ULTRABENCH: Benchmarking LLMs under Extreme Fine-grained Text Generation

Longfei Yun and **Letian Peng** and **Jingbo Shang***
University of California, San Diego
{loyun, lepeng, jshang}@ucsd.edu

Abstract

Fine-grained control is essential for precise and customizable text generation, yet existing benchmarks evaluate models on only a few attributes, typically fewer than five. We introduce ULTRABENCH, a new benchmark for extremely fine-grained controllable generation (EFCG), which evaluates large language models (LLMs) under dense, multi-attribute constraints. Each sample includes approximately 70 attributes, combining LLM-extracted soft constraints (e.g., style and tone) with programmatically enforced hard constraints (e.g., word count). Using ULTRABENCH, we conduct a comprehensive evaluation of state-of-the-art LLMs and prompting strategies. Models such as GPT-4.1 and Qwen3-8B perform well on individual constraints, achieving instruction-level accuracy above 70%, but consistently fail to satisfy all constraints simultaneously. To understand this limitation, we analyze model behavior across different dimensions. First, we observe a clear position bias: models tend to prioritize constraints presented later in the prompt while neglecting those that appear earlier. Second, we find that structural-related constraints are significantly more difficult to satisfy than content- or style-based ones, suggesting that current models struggle to coordinate global structure with token-level control. Finally, our error analysis reveals distinct failure modes: GPT-4.1 often attempts to follow constraints but falls short in precision, whereas LLaMA frequently omits constraints, particularly in multi-turn settings. These findings highlight fundamental limitations in EFCG and underscore the need for new methods that support dense, instruction-sensitive generation[1].

1 Introduction

Large language models (LLMs) have demonstrated impressive instruction-following capabilities (Lou et al., 2023; OpenAI et al., 2024), enabling them to interpret natural language commands and perform complex tasks such as multi-turn dialogues (Ouyang et al., 2022), question answering (Zhang et al., 2023), and structured writing (Sun et al., 2024). These capabilities have powered the rapid development of controllable text generation (CTG) in various domains (Zhou et al., 2023a; Chang et al., 2024).

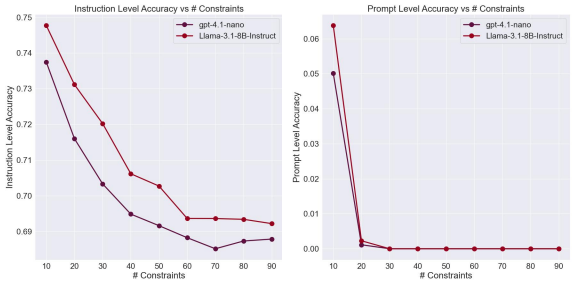

Figure 1: Instruction-level accuracy (left) and prompt-level accuracy (right) as a function of the number of constraints for GPT-4.1-nano and LLaMA-3.1-8B-Instruct. As the number of constraints increases, instruction-level accuracy gradually declines, while prompt-level accuracy drops sharply, indicating the increasing difficulty of satisfying all constraints simultaneously under extreme fine-grained controllability settings.

However, real-world applications including personalized content generation, product recommendation, and creative writing, often require simultaneously satisfying dozens of fine-grained constraints. These constraints span content elements, stylistic features, structural formats, and pragmatic goals, far exceeding the complexity handled in existing benchmarks, which typically focus on 3–5 attributes or incrementally added constraints in multi-turn setups (Huang et al., 2023; Jiang et al., 2023; He et al., 2024). As illustrated in Figure 1, model performance declines as the number of constraints increases, even for state-of-the-art LLMs. While these models may achieve reasonable accuracy on individual constraints, their ability to satisfy all constraints simultaneously deteriorates rapidly un-

* Corresponding author.

[1]Code and data is available at https://github.com/LongfeiYun17/ultrabench

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 15438–15453
November 4-9, 2025

der denser constraint settings. This highlights the growing challenge of extreme fine-grained controllability as constraint scales.

To address this gap, we propose ULTRABENCH, a benchmark for Extremely Fine-Grained Controllable Generation (EFCG). ULTRABENCH synthesizes training data with over 70 mixed hard and soft constraints per sample from realistic high-quality corpus, leveraging large language models for soft attribute extraction and programmatic rules for hard constraint enforcement. The key idea behind ULTRABENCH is to construct a large and diverse set of constraint types, some of which can be instantiated into multiple attributes; by adapting naturally occurring texts to accommodate more constraints and enriching them with 3–5 soft constraints extracted via GPT-4.1 (OpenAI, 2025), we build high-quality, densely labeled examples that reflect realistic fine-grained control demands. To ensure data quality, we incorporate a multi-stage quality control pipeline involving both automated validation and human verification (see §2.6). ULTRABENCH offers a rigorous testbed for evaluating LLMs' ability to satisfy dense, diverse, and interactive control conditions.

Through extensive experiments on ULTRABENCH, we conduct a comprehensive evaluation of several state-of-the-art LLMs. Our findings are as follows:

- ULTRABENCH presents a significantly more challenging benchmark than existing controllable generation datasets, highlighting considerable room for improvement in this area.
- Constraints appearing later in the prompt are more likely to be satisfied, while earlier ones are often ignored, indicating attention and memory limitations in long-context inputs.
- Compared to surface-level content or style constraints, global layout and structure constraints (e.g., sections, paragraph segmentation) are frequently violated, revealing limitations in compositional reasoning.
- Proprietary models often detect and attempt to follow most constraints but fails to execute them precisely. In contrast, open-source small models frequently omit constraints, reflecting weaker contextual memory and constraint grounding.

Overall, these results highlight substantial challenges in current LLMs' controllability under extremely fine-grained settings. ULTRABENCH provides a rigorous and realistic testbed to drive future research toward more reliable and interpretable controllable text generation.

2 ULTRABENCH Construction

In this section, we first formally define the EFCG task in §2.1, and then introduce the construction process of ULTRABENCH. We categorize the constraints into two types, hard constraints (§2.2) and soft constraints (§2.3), and describe how each type is instantiated in the benchmark. In §2.4, we discuss the key capabilities required for successful performance on this benchmark. We then present the constraint distribution statistics in §2.5, and describe the quality control procedures used to ensure data integrity in §2.6.

2.1 Problem Formulation

The task of EFCG generalizes traditional controllable text generation (CTG) by requiring models to satisfy a large set of diverse and precise constraints simultaneously. Formally, given an input X and a constraint set c, the objective is to generate an output Y such that Y adheres to c:

$$P(Y \mid X, c) = \prod_{i=1}^{n} P_\theta(Y_i \mid Y_{<i}, X, c)$$

where n is the output length, θ denotes model parameters, and $Y_{<i}$ is the generated prefix.

2.2 Hard Constraints

Hard constraints are verifiable constraints defined by deterministic rules that can be programmatically checked, such as requiring a fixed number of paragraphs, enforcing keyword presence, or forbidding specific punctuation. In contrast, soft attributes like tone or style rely on subjective interpretation.

