OpenAI's Structured Outputs guidance reflects this exact philosophy: when the application must enforce a schema, JSON is the preferred target format because it can be validated and constrained.

- ❏ This difference matters because LLM use cases do not all sit on the same side of the interface. Some tasks are conversational, exploratory, and document-centric. Others are transactional, API-bound, and validation-centric. **Markdown tends to excel in the former. JSON tends to excel in the latter.** Neither dominance invalidates the other; rather, it highlights that format choice should follow workload design.

4. What the Research Says About Format Sensitivity

The strongest direct evidence that format matters comes from the 2024 study *"Does Prompt Formatting Have Any Impact on LLM Performance?"* The authors compared plain text, Markdown, JSON, and YAML across reasoning, code generation, and translation tasks using GPT-family models. Their results showed that prompt format can significantly affect performance and that the effect varies by task and model size.

42%

JSON over Markdown

Accuracy improvement on a code translation task when using JSON prompt format vs. Markdown (2024 study)

~10K

JSON Schemas

Real-world JSON schemas used in JSONSchemaBench to evaluate constrained decoding systems on efficiency, coverage, and quality

2026

LLMStructBench

Extended structured-generation evaluation into 2026, covering valid JSON generation across diverse parsing scenarios

This result is important for two reasons. First, it disproves the common assumption that semantic equivalence implies formatting equivalence. Two prompts can ask for the same thing yet still produce meaningfully different outputs because their structure changes how the model interprets context boundaries and expectations. Second, it suggests that "best format" is not a universal property. It depends on the task. A format that performs well for explanatory generation may perform worse for constrained translation or structured extraction.

The structured-generation literature strengthens the JSON side of this comparison. JSONSchemaBench was introduced as a benchmark of roughly 10,000 real-world JSON schemas to evaluate constrained decoding systems on efficiency, coverage, and quality. Its existence reflects a broader industry shift: generating valid JSON is not trivial, and

9. Recommended Evaluation Framework for Teams

Organizations deciding between Markdown and JSON should benchmark them across the same task families they actually deploy. A credible evaluation should include at least five categories:

01	02	03
Long-Form Generation	Retrieval-Based Q&A	Structured Extraction
Evaluate narrative quality, coherence, and editability of extended outputs.	Assess how well each format supports synthesis from retrieved context.	Measure field recovery accuracy, schema compliance, and parser success rate.
04	05	
Workflow Orchestration	Tool- or API-Bound Execution	
Test multi-step agent coordination and handoff reliability between components.	Evaluate downstream integration success, validation pass rates, and failure handling.	

Speed Metrics to Measure

- Time to first token
- Total latency
- Token count
- Human editing time
- Parser success rate

Quality Metrics to Measure

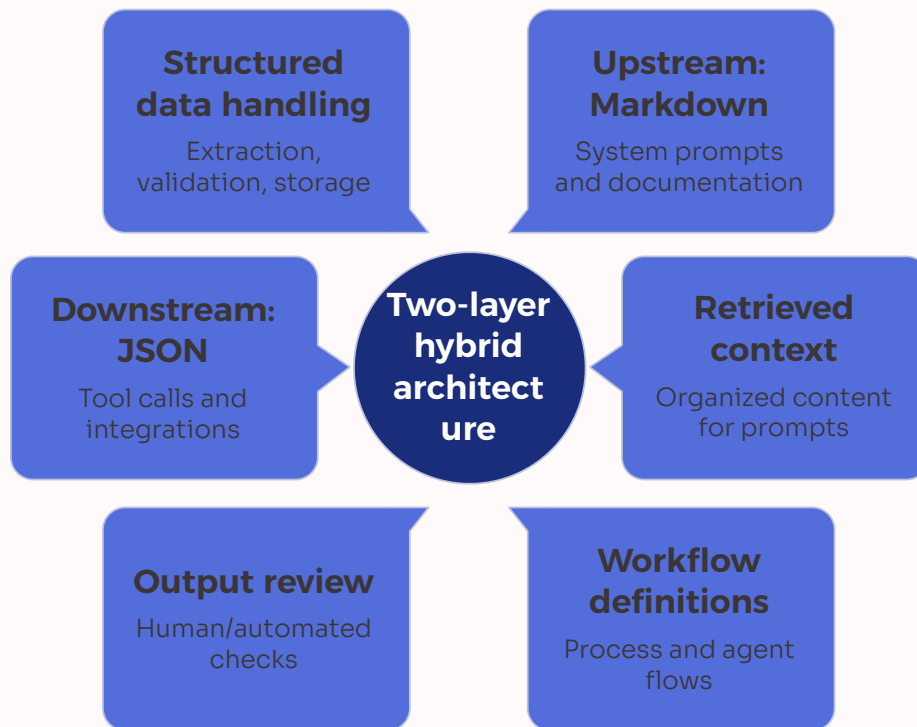
- Semantic accuracy
- Readability
- Schema validity
- Completeness
- Downstream task success

