The Hidden Cost of Structure: How Constrained Decoding Affects Language Model Performance

Maximilian Schall
Hasso Plattner Institute, Germany
Maximilian.Schall@hpi.de

Gerard de Melo
Hasso Plattner Institute, Germany
Gerard.deMelo@hpi.de

Abstract

Large Language Models excel at generating fluent text, but real-world applications increasingly demand structured outputs like JSON that can be programmatically processed. While prior work examines either task performance or format compliance in isolation, we investigate their interaction through comprehensive experiments across 11 models and multiple benchmarks. We uncover a fundamental divergence between base and instruction-tuned models under structural constraints. Base models often benefit from constrained decoding, producing more precise outputs, while instruction-tuned models frequently suffer performance degradation on generation tasks despite maintaining stability on classification tasks. Our log probability analysis reveals the underlying mechanism: constrained decoding forces models away from their preferred natural language patterns into lower-confidence structured alternatives. We demonstrate that successful constrained generation requires both adapted prompts and sufficient few-shot examples, with constrained models showing steeper performance gains from additional demonstrations compared to unconstrained generation. Notably, we find that base model performance under constraints can serve as an early indicator of post-training structured output capabilities, offering a practical evaluation tool for model development. These findings suggest that current instruction-tuning practices may inadvertently reduce models' structured output capabilities and highlight the need for training-time integration of structural constraints in future model development.

1 Introduction

Large Language Models (LLMs) have achieved remarkable success across diverse linguistic domains, excelling in both *natural language processing* (Zhao et al., 2023; Touvron et al., 2023) and *formal language generation* (Rozière et al., 2023; Lozhkov et al., 2024). Beyond traditional conversational applications, LLMs increasingly serve as components in complex computational pipelines that demand precisely structured outputs, such as JSON or YAML, where format accuracy is critical for downstream processing. Another important application is *function calling*, enabling LLMs to interface with external tools (Schick et al., 2023; Hao et al., 2023) and APIs (Patil et al., 2024; Qin et al., 2024) through structured command generation.

Two primary approaches exist for obtaining structured outputs from LLMs: prompt-based guidance and *constrained decoding* (Deutsch et al., 2019; Geng et al., 2023). Prompt-based methods rely on natural language instructions to guide format compliance but remain vulnerable to generation errors, necessitating robust error handling mechanisms. Constrained decoding, guarantees structural validity by restricting the model's token-level predictions during generation. Despite the growing adoption of constrained decoding, its impact on models' core task performance remains poorly understood.

Drawing inspiration from Wei et al. (2023)'s findings on adversarial prompting and cognitive load in multi-task scenarios, we hypothesize that forcing models to satisfy both task objectives and structural constraints simultaneously may impair their problem-solving capabilities, particularly when dealing with complex output schemas. While existing literature has examined either task-specific model capabilities (Min et al., 2023; Hendrycks et al., 2021; Lin et al., 2022) or structural compliance (Zhou et al., 2023; Xia et al., 2024) in isolation, no comprehensive study has investigated their interaction. Our work addresses this gap by systematically analyzing how structural output requirements affect task performance across diverse model architectures, training paradigms, and prob-

Proceedings of Recent Advances in Natural Language Processing,pages 1074–1084
Varna, Sep 8–10, 2025
https://doi.org/10.26615/978-954-452-098-4-124

lem complexities. As illustrated in Figure 1, unconstrained generation often produces correct but verbose responses, while constrained generation may yield structurally valid but factually incorrect output. Our contribution are as follows:

1. **Systematic performance analysis:** We conduct a comprehensive evaluation revealing how constrained decoding affects task accuracy across multiple dimensions; including model architectures, training approaches (base vs. instruction-tuned), task types, and output complexity levels. We identify when and why performance degradation occurs.

2. **Mechanistic insights through comparative analysis:** We uncover the underlying mechanisms of constrained decoding's impact by analyzing log probability distributions, showing how structural constraints force models away from their preferred token choices, leading to measurable confidence reduction and performance drops.

3. **Evidence-based implementation guidelines:** Based on our empirical findings, we provide concrete recommendations for effectively deploying constrained decoding, including the critical importance of adapted prompting, the value of few-shot examples, and strategies for pre-assessing model suitability for pre-training.

2 Related Work and Background

Constrained decoding guides language models to produce outputs that satisfy specific requirements by modifying token probabilities during generation. Early work by Hokamp and Liu (2017) and Anderson et al. (2017) pioneered lexical constraints , enabling models to include or exclude predetermined words and phrases. Subsequent improvements by Hasler et al. (2018) introduced explicit alignment mechanisms, while Post and Vilar (2018) significantly reduced computational overhead through algorithmic optimizations. However, these approaches addressed only lexical constraints, leaving structural formatting unexplored.