Hard Constraints Typology We implement a comprehensive set of 27 hard attribute types spanning six functional categories (see Appendix A). These include:

1. **Language & Language Code**: constraints that specify the response language.
2. **Linguistic Style & Surface Form**: case sensitivity (uppercase/lowercase), punctuation restrictions, quotation formatting, and stylistic highlights;
3. **Length & Structural Layout**: sentence/word/-paragraph counts, section divisions, and structured layouts like bullet points;
4. **Content Presence & Semantics**: requirements on keyword inclusion/exclusion, named entities, numeric values, and specific content units;

5. **Content Frequency & Distribution**: constraints on the repetition frequency of keywords, letters, or capital words;
6. **Output Format & Encoding**: response formatting in JSON, hyperlink inclusion, constrained or question endings, and placeholder enforcement.

Each constraint is expressed in natural language to resemble realistic user prompts and is paired with a validator that automatically checks compliance. For example, the instruction *Your response should contain at least 12 sentences.* is evaluated by a rule-based checker that tokenizes the output and verifies the sentence count meets the threshold.

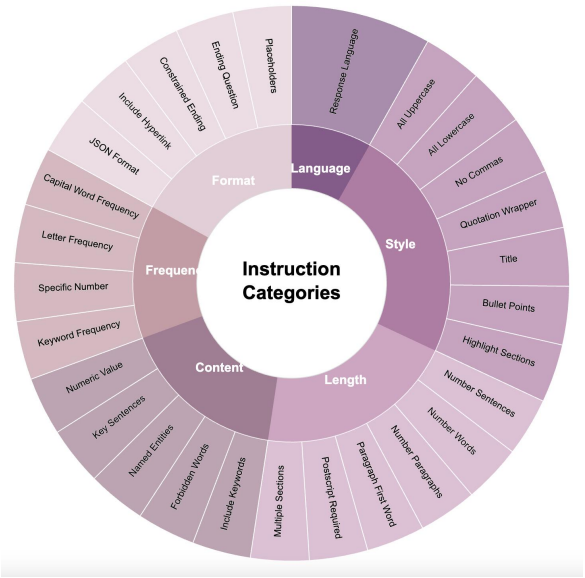

Figure 2: Hierarchical ring chart illustrating the taxonomy of instruction categories and subcategories used in EFCG task.

Attribute Identification and Augmentation Unlike synthetic prompt-generation methods (Lambert et al., 2024b), which often result in low output diversity (Long et al., 2024), our dataset is built by identifying and minimally adapting inherent properties of real-world texts to support a wide range of verifiable constraints.

We begin with the FineWeb corpus (Penedo et al., 2024), filtering samples based on token length and retaining those between 256 and 2,048 tokens. This range excludes short, noisy content (Soldaini et al., 2024) and avoids excessively long texts that pose computational challenges. We exclude short texts mainly because they do not provide sufficient capacity or structure for testing such dense constraint adherence and would therefore not serve our primary evaluation goals. Moreover, short texts in large-scale pretraining corpora often include low-information content (e.g., navigation menus, copyright disclaimers, advertisements). For each retained sample, we extract a variety of pre-existing textual attributes, including sentence count, named entities, capitalized terms, and bullet structures. These attributes are then rephrased into natural language constraints to simulate realistic user instructions.

Programmatic Text Transformation To support more constraints for each text, we apply a set of rule-based transformations to each input. These include structural edits such as inserting markdown highlights, splitting content into sections, and replacing entities with bracketed placeholders (e.g., `[location]`). We also enforce content-level constraints by injecting specific sentences and appending required postscript formats. These transformations enable each text to satisfy dozens of fine-grained, verifiable requirements.

Example of Soft Attribute Extraction

Input Text: Electric vehicles (EVs) are not only a solution to rising fuel costs, but also a major step toward environmental sustainability. With increasing government incentives and expanding charging infrastructure, EVs are becoming more accessible to everyday consumers.

Extracted Soft Attribute Instructions:

- **Content Theme:** Write about the economic and environmental benefits of electric vehicles.
- **Situation Context:** Assume the audience consists of environmentally conscious consumers.
- **Writing Style:** Use an informative and journalistic writing style.
- **Tone / Emotion:** Maintain a positive and encouraging tone throughout.
- **Example Pattern:** Include two to three concrete advantages as examples.

2.3 Soft Constraints

Rather than assigning a large number of soft attributes to each text, which can lead to semantic overlap and unreliable supervision due to their subjective nature, we focus on five core dimensions that are naturally grounded in most writing: (1) content theme, (2) situation context, (3) writing style, (4) emotional tone, and (5) example usage pattern. These dimensions provide orthogonal control handles for generation, enabling fine-grained guidance without introducing artificial or speculative attributes.

2.4 Core Abilities Needed in ULTRABENCH

To successfully perform the EFCG task, language models must demonstrate the following capabilities:

1. **Multi-Constraint Tracking:** The model must jointly track heterogeneous constraints, including discrete (e.g., keyword counts), continuous (e.g., word limits), structural (e.g., formatting), and symbolic (e.g., named entities or identifiers), maintaining a constraint-aware internal state throughout generation.
2. **Global-Local Coordination:** Models must align high-level structure (e.g., section layout) with low-level lexical fidelity (e.g., exact token usage), as decisions across levels are tightly coupled.
3. **Constraint-Aware Planning:** Effective generation requires anticipating future constraint targets, allocating content budget, and avoiding early commitments that jeopardize later satisfaction.
4. **Compositional Reasoning:** Constraints interact in complex ways, such as between sentence count and keyword placement, requiring models to resolve conflicts and compose feasible solutions that jointly satisfy all requirements.

We design ULTRABENCH to systematically evaluate these capabilities across diverse and densely constrained generation scenarios, enabling stress-testing of fine-grained controllability in modern LLMs.

2.5 Constraint Statistics

The `train` split contains 12,844 examples, while the `test` split contains 1,293 examples. On average, each example includes 74.6 hard constraints and 4.79 soft constraints in the `train` set, and 74.7 hard constraints and 4.80 soft constraints in the `test` set. While the total number of constraints per example is consistent, the type distribution is imbalanced to reflect different levels of control. Global constraints (e.g., `JSON Format`, `Title`, `Quotation Wrapper`) apply broadly across texts, while local constraints (e.g., `Keywords`, `Named Entities`) are input-specific and depend on semantic compatibility. We provide a detailed overview of the hard constraint distribution in ULTRABENCH benchmark in Figure 3.

2.6 Quality Control

We apply separate quality control procedures for hard and soft constraints to ensure the consistency and high quality of the dataset.

Hard Constraint Validation. To avoid violations of mutually exclusive constraints (e.g., `All Uppercase` vs. `All Lowercase`), each transformed sample is programmatically validated to ensure that all assigned constraints are simultaneously satisfied. Only examples that pass all checks are retained.