❏ A format that slightly increases token count but dramatically reduces manual correction may still be the superior system choice. Conversely, a beautifully readable format that fails downstream validation too often may be unacceptable in production.

The best benchmark hypothesis is not "Markdown beats JSON" or "JSON beats Markdown." It is that **Markdown is the better default for mixed human-machine reasoning and content workflows**, while **JSON is the better default for schema-bound execution and structured automation**. That is the hypothesis most consistent with the literature and with official product guidance.

10. Best-Practice Architecture: Markdown Upstream, JSON Downstream

In mature LLM systems, the strongest pattern is often hybrid rather than exclusive.



Markdown – Upstream Layer

- Writing system prompts
- Organizing retrieved context
- Defining workflows
- Documenting agent behavior
- Reviewing generated outputs

JSON – Downstream Layer

- Tool calls
- Data extraction
- Structured storage
- Validation
- Application integration

This architecture mirrors how teams actually work. Humans think and collaborate in readable documents. Software executes against typed structures. Asking JSON to solve the entire collaboration problem is inefficient. Asking Markdown to solve the entire validation problem is unreliable. The combination is more powerful than either one alone.

Use Markdown when people need to read, refine, and reason. Use JSON when systems need to trust, parse, and act. When a workflow spans both, **treat Markdown as the workspace and JSON as the contract.**

11. Limitations

This comparison has several limitations that should be acknowledged before drawing firm conclusions.

1 Limited Direct Head-to-Head Literature

Direct head-to-head literature specifically isolating Markdown versus JSON across many modern frontier models remains limited; much of the evidence comes from broader prompt-format studies or structured-generation benchmarks rather than a single canonical comparison.

2 Model Behavior Changes Over Time

Improvements in structured decoding, output controls, and prompt handling can alter the relative strengths of each format. That means conclusions should be treated as current best evidence, not permanent law. OpenAI's own product evolution from JSON mode to Structured Outputs illustrates how quickly this area advances.

3 Context-Dependent Quality

Human-facing publishing tasks and machine-facing automation tasks do not reward the same properties. Any blanket statement that one format is better without specifying the task is therefore analytically weak.

12. Conclusion

The comparison between Markdown and JSON for LLMs is best understood as a comparison between two kinds of structure. Markdown provides lightweight semantic structure for humans and models working together. JSON provides explicit formal structure for models and software working together. Both are valuable. Both improve on unstructured text in different ways.

When Markdown is the Better Default

When the primary goals are **readability, reasoning, collaboration, and long-form content quality**, Markdown is usually the better default.

When JSON is the Better Default

When the primary goals are **schema fidelity, parser reliability, automation, and machine consumption**, JSON is usually the better default.

The evidence does not support a universal winner. It supports a division of labor.

The most robust production strategy is therefore not Markdown *or* JSON. It is Markdown *and* JSON, used at different layers of the same system. **Markdown should structure thought. JSON should structure execution.** That is the cleanest answer to the speed-and-quality question, and it is the one most consistent with both benchmark evidence and real-world LLM architecture.

References

- 1 He, J., Rungta, M., Koleczek, D., Sekhon, A., Wang, F. X., and Hasan, S. **Does Prompt Formatting Have Any Impact on LLM Performance?** *arXiv:2411.10541*.
 - 2 Geng, S., Cooper, H., Moskal, M., Jenkins, S., Berman, J., Ranchin, N., West, R., Horvitz, E., and Nori, H. **JSONSchemaBench: A Rigorous Benchmark of Structured Outputs for Language Models.** *arXiv:2501.10868*.
 - 3 Tenckhoff, S., et al. **LLMStructBench: Benchmarking Large Language Model Structured Data Extraction.** *arXiv:2602.14743*.
 - 4 OpenAI. **Structured model outputs.**
 - 5 OpenAI. **Introducing Structured Outputs in the API.**
-