Grammar-constrained decoding extends these concepts to enforce syntactic validity according to formal grammars. Deutsch et al. (2019) introduced a general framework using finite state automata

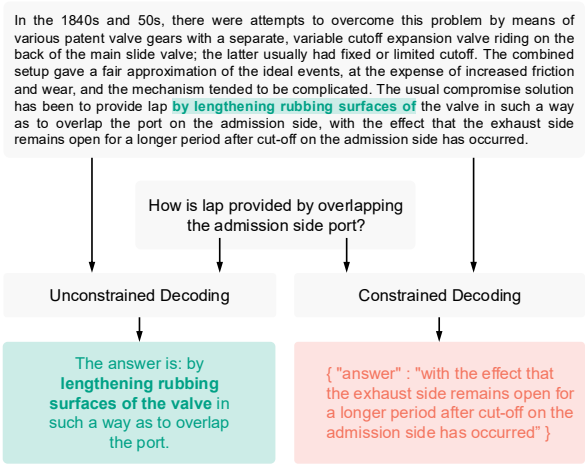

Figure 1: Example demonstrating performance degradation under constrained decoding. While `Llama3.1`'s unconstrained generation produces a correct answer embedded in natural language, the same model generates an incorrect answer when forced to comply with a structured JSON format, illustrating the trade-off between structural compliance and task accuracy.

(FSAs) for *regular grammars* and pushdown automata (PDAs) for *context-free grammars*, enabling precise structural control over generated outputs. Building on this foundation, Geng et al. (2023) demonstrated that numerous NLP tasks can be reformulated as formal language generation problems, though their evaluation remained limited to relatively simple task structures.

The evaluation landscape for LLMs has evolved along two parallel tracks. Content-focused assessments dominate the field, encompassing both closed-ended benchmarks (Hendrycks et al., 2021; Lin et al., 2022) and open-ended generation tasks (Rajpurkar et al., 2016; Min et al., 2023). More recently, researchers have begun evaluating structural compliance capabilities, examining models' ability to generate outputs conforming to specific formats such as JSON schemas or prescribed document structures (Zhou et al., 2023; Xia et al., 2024). Despite these advances, the interaction between content accuracy and structural constraints remains understudied, a gap our work addresses through systematic evaluation across diverse tasks and model architectures.

2.1 Constrained Decoding for Structured Generation

We formalize constrained decoding as the process of generating a sequence

$$x = [x_1, x_2, \ldots, x_{|x|}],$$

where each token x_i is constrained to follow a predefined structure S. Let $p(x_i \mid x_{<i}, S)$ represent the probability of generating the next token x_i given the preceding tokens $x_{<i}$ and the structure S. The goal of constrained decoding is to ensure that $x \in S$, i.e., that the generated sequence adheres conforms to the schema or pattern defined by S. The constrained set of tokens at each step, C_i, is derived by filtering the model's next-token distribution to exclude tokens that would violate S. Formally, we select x_i from the restricted set $C_i \subseteq V$ (where V is the vocabulary), ensuring:

$$p(x_i \mid x_{<i}, S) = 0 \quad \text{for} \quad x_i \notin C_i$$

3 Experimental Setup

3.1 Language Models

We evaluate eleven open-source language models spanning five architectures, each with 4-8 billion parameters. Our model selection includes Llama (Touvron et al., 2023), Mistral (Jiang et al., 2023), Phi (Abdin et al., 2024), DeepSeek (DeepSeek-AI et al., 2024), and OLMo (Groeneveld et al., 2024). For each architecture, we examine both base and instruction-tuned variants when available, enabling systematic comparison of how training paradigms affect constrained generation performance.

We include OLMo specifically to eliminate potential data contamination concerns, as its transparent training dataset (Soldaini et al., 2024) guarantees no exposure to our evaluation benchmarks. Additionally, we incorporate DeepSeek-Coder (Guo et al., 2024) to investigate whether code-specific pretraining confers advantages for structured output generation.

3.2 Tasks and Metrics

We evaluate model performance across multiple generation and classification tasks.

SQuAD (Rajpurkar et al., 2016) tests reading comprehension through extractive question answering. We limit generation to 64 tokens to prevent models from simply reproducing input passages. For unanswerable questions, we allow models to generate either predefined templates (e.g., "*No Answer*") for unconstrained generation or `null` values for constrained generation.