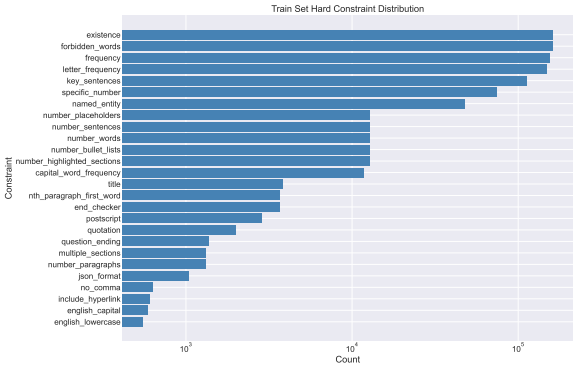

Figure 3: Distribution of hard constraint categories in the training set.

Soft Attribute Verification. To ensure the reliability of extracted soft attributes, we implement a two-stage verification process:

- **Stage 1: LLM-based Validation** After extracting soft attributes using a strong LLM, we conduct a second-pass check. Given the original text and the extracted attributes, the model evaluates each attribute independently with a binary "YES" or "NO" decision. If any attribute is marked "NO," the entire example is discarded, filtering out hallucinated or unsupported soft attributes. The extraction and judgment prompts could be found in Appendix B.
- **Stage 2: Human Spot-Check** The human annotators randomly sample 200 examples for manual review. Annotators are required to identify a supporting span from the source text for each soft attribute. Out of the 200 examples, 197 showed complete alignment, yielding a consistency rate of 98.5%. Minor deviations were observed in only 3 cases.

While minor inconsistencies may persist due to the subjective nature of soft attributes, this multi-stage process ensures strong overall alignment between the text and its annotated constraints.

2.7 Evaluation Metrics

To comprehensively evaluate EFCG capabilities, we adopt two complementary metrics for hard constraints: *Instruction-level Accuracy (IA)* and *Prompt-level Accuracy (PA)*, along with a model-based evaluation protocol for soft constraints, termed *Constraint Satisfaction Rate (CSR)*.

Instruction-level Accuracy. This metric assesses the model's ability to satisfy individual constraints, offering a fine-grained view of controllability under multi-attribute setups. Formally, given m examples where the ith example contains $n^{(i)}$ constraints, the instruction-level accuracy (IA) is defined as:

$$\text{IA} = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{n^{(i)}} \sum_{j=1}^{n^{(i)}} s_j^{(i)}$$

where $s_j^{(i)} \in \{0, 1\}$ indicates whether the jth constraint in the ith example is satisfied.

Prompt-level Accuracy. This metric captures whether the model satisfies *all* constraints within a single prompt, reflecting its ability to maintain compositional correctness across multiple simultaneous requirements. Formally, it is defined as:

$$\text{PA} = \frac{1}{m} \sum_{i=1}^{m} \prod_{j=1}^{n^{(i)}} s_j^{(i)}$$

Constraint Satisfaction Rate (CSR). For soft constraints, we rely on model-based evaluation. A strong LLM judge (e.g., GPT-4.1 (OpenAI, 2025)) is prompted with the generated text and its associated soft constraint list. For each constraint, the judge produces a binary output (*Yes* if fully satisfied, *No* otherwise), based solely on the observable content. CSR is computed analogously to IA:

$$\text{CSR} = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{n^{(i)}} \sum_{j=1}^{n^{(i)}} s_j^{(i)}$$

where $s_j^{(i)} \in \{0, 1\}$ is the binary decision from the judge. To minimize randomness in judgment, we set the decoding temperature to 0.1 to encourage deterministic generation.

Together, these metrics offer a comprehensive evaluation framework that reflects both fine-grained constraint adherence and holistic generation quality.

2.8 Comparison With Existing Datasets

Existing controllable generation benchmarks, such as IFBench (Pyatkin et al., 2025), FollowBench (Jiang et al., 2023), and CFBench (Zhang et al., 2024), typically evaluate only 3–5 constraints per instance. This limited scope quickly saturates with current SOTA models and fails to reveal fine-grained differences in instruction-following quality. By contrast, imposes over 70 constraints per example, combining hard and soft attributes to enable fine-grained assessment of control, compositionality, and robustness under high cognitive load. Moreover, it uniquely requires both surface-level fidelity (e.g., entities, digits, keywords) and global structural planning (e.g., layout, formatting, JSON compliance), reflecting the dual-level controllability increasingly demanded in real-world applications such as personalized writing and enterprise report generation.

Benchmarks	Num.	Type.	Meth.
IFEval	541	4	Python
CELLO	523	4	Python
FollowBench	820	5	LLM
InFoBench	500	5	LLM + Python
FoFoBench	494	1	LLM
ComplexBench	1150	4	LLM
CFBench	1000	10–25	LLM + Python
ULTRABENCH	1288	29	LLM + Python

Table 1: Comparison of different benchmarks

3 Experiments

Our experiments provide a comprehensive evaluation of state-of-the-art language models on ULTRABENCH benchmark. We aim to answer the following research questions:

- How well do current SOTA LLMs perform on the EFCG task in ULTRABENCH? (§3.1, §3.2)
- To what extent does constraint position within the prompt affect model adherence? (§3.3)
- Which types of constraints are the most difficult to satisfy under extreme conditions? (§3.4)
- What are the underlying causes of failure when models face densely constrained prompts? (§3.5)
- Can post-training on ULTRABENCH improve model performance on EFCG tasks? (§3.6 §3.7)

3.1 Models and Baselines

We evaluate a wide range of state-of-the-art large language models (LLMs) on ULTRABENCH, including LLaMA-3.1-8B-Instruct (Dubey et al., 2024), Qwen-3-8B (Team, 2025), as well as Tulu-3.1-8B (Lambert et al., 2024a). We also include proprietary models such as GPT-4.1 and GPT-4o (Achiam et al., 2023), Gemini-2.5-Flash (Anil et al., 2023), and Claude-3.5-Sonnet-v2 (Anthropic, 2024).

Model	Zero-shot		Grouped		Stepwise		Multi-turn-2		Multi-turn-3	
	IA	CSR	IA	CSR	IA	CSR	IA	CSR	IA	CSR
Qwen-3-8B	70.22	67.58	**73.99**	70.63	71.33	62.63	66.83	**71.69**	68.52	67.92
Llama-3.1-8B-Inst	72.33	88.66	**74.83**	90.21	72.27	88.26	71.39	**92.62**	70.71	90.14
Tulu-3.1-8B	72.22	84.58	**77.32**	**87.19**	66.07	71.16	60.36	51.26	52.99	26.09
GPT-4.1	80.26	96.16	**84.34**	**96.73**	78.76	95.51	79.98	96.59	79.67	95.22
GPT-4o	78.37	93.84	**79.43**	93.98	77.57	92.68	76.41	**94.44**	76.82	93.79
Gemini-2.0-Flash	79.92	**95.01**	**85.40**	92.53	76.88	91.49	76.71	83.69	74.84	79.87
Claude-3-5-Haiku	77.44	91.49	**79.44**	91.04	77.99	**92.07**	77.22	87.08	76.86	86.30

Table 2: Instruction adherence (IA) and constraint satisfaction rate (CSR) across prompting methods. (prompt-level accuracy, PA, is omitted because it is 0 for all settings).