We calculate $\mathbf{EM_{IN}}$ (Exact Match In), which measures whether the generated text contains the correct answer span.

SQuAD Advanced extends the basic task to evaluate complex structurs. Models must answer multiple questions simultaneously, returning results in a single JSON object. To maintain consistent few-shot learning conditions, we provided similar examples of context and questions. Given the complexity of this multi-question format, we focused our evaluation on structured data.

IFEval (Zhou et al., 2023) evaluates instruction-following capabilities through constraints like avoiding specific punctuation or formatting requirements. Following the original methodology, we report both strict and loose adherence metrics at prompt and instruction levels.

FActScore (Min et al., 2023) measures factual accuracy in biographical generation. We adapt the original implementation to use `Llama3.1-8B-Instruct` for fact extraction and verification. Our **FactScore Advanced** variant additionally prompts readily verifiable biographical details and the biographical text: birth year, birthplace, and occupation. We use EM_{IN} to verify the presence of these details in natural text responses. We incorporate these three biographical fields directly into our generation schema for constrained generation and evaluate those fields. We further exclude entities whose names are shared by multiple notable individuals.

Classification Tasks we evaluate include **MMLU** (Hendrycks et al., 2021) and **TruthfulQA** (Lin et al., 2022). We evaluate accuracy by computing log-likelihoods across answer candidates, simulating constrained decoding through structured prefixing.

3.3 Hyperparameters

We use vLLM[1] for inference with greedy decoding (temperature=0.0) to ensure reproducibility (Lee, 2023). Maximum generation length is set to 2,048 tokens unless task-specific limits apply.

For in-context learning, we provide $n = 9$ examples for SQuAD variants, $n = 5$ for MMLU, and $n = 0$ for TruthfulQA, FActScore, and IFEval, following original implementations. For tasks with training sets, we treat prompt selection as a hyperparameter, optimizing from a diverse prompt pool for each model-task-decoding combination.

We enforce the structured format with `outlines` (Willard and Louf, 2023):

[1] https://github.com/vllm-project/vllm

- **Simple schemas**: Basic tasks use `{"answer": [str|null]}`
- **Complex schemas**: Advanced variants employ structures with multiple fields
- **Classification constraints**: Simulated through `{"answer": "` prefixing

All prompts follow best practices, providing explicit format examples and clear structural specifications. Experiments utilized a single H100 GPU, consuming approximately 1,200 GPU hours total.

4 Results

4.1 SQuAD

In Table 1 we observe that unstructured outputs generally yield higher scores across most models, particularly for instruction-tuned variants. For instance, Phi-3-Instruct achieves an EM_{IN} score of 72.8% with unconstrained decoding, while the performance drops to 63.2% with constrained generation. Interestingly, we observe different behavior in base models, where structured outputs often lead to better results. Llama3-8B Base, for example, shows better performance with structured outputs (59.6%) compared to unconstrained generation (42.5%). This suggests that balancing the goal of adhering to format constraints with that of providing accurate answers remains challenging for instruction-tuned models. In contrast, the constraints appear to aid base models that lack instruction-tuning, serving as a form of output guidance. On **SQuAD Advanced**, the performance drops significantly across all models. Mistral-7B-Instruct, which performs best in both unconstrained (81.1%) and constrained (73.7%) scenarios, drops to 47.3% with advanced constraints. The base models show an similar degradation.

4.2 IFEval

Table 2 presents the strict and loose compliance scores under both constrained and unconstrained decoding for different models.

The results demonstrate that unconstrained decoding generally leads to higher compliance scores in both strict and loose evaluations. For instance, Llama3-8B-Instruct achieves a strict instruction compliance score of 84.1% with unconstrained decoding, compared to 52.6% with constrained decoding. This shows that models are better at following instructions when not constrained by a specific output format. As with SQuAD, most Base-models improve performance with constrained decoding.

Models		Unconstrained	Constrained	Advanced
Llama3-8B	Base	42.5	59.6	31.0
	Instruct	75.1	72.7	**59.5**
Deepseek	Base	75.9	53.3	23.1
	Chat	68.2	56.9	23.1
Deepseek-Coder	Base	75.4	58.1	25.2
	Instruct	67.6	57.2	36.9
Mistral-7B	Base	64.7	71.5	29.6
	Instruct	**81.1**	**73.7**	47.3
OLMo-7B	Base	72.0	38.0	9.3
	Instruct	59.4	50.6	11.8
Phi-3	Instruct	72.8	63.2	41.5

Table 1: Results for SQuAD on the EM_{IN} metric. Numbers are percentages rounded to one decimal place. The best metric per type is marked in **bold**.