Model	Zero-shot					Grouped Constraints					Stepwise Planning					Multi-turn-2				
	Str	Fmt	Ling	Cont	Freq	Str	Fmt	Ling	Cont	Freq	Str	Fmt	Ling	Cont	Freq	Str	Fmt	Ling	Cont	Freq
Qwen-3-8B	50.10	77.28	81.49	76.61	62.80	50.33	**81.64**	**81.97**	**77.07**	**71.18**	**65.13**	66.48	75.29	78.09	63.46	49.54	65.82	77.19	69.64	64.05
Llama-3.1-8B-Inst	56.52	49.59	68.88	**81.56**	63.78	**60.00**	**53.14**	**69.24**	80.08	**71.21**	55.90	49.03	74.11	80.68	64.37	50.63	38.28	65.70	80.79	62.11
Tulu-3.1-8B	59.44	44.16	84.66	78.99	65.37	**65.54**	**50.15**	**84.85**	**80.50**	**75.16**	57.11	34.48	69.57	71.33	61.66	49.33	21.55	59.47	62.54	60.64
GPT-4.1	67.13	86.25	92.19	89.12	69.20	**70.29**	**87.27**	**91.89**	**86.67**	**81.89**	61.56	90.82	93.98	88.30	66.64	67.04	76.11	90.65	87.55	71.09
GPT-4o	68.14	60.69	91.20	84.49	**71.55**	**70.58**	60.14	**90.86**	**82.20**	76.79	67.87	**68.91**	90.32	83.51	70.47	64.10	57.30	89.50	83.72	68.34
Gemini-2.5-pro	53.43	89.85	**92.41**	**90.49**	67.68	**61.65**	**92.34**	92.11	89.94	**80.97**	50.63	86.41	90.41	87.08	65.02	52.34	86.41	89.87	83.91	68.37
Claude-3-5-Haiku	63.27	62.47	88.09	83.08	71.59	61.65	61.61	**90.38**	83.05	**76.52**	**65.34**	**68.91**	89.99	**83.11**	72.24	62.01	47.36	84.78	81.76	**73.83**

Table 3: Category-level accuracy (%) for the five constraint types: Structure (Str), Format (Fmt), Linguistic (Ling), Content (Cont), and Frequency (Freq). Bold indicates the highest score and underlined the second highest within each model.

To assess the controllability of these models, we evaluate the following baseline prompting methods:

- **Zero-shot:** The model generates text given the complete list of hard and soft constraints in a single prompt, without any additional structuring or guidance.
- **Grouped Constraints:** Constraints are grouped into semantically related categories (e.g., content, style, structure) to shorten the prompt length and reduce cognitive load during generation.
- **Stepwise Planning:** The model first generates an intermediate plan or outline from the constraints, and then conditions on this plan to produce the final output.
- **Multi-turn:** Constraints are evenly partitioned across multiple turns. In each turn, the model generates a partial response based on the new subset of constraints and the preceding generation.

3.2 Overall Results

We use Sglang (Zheng et al., 2024) as the serving framework, with the following decoding parameters: `max_new_tokens`=2048, `temperature`=0.7, `top_p`=0.95, and `repetition_penalty`=1.0. Table 2 presents the performance of various LLMs and prompting strategies on ULTRABENCH. Among all evaluated models, GPT-4.1 achieves the highest instruction-level accuracy in the zero-shot setting at 80.26%, followed by Gemini-2.0-Flash (79.92%) and GPT-4o (78.37%). Despite their strong per-constraint performance, all models achieve near-zero prompt-level accuracy (PA), underscoring the inherent difficulty of satisfying all constraints simultaneously.

Grouped Constraints consistently yield the highest IA across all models, outperforming both Zero-shot and other prompting strategies. For instance, Tulu-3.1-8B improves from 72.22% (Zero-shot) to 77.32% (Grouped), a gain of +5.10%, demonstrating the benefit of presenting constraints in semantically structured groups. Zero-shot prompting typically ranks second, offering reasonable performance without additional planning or organization. In contrast, Stepwise Planning and Multi-turn prompting underperform, with Multi-turn-3 yielding the lowest IA and CSR across all models. This degradation likely stems from increased demands on constraint tracking: stepwise and multi-turn formats require the model to manage evolving constraint states and maintain long-range memory, which current LLMs struggle to do reliably. As shown in Table 3, across all prompting strategies, models perform best on linguistic style constraints, often exceeding 90% accuracy, reflecting strong lexical and syntactic control. In contrast, formatting constraints remain the most challenging, particularly under multi-turn prompting, where accuracy frequently drops below 60%, suggesting limited

capability for maintaining global formatting consistency.

3.3 Position Bias

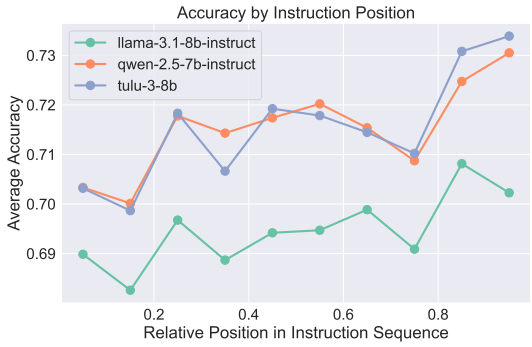

Figure 4: Average instruction-level accuracy by relative constraint position in the prompt. Later-positioned constraints are more likely to be satisfied, indicating a position bias across models.

In this section, we investigate whether large language models exhibit position bias when handling multiple hard constraints. To this end, we conduct a controlled experiment by randomly shuffling the order of instructions in the prompt. We select 1,080 evaluation samples and permute the constraint sequences before generation, aiming to disentangle the effects of position and inherent constraint difficulty.

Model responses are generated with a maximum length of 2,048 tokens and evaluated for instruction-level accuracy. For analysis, we group results by each instruction's relative position, computed as its index normalized by the total number of constraints in the prompt.

Figure 4 shows the instruction-level accuracy across relative positions for three models: `LLaMA-3.1-8B-Instruct` (Dubey et al., 2024), `Qwen-2.5-7B-Instruct` (Yang et al., 2024), and `Tulu-3-8B` (Lambert et al., 2024b). All models show a clear position bias, with later instructions more likely to be satisfied. This indicates a systemic issue in multi-constraint generation: LLMs tend to prioritize later instructions while underperforming on earlier ones.

To further understand this behavior, we analyze how attention is distributed across the prompt. We split each prompt into three segments evenly (early, middle, late), excluding the first token due to attention sink effects (Xiao et al., 2023). Model outputs are generated via greedy decoding. During generation, we log decoder self-attention weights every 20 steps across all layers and heads, aggregating attention over the three regions. This analysis is conducted on 256 randomly sampled prompts.

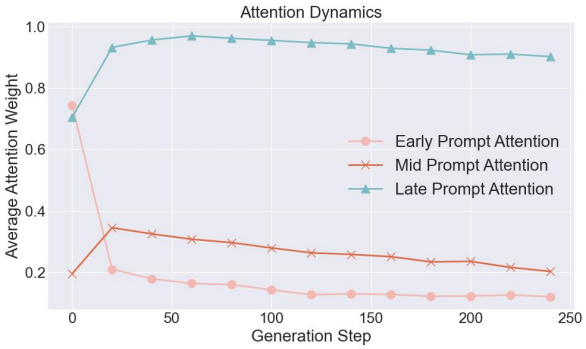

Figure 5: Average attention weights over generation steps across early, middle, and late prompt regions. Attention to late prompt tokens dominates throughout the generation, while attention to early instructions quickly decays, indicating position bias and potential omission of earlier constraints. Attention was recorded every 20 steps using greedy decoding.

As shown in Figure 5, attention to the late prompt region remains consistently high throughout the generation, while early regions receive persistently lower attention. These results reinforce the presence of position bias and suggest that LLMs disproportionately attend to recent instructions, potentially reducing global constraint adherence.

3.4 Attribute-Level Failure Patterns

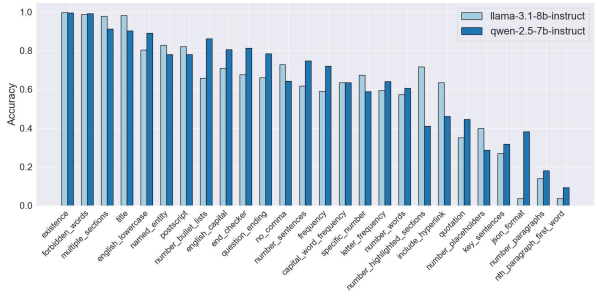

Figure 6: Bars represent model performance on individual constraint types, sorted by average difficulty. While both models perform well on surface-level constraints (e.g., content presence, style cues), accuracy declines significantly on structure-related and formatting-specific instructions, revealing challenges in global layout control and fine-grained formatting compliance.

To better understand which types of fine-grained constraints pose the greatest challenge to current LLMs, we evaluate instruction-level accuracy across all constraint categories for two representative models: `llama-3.1-8b-instruct` and `qwen-2.5-7b-instruct`. As shown in Figure 6, both models perform well on shallow, surface-level

Error Type	Definition	Example
Constraint Violation	Constraint is present in the output, but violated.	Output has 100 words when asked for at least 300; bullet count is incorrect.
Constraint Omission	The constraint has no trace in the output, the model ignored it entirely.	No keyword at all; missed key sentence or end-sentence constraints.
Soft Constraint Mismatch	Output has incorrect tone, style, or content.	Asked for optimistic tone, but output is neutral or negative.
Refusal or Non-response	Output is too generic or short to assess whether constraints are satisfied.	"I'm sorry, I can't help with that."

Table 4: Categorization of common error types observed in EFCG. Each type reflects a distinct failure mode in constraint adherence.

constraints such as keyword inclusion and forbidden words. However, accuracy drops substantially for constraints involving structural reasoning, such as JSON formatting, paragraph-level alignment, and key sentence insertion. These results indicate that failures in EFCG are not merely due to constraint overload but stem from fundamental limitations in modeling global-local compositionality.

3.5 Error Analysis

We categorize the most common errors into four types (Table 4): (1) **Constraint Violation**: The model attempts to satisfy a constraint, but fails to meet its exact requirements. (2) **Constraint Omission**: The model entirely ignores a constraint, with no attempt to address it. (3) **Soft Constraint Mismatch**: The response deviates from the expected tone, style, or emotional intent specified by soft constraints. (4) **Refusal or Non-response**: The model refuses to answer, deflects the instruction, or generates vague and generic output that lacks sufficient content to assess constraint adherence. We use an LLM-based evaluator to classify error types efficiently.

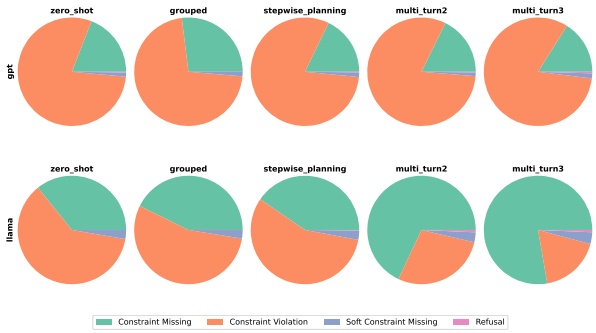

Figure 7: Each pie chart shows the proportion of four error types. GPT-4.1 errors are dominated by constraint violations across all settings, whereas LLaMA3 exhibits a shift from violations to omissions in multi-turn settings, indicating challenges with long-context constraint tracking.

We observe distinct error patterns between GPT-4.1 and LLaMA3 models across prompting strategies. For GPT-4.1, **constraint violation** remains the dominant error type across all settings, suggesting that the model consistently attempts to fulfill constraints but often fails to meet precise specifications. This indicates strong instruction-following intent, but limited precision in fine-grained constraint execution.

In contrast, LLaMA3 exhibits a different trajectory. Under zero-shot, grouped, and stepwise prompting, constraint violation remains the most frequent error type, but constraint missing also accounts for a substantial portion of failures. This indicates that the model not only struggles with precise execution but also frequently overlooks constraints altogether. In multi-turn settings, the error distribution shifts further, with omission errors surpassing violations, suggesting increasing difficulty in maintaining constraint representations over longer contexts. These patterns highlight LLaMA's limitations in long-context retention and constraint grounding, particularly when instructions are distributed across multiple conversational turns.

3.6 SFT Improves Hard but Hurts Soft Constraints

We fine-tuned three models: Llama-3.2-3B (Dubey et al., 2024), Qwen2.5-3B-Instruct (Yang et al., 2024), and Gemma-2-2b-it (Team, 2024) on the training set of ULTRABENCH using supervised fine-tuning (SFT), where the adapted text served as the supervision signal. Evaluation of the test set in the zero-shot setting reveals a consistent trade-off: SFT leads to substantial gains in instruction adherence (IA), particularly for Gemma (+12.7) and Qwen2.5 (+5.5), suggesting improved understanding and alignment with hard constraints. However, these gains come at the cost of degraded constraint satisfaction rate (CSR), especially for Llama-3.2 and Gemma, where CSR drops by 24.5 and 11.0 points, respectively.