4.3 FActScore

Table 3 the results on the evaluation for standard and advanced FActScore. For standard FActScore, the differences in factual accuracy between constrained and unconstrained outputs are modest. Base models such as Llama3-8B-Base achieve similar scores under both conditions (47.3% for constrained versus 44.6% for unconstrained).

In the advanced FActScore evaluation, we observe larger performance gaps. Notably, OLMo-Base exhibits a substantial difference between constrained (26.3%) and unstructured (54.4%) outputs. In contrast, instruction-tuned models such as Llama3-8B-Instruct show greater stability, with scores showing slight variation between constrained (44.2%) and unconstrained (37.8%) generation.

The generation of specific biographical details, shows a big contrast between base and instruction-tuned models. For example, Llama3-8B-Base achievs an accuracy of 35.3% in a constrained format while completely failing (0%) in an unconstrained format. This further shows that structural constraints can significantly aid base models in following instructions. Instruction-tuned models, however, demonstrate more consistent performance across formats. For instance, Mistral-7B-Instruct maintains an accuracy of around 34% in both conditions.

An interesting pattern arises regarding the number of facts generated: base models consistently produce far more facts in unconstrained settings.

Models		Strict				Loose			
		Instruction		Prompt		Instruction		Prompt	
		Constrained	Unconstrained	Constrained	Unconstrained	Constrained	Unconstrained	Constrained	Unconstrained
Llama3.1-8B	Base	28.7	3.2	16.6	1.8	28.4	3.7	16.5	2.6
	Instruct	**52.6**	**84.1**	**38.6**	**77.4**	**52.6**	**86.7**	**38.6**	**80.6**
Deepseek	Base	27.3	29.9	14.4	16.1	27.3	31.1	14.4	16.8
	Chat	31.2	47.4	19.8	36.8	31.2	50.5	19.8	40.5
Deepseek-Coder	Base	30.7	16.1	18.7	11.5	30.7	16.7	18.7	11.6
	Instruct	30.7	39.2	20.9	29.4	30.8	39.9	20.9	30.3
Mistral-7B	Base	26.0	27.5	12.8	17.9	26.0	28.9	12.8	18.5
	Instruct	44.2	60.0	31.4	49.2	44.4	63.5	31.6	52.9
OLMo-7B	Base	27.1	33.9	14.2	20.5	27.1	34.9	14.2	21.4
	Instruct	28.9	42.7	18.5	31.6	28.7	46.5	18.5	35.5
Phi-3	Instruct	40.9	60.4	28.5	49.0	41.0	64.5	28.7	53.8

Table 2: IFEval results for different models across constrained and unconstrained instruction and prompt settings, with metrics rounded to one decimal place. The results are split between strict and loose evaluation criteria. The best metric per type is marked in **bold**.

Llama3-8B-Base generates an average of 832 facts per response in the unconstrained advanced evaluation, compared to just 28 facts in the structured format. Instruction-tuned models show more restrained behavior in terms of fact generation. For example, Mistral-Instruct generates around 70 facts per response in the unconstrained advanced evaluation, compared to 16 in the constrained evaluation, a much smaller ratio than that seen in base models.

4.4 Classification Task

The results in Table 4 shows a smaller impact of constrained decoding on classification tasks, compared to open-ended generation tasks. For MMLU, most models exhibit similar performance under both structured and unstructured settings. For instance, Mistral-7B-Instruct achieves 60% accuracy under constrained decoding and 61% in unconstrained settings.

However, one exception is Phi-3, which shows an improvement from 55% to 67% when using unconstrained decoding on MMLU. For TruthfulQA, we observe mixed results in performance under constrained decoding. For example, Llama3-8B-Instruct achieves 55% accuracy with structured outputs but this decreases to 52% with unstructured outputs, while for OLMo-7B-Instruct, the accuracy increases from 24% with structured outputs to 27% with unstructured outputs.

Unlike in generation tasks, the performance of classification tasks remains relatively stable under output format constraints. We argue that this stability comes from the models merely being tasked with selecting among predefined answers, making it inherently a constrained decoding task.

4.5 The Effect of Base vs. Instruction-Tuned Models

Our experiments reveal different patterns between base and instruction-tuned models under constrained decoding. While instruction-tuned models generally outperform base models in both structured and unstructured settings, they often exhibit larger performance drops when constraints are applied. This performance reduction may stem from a conflict between their learned objective to follow diverse human instructions and the structure imposed by constrained output formats. The models appear to struggle with balancing adherence to specific output structures while interpreting and executing given instructions effectively.

Base models, on the other hand, frequently benefit from structured constraints. This improvement suggests that for models without instruction-tuning, constrained decoding provides valuable guidance that helps focus their outputs. This finding has important implications: evaluating base models with constrained decoding before instruction tuning could provide insights into their potential structured output capabilities post-tuning.

4.6 Does Code-Specific Pretraining Help?

We also examined the effects of code-specific pretraining on model performance, particularly in the context of constrained decoding. Our hypothesis was that pretraining on source code could enhance a model's ability to handle structured outputs due to the inherently structured nature of programming languages.

However, as shown in Table 1 and Table 2, the impact of code-specific pretraining is not straightforward. In the SQuAD task, the DeepSeek-Coder-7B-Base model demonstrates higher per-

Models		Standard		Advanced		Bio Accuracy	
		Constrained	Unconstrained	Constrained	Unconstrained	Constrained	Unconstrained
Llama3-8B	Base	47.3 (38.7)	**44.6** (434.7)	39.3 (28.1)	52.1 (832.1)	35.3	0
	Instruct	46.7 (45.3)	36.4 (102.9)	**44.2** (20.5)	37.8 (12.3)	**36.2**	**35.2**
Deepseek	Base	39.1 (84.5)	39.2 (431.2)	30.6 (18.9)	33.4 (501.4)	22.1	26.4
	Chat	38.5 (46.5)	34.3 (105.9)	**44.2** (22.7)	41.4 (68.1)	24.3	28.3
Deepseek-Coder	Base	29.9 (50.0)	28.7 (597.0)	26.4 (22.9)	35.2 (646.2)	8.1	8.2
	Instruct	30.2 (32.7)	19.5 (30.4)	31.2 (30.2)	14.0 (45.1)	12.3	0
Mistral-7B	Base	**48.5** (39.6)	38.3 (448.3)	37.8 (20.1)	40.3 (521.8)	28.2	33.8
	Instruct	37.2 (38.8)	33.0 (100.4)	42.4 (16.3)	39.7 (70.6)	34.7	34.3
OLMo-7B	Base	31.3 (36.7)	40.3 (548.1)	26.3 (28.6)	**54.4** (465.1)	10.8	6.4
	Instruct	26.3 (52.3)	28.8 (169.6)	23.8 (42.7)	34.3 (65.1)	15.2	17.8
Phi-3	Instruct	35.4 (54.9)	28.7 (163.3)	39.0 (46.1)	33.1 (110.6)	26.3	28.3

Table 3: FActScore results across basic and advanced evaluation settings, reporting factual accuracy scores along with the average number of facts per response (in parentheses). The Bio Accuracy columns show performance on biographical detail extraction. The best metric per type is marked in **bold**.

Models		MMLU		TruthQA	
		Constrained	Unconstrained	Constrained	Unconstrained
Llama3-8B	Base	24.4	24.3	23.8	18.5
	Instruct	41.0	54.4	56.1	51.8
Deepseek	Base	46.2	48.1	20.7	19.8
	Chat	46.7	48.3	27.2	31.9
Deepseek-Coder	Base	47.4	48.9	21.5	22.2
	Instruct	33.3	37.4	32.7	16.9
Mistral-7B	Base	52.9	48.7	23.0	31.2
	Instruct	**59.9**	61.1	37.5	49.2
OLMo-7B	Base	27.5	27.9	24.6	22.8
	Instruct	46.8	48.0	23.6	26.6
Phi-3	Instruct	55.4	**67.4**	**56.1**	**62.9**

Table 4: Accuracy for MMLU and TruthQA. The best accuracy per type and task is marked in **bold**.

formance with unconstrained decoding (EM_{IN} score of 75.4%) compared to constrained decoding (58.1%), unlike to other base models. The DeepSeek-Coder-7B-Instruct model also follows this pattern, performing better with unconstrained decoding.

For IFEval, the DeepSeek-Coder models do not show significant improvements over models without code-specific pretraining. The DeepSeek-Coder-7B-Instruct model achieves a strict instruction compliance score of 30.7 with structured outputs, which is similar to the non-code pretrained models.

These findings suggest that code-specific pretraining does not necessarily improve the model's performance in tasks requiring structured outputs in natural language.

5 Further Investigations

This section presents detailed analyses examining why constrained decoding affects model performance differently than unconstrained decoding. We investigate the causes of these performance differences and explore potential mitigation strategies, focusing our analysis on the SQuAD dataset.

5.1 Do Models Require Adapted Prompts?

We investigate whether models can successfully adapt to output constraints without explicit prompting. To test this, we apply constrained decoding to models using natural prompt, keeping all other parameters constant. This approach allows us to isolate the effect of schema enforcement from prompt adaptation.