Models	IA	PA	CSR
Llama-3.2-Instruct	70.75	0.00	82.95
Llama-3.2-Instruct W/ SFT	75.88	0.78	58.45
Qwen2.5-3B-Instruct	70.07	0.00	85.50
Qwen2.5-3B-Instruct W/ SFT	75.58	0.00	75.23
Gemma-2-2b-it	24.42	0.00	17.03
Gemma-2-2b-it W/ SFT	37.12	0.00	5.99

Table 5: Performance comparison for SFT model and base model.

Model	IA	CSR
Base Model	70.75	82.95
SFT Model	75.88	58.45
DPO Model	73.06	54.30

Table 6: Performance of different models on ULTRABENCH using IA and CSR.

3.7 RL Fine-tuning Challenges in Controllable Generation

To evaluate ULTRABENCH under reinforcement learning paradigms, we fine-tuned LLaMA-3.2-3B-Instruct with Direct Preference Optimization (DPO) (Rafailov et al., 2023), using GPT-4o to generate candidate responses and selecting preferences via a weighted combination of IA and CSR. While the DPO model achieved higher IA than the base model, its CSR dropped sharply, indicating that optimizing scalar preference signals may improve instruction adherence at the cost of stylistic and semantic fidelity. This highlights a fundamental challenge in controllable generation: densely packed, heterogeneous constraints are difficult to capture with single-dimensional rewards, motivating future work on constraint-aware reinforcement learning methods.

4 Case Study

We show an example in Figure 8. In this case, the model demonstrated several notable successes. It followed the required structure with a proper Markdown title, highlighted section, placeholders, and sufficient length, while also incorporating most of the mandated keywords (such as airplane, steward, flagged, and molto). It further satisfied the NER constraints by including both a PERSON (Chef Marco) and GPE entities (Italy, Curinga), and it maintained the intended casual and lighthearted blog style throughout.

At the same time, the model exhibited important failures. It omitted the PRODUCT entity, missed three lexical requirements related to keyword frequency and a fixed sentence, and violated strict character-level quotas (for g, i, and d). Placeholders were present but not aligned with PRODUCT constraints, and attempts to repeat certain keywords such as glutine created tradeoffs with the character-level rules.

Overall, this case illustrates how the model handles high-level stylistic and structural demands effectively, but struggles with symbolic constraints and multi-attribute alignment. It underscores the need for constraint-aware decoding strategies and more nuanced reward formulations in controllable generation.

5 Related Work

Controllable Text Generation CTG tasks involve hard constraints (e.g., text length, keyword inclusion)(Takase and Okazaki, 2019; Carlsson et al., 2022) and soft constraints (e.g., sentiment, topic)(Gu et al., 2022; Lu et al., 2022). Fine-tuning LLMs with instructional data improves their constraint-following ability (Weller et al., 2020; Sanh et al., 2021; Mishra et al., 2022; Jiang et al., 2024), but evaluations show LLMs often fail to meet all constraints (Jiang et al., 2023; Qin et al., 2024; Ren et al., 2025). Despite this, these works primarily focus on a relatively small number of attributes or conditions, typically from 3 to 5, leaving a gap in understanding LLM's performance under more extreme requirements.

Evaluation of CTG Evaluating LLM's adherence to constraints is challenging and typically involves automatic and programmatic assessments using various metrics (Yao et al., 2023; Zhou et al., 2023c; Chen et al., 2022). Zhou et al. (2023b) centers on assessing 25 verifiable instructions. Jiang et al. (2023) progressively integrates fine-grained constraints to develop multi-level instructions, thereby enhancing complexity across six distinct types. Wen et al. (2024) constructs a novel benchmark by synthesizing and refining data from the aforementioned benchmarks, with an emphasis on the combinatorial types of constraints. Zhang et al. (2024) proposes a comprehensive constraint-following benchmark over 50 NLP tasks. However, none of them investigate the effects of extreme fine-grained attributes.

6 Conclusions and Future Work

We present ULTRABENCH, a benchmark designed to stress-test LLMs under extremely fine-grained controllability. Through comprehensive evaluation, we uncover systematic limitations in current models' ability to handle 70+ compositional constraints, such as position bias and structural failures.

Future work may investigate training-time interventions such as compositional instruction tuning and constraint-aware decoding. Incorporating richer supervision signals, including constraint-level rewards or curriculum-based training, could improve model adherence to complex constraint sets.

7 Acknowledgement

Our work is sponsored in part by NSF CAREER Award 2239440, NSF Proto-OKN Award 2333790, Sponsored Research Projects from companies like Cisco and eBay, as well as generous gifts from Google, Adobe, and Teradata. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and should not be interpreted as necessarily representing the views, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes not withstanding any copyright annotation hereon.

Limitations

While ULTRABENCH underscores the challenges of extremely fine-grained controllable generation, several caveats remain. First, the current inventory of roughly 29 constraint types is broad but not exhaustive; nuanced cases such as nested or conditional dependencies are left for future work. Second, soft-constraint adherence is judged by another LLM, which may introduce subtle evaluation bias. Finally, we benchmark moderate context lengths only; extending ULTRABENCH to long-context, multi-turn settings with dynamically evolving constraint scenarios remains an open direction.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rohan Anil et al. 2023. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Anthropic. 2024. Claude 3.5 sonnet. Accessed: 2025-05-05.

Fredrik Carlsson, Joey Öhman, Fangyu Liu, Severine Verlinden, Joakim Nivre, and Magnus Sahlgren. 2022. Fine-grained controllable text generation using non-residual prompting. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6837–6857.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45.

Howard Chen, Huihan Li, Danqi Chen, and Karthik Narasimhan. 2022. Controllable text generation with language constraints. *arXiv preprint arXiv:2212.10466*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yuxuan Gu, Xiaocheng Feng, Sicheng Ma, Lingyuan Zhang, Heng Gong, and Bing Qin. 2022. A distributional lens for multi-aspect controllable text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1023–1043, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yun He, Di Jin, Chaoqi Wang, Chloe Bi, Karishma Mandyam, Hejia Zhang, Chen Zhu, Ning Li, Tengyu Xu, Hongjiang Lv, et al. 2024. Multi-if: Benchmarking llms on multi-turn and multilingual instructions following. *arXiv preprint arXiv:2410.15553*.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Preprint*, arXiv:2305.08322.

Yuxin Jiang, Yufei Wang, Chuhan Wu, Wanjun Zhong, Xingshan Zeng, Jiahui Gao, Liangyou Li, Xin Jiang, Lifeng Shang, Ruiming Tang, Qun Liu, and Wei Wang. 2024. Learning to edit: Aligning llms with knowledge editing. *CoRR*, abs/2402.11905.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2023. Followbench: A multi-level fine-grained constraints following benchmark for large language models. *arXiv preprint arXiv:2310.20410*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. 2024a. Tülu 3: Pushing frontiers in open language model post-training.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. 2024b. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. 2024. On llms-driven synthetic data generation, curation, and evaluation: A survey. *arXiv preprint arXiv:2406.15126*.