Table 5 shows a accuracy degradation when constrained decoding is applied without adapted prompts. Most notably, Mistral-7B-Base accuracy drops from 71.5% with to just 4.6% when using the unstructured prompt with constrained decoding. These findings demonstrate that schema enforcement alone is insufficient and models require prompts explicitly designed to guide structured output generation. Without such guidance, models struggle to adapt their natural generation patterns with the imposed structural constraints.

Models		Constrained	$\text{Constrained}_{\text{Unconstrained-Prompt}}$
Llama3-8B	Base	59.6	37.7
	Instruct	72.7	**71.5**
Deepseek	Base	53.3	4.8
	Chat	56.9	53.5
Deepseek-Coder	Base	58.1	12.1
	Instruct	57.2	57.3
Mistral-7B	Base	71.5	4.6
	Instruct	**73.7**	70.9
OLMo-7B	Base	38.0	14.8
	Instruct	50.6	17.4
Phi-3	Instruct	63.2	70.0

Table 5: Performance comparison of constrained decoding with adapted prompts against constrained decoding against natural prompts. The best metric per type is marked in **bold**.

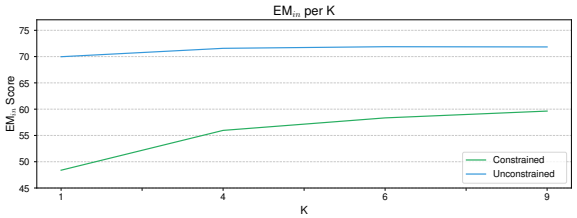

Figure 2: Average EM_{IN} scores across all models as a function of the number of in-context examples (k). Constrained decoding shows steeper improvements with additional examples compared to unconstrained decoding.

5.2 How Many Examples Are Necessary?

We examined how the number of in-context examples (k-shot learning) affects model performance under both constrained and unconstrained decoding. We hypothesize that providing more examples would improve performance by clarifying both the task requirements and the expected output format.

We evaluate models with k ranging from 0 to 9 shots, measuring average EM_{IN} scores across all models for each configuration.

Figure 2 shows that constrained decoding has substantial improvement as k increases, while unconstrained decoding achieves only modest gains. This disparity suggests that models not specifically trained for structured output generation require more demonstrative examples to effectively learn the output format constraints.

Combined with our findings from subsection 5.1, these results establish that effective constrained decoding benefits from explicit structural specifications in prompts and sufficient demonstrative examples.

5.3 Does Reasoning Improve Performance?

Inspired by Kojima et al. (2022)'s findings that explicit reasoning steps enhance language model performance, we investigate whether incorporating explanatory reasoning into the structured output format improves answer quality.

We augment the output structure with a `thinking` field that precedes the `answer` field in SQuAD tasks. Prompts are modified to explicitly request explanatory reasoning, using instructions such as "Provide an explanation for your answer before stating the final answer in the *thinking* field."

Table 6 shows that this modification generally degrades performance across models, with some high performance drops (e.g., Phi-3 declined from 63.2% to 36.8%). These results suggest that managing both reasoning generation and structural compliance simultaneously may exceed current model capabilities, leading to degraded performance on both aspects of the task.

Models		Constrained	$\text{Constrained}_{\text{Thinking}}$
Llama3-8B	Base	59.6	21.0
	Instruct	72.7	**70.5**
Deepseek	Base	53.3	36.4
	Chat	56.9	40.0
Deepseek-Coder	Base	58.1	43.1
	Instruct	57.2	55.4
Mistral-7B	Base	71.5	40.4
	Instruct	**73.7**	37.3
OLMo-7B	Base	38.0	21.0
	Instruct	50.6	13.2
Phi-3	Instruct	63.2	36.8

Table 6: Performance impact of requiring explanatory reasoning before answer generation. The best metric per type is marked in **bold**.

5.4 Log Probability Analysis

To understand the mechanism underlying performance degradation in constrained generation, we analyze log probabilities for cases where unconstrained generation succeeded but constrained generation failed. For each such case, we compared: (1) the log probability of the first answer token under unconstrained generation, and (2) the log probability of the first token following the answer delimiter under constrained generation.

Figure 3 reveals significantly lower log probabilities under constrained generation. This indicates that grammar constraints prevent models from generating their preferred tokens, typically natural language constructs like explanatory phrases or contextual statements. Instead, constrained decoding enforces less confident alternatives that comply with the structural requirements.

The substantial probability gap suggests that models' highest-confidence predictions often involve natural language patterns incompatible with strict structural constraints. This forces selection from lower-probability distributions, explaining the observed performance degradation.

Our analysis extends Wang et al. (2024)'s findings on misalignment between first-token predictions and final outputs by identifying the specific mechanism: grammar constraints force models away from their preferred natural language patterns into less confident structured alternatives.

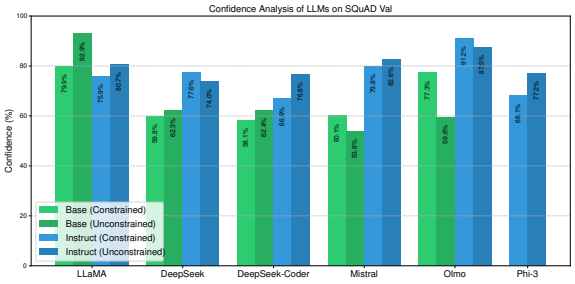

Figure 3: Log probability comparison between unconstrained and constrained generation for cases where unconstrained succeeded but constrained failed. Lower probabilities under constrained generation indicate reduced model confidence when forced to comply with structural constraints.

6 Conclusion

In this paper we investiaged how constrained decoding affects LLMs when generating structured JSON outputs. Our comprehensive evaluation reveals several critical insights with significant implications for both research and practical applications.

First, we discovered a fundamental divergence between base and instruction-tuned models under constrained decoding. Base models often benefit from structural constraints, producing more precise and concise outputs. Contrarily, instruction-tuned models exhibit mixed results: while maintaining performance on classification tasks, they suffer notable degradation on generation tasks. This finding suggests a practical implication for model development: researchers and practitioners could assess a base model's inherent capacity by testing it with constrained decoding before investing resources in instruction tuning. Such pre-tuning evaluation could serve as a indicator of the model's post-tuning performance, enabling more informed decisions about which base models are best suited for instruction-tuning.

Second, model performance show strong sensitivity to structural complexity. While simple structured outputs are no challenge, performance deteriorates quickly as output structures become more complex. This degradation is evident when comparing simple question-answering tasks with our advanced multi-question evaluation scenarios.

Third, successful constrained decoding requires more than just schema enforcement. It requires a combination of carefully crafted prompts and sufficient demonstrative examples. Our experiments reveal that models cannot effectively adapt to structural constraints through prompt engineering alone; they also require in-context examples to learn the output format. While unconstrained generation shows modest improvements with additional examples, constrained decoding shows steeper performance gains as the number of demonstrations increases. This finding underscores that effective structured output generation necessitates both explicit structural guidance in prompts and adequate few-shot examples to help models internalize the required format.

Fourth, our log probability analysis reveals the mechanism underlying performance degradation: constrained decoding prevents models from generating their preferred natural language constructs, forcing selection from lower-confidence token distributions that comply with structural requirements. This constraint-induced confidence reduction explains much of the observed performance decline in constrained settings.

These findings highlight promising avenues for future research. Key directions include integrating structured output examples directly into instruction-tuning or incorporating constrained decoding mechanisms into the training process itself. By exposing models to structural constraints during training rather than only at inference time, we could potentially develop models that naturally generate well-formed structured outputs without the performance penalties currently observed.

We release our evaluation framework and code at https://github.com/Maxscha/complex-constrained-decoding-eval to facilitate further research in this area.

Acknowledgement

We thank the German Federal Ministry for Education and Research (BMBF) for their compute grant through the project *KI-Servicezentrum Berlin Brandenburg* (01IS22092).

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Caio César Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Parul Chopra, Allie Del Giorno, Gustavo de Rosa, Matthew Dixon, Ronen Eldan, Dan Iter, Amit Garg, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett,

Jamie Huynh, Mojan Javaheripi, Xin Jin, Piero Kauffmann, Nikos Karampatziakis, Dongwoo Kim, Mahoud Khademi, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Chen Liang, Weishung Liu, Eric Lin, Zeqi Lin, Piyush Madan, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Xia Song, Masahiro Tanaka, Xin Wang, Rachel Ward, Guanhua Wang, Philipp Witte, Michael Wyatt, Can Xu, Jiahang Xu, Sonali Yadav, Fan Yang, Ziyi Yang, Donghan Yu, Chengruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. 2024. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2017. Guided open vocabulary image captioning with constrained beam search. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 936–945, Copenhagen, Denmark. Association for Computational Linguistics.