Renze Lou, Kai Zhang, and Wenpeng Yin. 2023. A comprehensive survey on instruction following. *arXiv preprint arXiv:2303.10475*.

Ximing Lu, Sean Welleck, Jack Hessel, Liwei Jiang, Lianhui Qin, Peter West, Prithviraj Ammanabrolu, and Yejin Choi. 2022. Quark: Controllable text generation with reinforced unlearning. In *Advances in Neural Information Processing Systems*, volume 35, pages 27591–27609. Curran Associates, Inc.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland. Association for Computational Linguistics.

OpenAI. 2025. Introducing GPT-4.1 in the api. Accessed: 2025-05-05.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. 2024. The fineweb

datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. 2025. Generalizing verifiable instruction following. *Preprint*, arXiv:TODO.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. 2024. Infobench: Evaluating instruction following ability in large language models. *arXiv preprint arXiv:2401.03601*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Qingyu Ren, Jie Zeng, Qianyu He, Jiaqing Liang, Yanghua Xiao, Weikang Zhou, Zeye Sun, and Fei Yu. 2025. Step-by-step mastery: Enhancing soft constraint following ability of large language models. *arXiv preprint arXiv:2501.04945*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, et al. 2024. Dolma: An open corpus of three trillion tokens for language model pretraining research. *arXiv preprint arXiv:2402.00159*.

Haoran Sun, Lixin Liu, Junjie Li, Fengyu Wang, Baohua Dong, Ran Lin, and Ruohui Huang. 2024. Conifer: Improving complex constrained instruction-following ability of large language models. *arXiv preprint arXiv:2404.02823*.

Sho Takase and Naoaki Okazaki. 2019. Positional encoding to control output sequence length. In *Proceedings of NAACL-HLT*, pages 3999–4004.

Gemma Team. 2024. Gemma.

Qwen Team. 2025. Qwen3: Think deeper, act faster.

Orion Weller, Nicholas Lourie, Matt Gardner, and Matthew E. Peters. 2020. Learning from task descriptions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1361–1375, Online. Association for Computational Linguistics.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, et al. 2024. Benchmarking complex instruction-following with multiple constraints composition. *arXiv preprint arXiv:2407.03978*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Shunyu Yao, Howard Chen, Austin W Hanjie, Runzhe Yang, and Karthik Narasimhan. 2023. Collie: Systematic construction of constrained text generation tasks. *arXiv preprint arXiv:2307.08689*.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

Tao Zhang, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, Bin Cui, et al. 2024. Cfbench: A comprehensive constraints-following benchmark for llms. *arXiv preprint arXiv:2408.01122*.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. 2024. Sglang: Efficient execution of structured language model programs. *Advances in Neural Information Processing Systems*, 37:62557–62583.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023a. Instruction-following evaluation for large language models. *Preprint*, arXiv:2311.07911.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023b. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. 2023c. Controlled text generation with natural language instructions. In *International Conference on Machine Learning*, pages 42602–42613. PMLR.

A Hard Attributes

To evaluate fine-grained controllability, ULTRABENCH defines 27 verifiable hard constraint types, grouped into six major categories based on control intent: (1) language specification, (2) linguistic style and surface form, (3) length and structural layout, (4) content semantics, (5) content frequency and distribution, and (6) output formatting and encoding. These constraints are automatically applied and verified during data construction, enabling precise and reproducible supervision across diverse dimensions.

Each hard attribute is explicitly checkable via string-level rules. For instance, structural constraints such as exact sentence or paragraph counts (`Number Sentences`, `Number Paragraphs`), or formatting constraints like JSON compliance or markdown bullet points, can be programmatically validated. Content-based constraints require models to insert specific keywords, named entities, or key sentences extracted from the original passage, while frequency constraints enforce fine-grained control over token or letter repetition.

By covering both global and local aspects of generation, including layout, surface form, semantic content, and output structure, these hard attributes allow us to systematically evaluate a model's ability to follow detailed and interacting generation instructions. A complete listing and description of all hard attributes are provided in Table 7.

B Prompts

Prompt for Soft Constraints Extraction. To construct soft constraints that guide the generation process at a stylistic and semantic level, we design a dedicated prompt that instructs a language model to extract up to five high-level attributes from a given input text. These attributes include content theme, situational context, writing style, tone, and example usage pattern. As shown in Appendix B, the prompt asks the model to generate concise instructional sentences for each attribute type, but only when the attribute is clearly supported by the source text. This selective and instruction-formatted extraction ensures that the soft constraints are grounded in the input and suitable for controllable generation tasks.

Prompt for Soft Attribute Extraction

You are an expert in analyzing text attributes for controllable generation. Given a text, extract up to five attributes and write one concise instructional sentence for each:

1. **Main Content:** Summarize the main topic and phrase it as an instruction (e.g., "Write an article about...").
2. **Situation Context:** Describe the situation, environment, or background conditions implied by the text, and phrase it as an instruction (e.g., "Assume the text takes place during an environmental summit.").
3. **Writing Style:** Specify the writing style (e.g., "Use a journalistic writing style.").
4. **Tone or Emotion:** Specify the tone or emotional attitude (e.g., "Maintain an optimistic and persuasive tone.").
5. **Example Pattern:** If examples are present, describe their pattern (e.g., "Provide at least three reasons in bullet points.").

Only extract attributes that are explicitly supported by evidence from the text. Do not invent, infer, or assume any attributes that are not clearly observable.
If an attribute is not clearly present, skip it.
Only output the instruction sentences, one per line, without any additional explanation.
Input: {text}

Prompt for Constraint Satisfaction Verification. To evaluate whether a generated response satisfies a set of fine-grained constraints, we design a judgment prompt that treats the model as a strict verifier. Given a generated text and a list of constraints, the prompt instructs the model to assess each constraint independently based solely on the surface evidence in the text. The model must answer "YES" only if a constraint is fully and unambiguously satisfied, and "NO" if it is violated, partially satisfied, or unclear. To ensure clarity and consistency, the prompt requires one binary judgment per constraint, with no additional explanation or free-form commentary. This verification prompt enables efficient, automatic scoring of constraint adherence in dense controllable generation tasks.

Prompt for Constraint Satisfaction Verification

You are a strict evaluator of constraint satisfaction in generated text. Given a set of constraints and a generated text, determine for each constraint whether it is fully satisfied by the text. You must base your judgment only on the content of the text, without guessing or inferring missing information.
For each constraint, answer strictly "YES" if it is clearly and fully satisfied, or "NO" if it is partially satisfied, unclear, or not satisfied.
In the final line, output {number} lines, each containing only YES or NO, indicating whether the answer satisfies each constraint. Do not generate other irrelevant text.