DeepSeek-AI, Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, Zhewen Hao, Ying He, Wenjie Hu, Panpan Huang, Erhang Li, Guowei Li, Jiashi Li, Yao Li, Y. K. Li, Wenfeng Liang, Fangyun Lin, A. X. Liu, Bo Liu, Wen Liu, Xiaodong Liu, Xin Liu, Yiyuan Liu, Haoyu Lu, Shanghao Lu, Fuli Luo, Shirong Ma, Xiaotao Nie, Tian Pei, Yishi Piao, Junjie Qiu, Hui Qu, Tongzheng Ren, Zehui Ren, Chong Ruan, Zhangli Sha, Zhihong Shao, Junxiao Song, Xuecheng Su, Jingxiang Sun, Yaofeng Sun, Minghui Tang, Bingxuan Wang, Peiyi Wang, Shiyu Wang, Yaohui Wang, Yongji Wang, Tong Wu, Y. Wu, Xin Xie, Zhenda Xie, Ziwei Xie, Yiliang Xiong, Hanwei Xu, R. X. Xu, Yanhong Xu, Dejian Yang, Yuxiang You, Shuiping Yu, Xingkai Yu, B. Zhang, Haowei Zhang, Lecong Zhang, Liyue Zhang, Mingchuan Zhang, Minghua Zhang, Wentao Zhang, Yichao Zhang, Chenggang Zhao, Yao Zhao, Shangyan Zhou, Shunfeng Zhou, Qihao Zhu, and Yuheng Zou. 2024. DeepSeek LLM: Scaling Open-Source Language Models with Longtermism.

Daniel Deutsch, Shyam Upadhyay, and Dan Roth. 2019. A general-purpose algorithm for constrained sequential inference. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 482–492, Hong Kong, China. Association for Computational Linguistics.

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.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Raghavi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hannaneh Hajishirzi. 2024. OLMo: Accelerating the Science of Language Models.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. 2024. DeepSeek-Coder: When the Large Language Model Meets Programming – The Rise of Code Intelligence.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Eva Hasler, Adrià de Gispert, Gonzalo Iglesias, and Bill Byrne. 2018. Neural machine translation decoding with terminology constraints. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 506–512, New Orleans, Louisiana. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao,

Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Minhyeok Lee. 2023. A Mathematical Investigation of Hallucination and Creativity in GPT Models. *Mathematics*, 11(10):2320.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, Tianyang Liu, Max Tian, Denis Kocetkov, Arthur Zucker, Younes Belkada, Zijian Wang, Qian Liu, Dmitry Abulkhanov, Indraneil Paul, Zhuang Li, Wen-Ding Li, Megan Risdal, Jia Li, Jian Zhu, Terry Yue Zhuo, Evgenii Zheltonozhskii, Nii Osae Osae Dade, Wenhao Yu, Lucas Krauß, Naman Jain, Yixuan Su, Xuanli He, Manan Dey, Edoardo Abati, Yekun Chai, Niklas Muennighoff, Xiangru Tang, Muhtasham Oblokulov, Christopher Akiki, Marc Marone, Chenghao Mou, Mayank Mishra, Alex Gu, Binyuan Hui, Tri Dao, Armel Zebaze, Olivier Dehaene, Nicolas Patry, Canwen Xu, Julian McAuley, Han Hu, Torsten Scholak, Sebastien Paquet, Jennifer Robinson, Carolyn Jane Anderson, Nicolas Chapados, Mostofa Patwary, Nima Tajbakhsh, Yacine Jernite, Carlos Muñoz Ferrandis, Lingming Zhang, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2024. StarCoder 2 and The Stack v2: The Next Generation. *ArXiv preprint*.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1314–1324, New Orleans, Louisiana. Association for Computational Linguistics.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code Llama: Open Foundation Models for Code. *arXiv preprint*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. 2024. Dolma: An Open Corpus of Three Trillion Tokens for Language Model Pretraining Research.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller,

Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models.

Xinpeng Wang, Bolei Ma, Chengzhi Hu, Leon Weber-Genzel, Paul Röttger, Frauke Kreuter, Dirk Hovy, and Barbara Plank. 2024. "my answer is C": First-token probabilities do not match text answers in instruction-tuned language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7407–7416, Bangkok, Thailand. Association for Computational Linguistics.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. Jailbroken: How does LLM safety training fail? In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Brandon T Willard and Rémi Louf. 2023. Efficient guided generation for llms. *arXiv preprint arXiv:2307.09702*.

Congying Xia, Chen Xing, Jiangshu Du, Xinyi Yang, Yihao Feng, Ran Xu, Wenpeng Yin, and Caiming Xiong. 2024. FOFO: A Benchmark to Evaluate LLMs' Format-Following Capability.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-Following Evaluation for Large Language Models.