—

Generated Text:
{text}

Constraints:
{constraints}

—

Answer:

Prompt for Stepwise Planning. As part of the Stepwise Planning baseline, we decompose gen-

Category	Instruction	Description of the verifiable constraint
	1. Language & Language Code	
Language	Response Language	Entire response must be written in a specified ISO-639-1 language (e.g., "en", "zh").
	2. Linguistic Style & Surface Form	
Case	All Uppercase	Entire response must be in UPPERCASE.
Case	All Lowercase	Entire response must be in lowercase (no capitals).
Punctuation	No Commas	Commas are disallowed in the response.
Formatting	Quotation Wrapper	Wrap the entire response in double quotation marks.
Formatting	Title	Include a title wrapped in `<<double angle brackets>>`.
Formatting	Bullet Points	Provide exactly N markdown bullet items.
Formatting	Highlight Sections	Highlight at least N phrases using `*italic*` or `**bold**`.
	3. Length & Structural Layout	
Length	Number Sentences	Response must contain *less than / at least* N sentences.
Length	Number Words	Response must contain *less than / at least* N words.
Length	Number Paragraphs	Response must contain exactly N paragraphs separated by "`\n\n`" or `***`.
Options	Paragraph First Word	Produce N paragraphs; the k^{th} starts with a specified word.
Formatting	Postscript Required	Add a postscript starting with "P.S." (or similar) at the end.
Formatting	Multiple Sections	Ensure the response is divided into N named sections.
	4. Content Presence & Semantics	
Keywords	Include Keywords	All specified keywords must appear in the response.
Keywords	Forbidden Words	Specified words must *not* appear anywhere.
Keywords	Named Entities	Response must contain named entities of specified types (e.g., PERSON, ORG).
Keywords	Key Sentences	Include exactly M sentences from a given sentence set.
Keywords	Numeric Value	Include a number within a specified range (e.g., 10–20).
	5. Content Frequency & Distribution	
Keywords	Keyword Frequency	A keyword must appear *less than / at least* K times.
Keywords	Specific Number	A number from the input must appear a specific number of times.
Character Frequency	Letter Frequency	A given letter must appear *less than / at least* K times.
Character Frequency	Capital Word Frequency	ALL-CAPS words must appear *less than / at least* K times.
	6. Output Format & Encoding	
Formatting	JSON Format	Output must be valid JSON, optionally wrapped in markdown code fences.
Options	Include Hyperlink	Output must include at least one valid HTTP(S) hyperlink.
Options	Constrained Ending	Output must end with a specified phrase; nothing follows.
Options	Ending Question	Output must end with a question (e.g., rhetorical or clarifying).
Formatting	Placeholders	Include at least N bracketed placeholders such as `[address]` or `[date]` in the response.

Table 7: Reorganized list of 27 verifiable `Instruction` subclasses grouped into six constraint categories: language specification, surface form, structure, content semantics, frequency control, and output format.

eration into two phases: first, the model produces a high-level structural plan based on the provided constraints; second, it generates the final text conditioned on this plan. The prompt shown in Appendix B corresponds to Step 1 only, where the model is instructed to output a concise outline that organizes the structural aspects of the target text—such as paragraph count, bullet point usage, or placement of quotations. Crucially, the model is explicitly prohibited from generating any final output text at this stage. This planning phase is designed to encourage global structure awareness before lexical realization.

Prompt for Stepwise Planning (Step 1 Only)

You are a helpful assistant tasked with generating a plan for producing a text that satisfies a large number of specific constraints.
Your task is to perform **Step 1 only**: Given the constraints below, generate a plan that describes how the final output should be structured to satisfy all constraints. Do **not** generate the final text.
Focus on:

- How to organize the structure (e.g., paragraph count, bullet points, quotation placement).

—
Instruction:
`{instruction}`
—
Write a concise outline that addresses the structural category. Do not generate any final output text or other irrelevant text.
Step 1: Plan

C Case Study

This case study Figure 8 illustrates how ULTRABENCH evaluates controllable generation under dense and diverse constraints. The task setup specifies both soft attributes (topic, style, and tone) and hard attributes (lexical quotas, structural and formatting rules, character-level limits, NER requirements, and banned tokens). The generated response excerpt demonstrates partial success: it satisfies many lexical and entity constraints, uses the required stylistic framing, and integrates placeholders. At the same time, it also exposes failures such as structural mismatches and potential overuse or underuse of certain characters. This example highlights ULTRABENCH's ability to reveal both strengths and weaknesses of models in fine-grained controllability.

Case Study: Gluten-Free Travel in Italy

Task Setup. We instructed the model to generate a blog post about gluten-free travel experiences in Italy, framed around genealogy research. The task required a casual tone while satisfying dense constraints:

- **Soft Attributes:**
  - Write a blog post about gluten-free travel experiences in Italy.
  - Assume the text takes place during a vacation involving genealogy research in Italy.
  - Use a casual and conversational writing style.
  - Maintain an appreciative and lighthearted tone.
- **Hard Attributes:**
  - Lexical constraints: Inclusion of 18 specific keywords (e.g., airplane, steward, senza, flagged) with frequency requirements
  - Structural/formatting constraints: Markdown title, square-bracket placeholders, use of highlighted section, no bullet points
  - Character-level constraints: e.g., 'g' $<$ 34 times, 'k' $\geq$ 6 times, 'z' $\geq$ 1 time, 'i' $<$ 69 times
  - NER constraints: Must contain at least one PERSON and one GPE entity, and one PRODUCT
  - Negation constraints: Must not include 13 banned tokens (e.g., "current", "pull", "score")

Generated Response (Excerpt). `<<Gluten-Free Adventures: Eating Senza Glutine While Chasing My Roots in Italy>> \n \n I always thought traveling with a senza glutine (gluten-free) diet would be tough, especially on a vacation packed with genealogy research in a small Italian town like Curinga. But with this post, I would like to report that it has been very easy to eat gluten-free in Italy. From the moment I stepped off my Alitalia flight, flagged suitcase in hand, I felt welcome. Ehh... it was typical airplane food. Was the food good? Let's just say, the steward did his best, and I didn't know about it until I heard the steward speaking to a passenger behind me about the senza glutine option. \n\n*highlighted section* \n The making of a truly enjoyable gluten-free travel experience started with my first meal in [restaurant name], where the chef (PERSON: Chef Marco) personally walked me through their senza glutine menu. The food was molto tasty, and I felt like royalty! Even the local bakery at [address] had a special flagged tray for glutine-free goodies. I was blown away by how seriously they took cross-contamination, making me feel like eating in Italy wasn't a challenge, but a pleasure.\n\n Genealogy research in Curinga led me to relatives I never knew existed, and every meal with them was a celebration of food, family, and discovery. If you're worried about traveling with dietary restrictions, follow my blog for more tips, because Italy is an amazing option for anyone senza glutine. There's nothing like enjoying a big plate of pasta and knowing it's safe for you. Plus, I even found a gluten-free pizza at [pizzeria name] that was out of this world! Whether you're on a vacation or chasing your roots, let your taste buds explore.`

Figure 8: Case study showing successes and failures on a dense, multi-constraint ULTRABENCH task.