Inference-Time Feedback for Reasoning Controllability in Diffusion Language Models

Clovis Barbour and Huixin Zhan
New Mexico Institute of Mining and Technology
Socorro, NM, USA
{clovis.barbour, huixin.zhan}@nmt.edu

Abstract

Scientific NLP systems often require outputs that satisfy strict, machine-checkable constraints. In this work, we study structured-generation controllability along three axes: structural control, iterative correction, and decoding dynamics. Diffusion decoding is of particular interest because its iterative refinement may improve global structure and revision behavior, but may also introduce distinct failure modes such as termination instability and repetition. To quantify controllability, we evaluate compliance with five machine-checkable constraints: (i) required headings and (ii) correct ordering, which reflect global structural control; (iii) explicit end markers and (iv) per-section bullet constraints, which probe local constraint adherence; and (v) repetition avoidance, which captures generation stability under different decoding dynamics. We use these metrics to assess both single-pass generation and changes under iterative correction. Our goal is to isolate structural reliability under parser-facing requirements rather than to directly measure scientific correctness. Across our benchmark, diffusion models tend to better preserve global structure, while iterative improvement substantially improves explicit termination and other local control constraints. Hybrid systems show mixed behavior depending on decoding order. These results suggest that machine-checkable controllability can be usefully decomposed into global structure versus local control, and that the two may benefit from different inference-time strategies.

1 Introduction

In scientific NLP applications, model outputs often serve as interfaces to downstream systems that assume strict structural requirements. Recent evidence from systematic reviews of automated data extraction shows that large language models are increasingly used in scientific pipelines, while evaluation practices remain inconsistent in ways that can bias perceived performance (Schmidt et al., 2025). Many such systems rely on automated evaluation scripts or parsers to extract or summarize information in a prescribed format (Feldhoff et al., 2025). Incorrect formatting, such as missing sections or tokens, is common in many NLP models (Neveditsin et al., 2025), and can cause severe downstream issues even when the output is semantically accurate. These failures motivate examining not only model capabilities, but also how different generation dynamics influence a model's ability to satisfy strict structural constraints.

Unlike autoregressive language models, which generate one token at a time until the sequence is complete, diffusion-based language models iteratively corrupt or mask a sequence and then denoise it until the full output is refined, following the broader diffusion modeling paradigm introduced by Ho et al. (2020). In NLP, diffusion models have increasingly been explored as an alternative generative method because they enable parallel token generation and exhibit distinct structural properties (Zou et al., 2023). Because they are non-sequential, diffusion models can produce more globally structured and contextually aware outputs, but common autoregressive mechanisms for ending sequences do not directly apply. In practice, this lack of explicit termination can make diffusion models more prone to repetition or partial completion. Recent work introduces grammar-level constraints for diffusion models (Mündler et al., 2025), but these approaches do not directly address practical failures such as premature termination or repetition across sections.

Motivated by the distinct structural and termination-related failure modes introduced by different decoding dynamics, we present an initial empirical framework for studying controllable structured generation in scientific NLP outputs. We evaluate autoregressive, diffusion, and hybrid systems under the same structural criteria, both with

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop), pages 675–682

and without a simple inference-time critique-and-revision loop.

Using this framework, we analyze how different controllability mechanisms interact with structural and termination-related constraints. Our goal is to isolate structural reliability under machine-checkable, parser-facing requirements rather than to directly evaluate scientific correctness. In our benchmark, diffusion-based decoding tends to better preserve global output structure, while inference-time iterative correction is particularly effective at improving local control constraints such as explicit termination and repetition avoidance. Taken together, these findings suggest a meaningful distinction between global structural controllability and local control mechanisms in scientific NLP generation.

Our contributions are as follows:

- We present an initial evaluation framework for structured outputs in scientific NLP settings with machine-checkable formatting requirements.
- We compare autoregressive, diffusion, and hybrid decoding mechanisms under identical structural constraints.
- We study a simple inference-time critique-and-revision loop for repairing constraint violations without fine-tuning or external tools.
- We provide evidence that global structure and local control behave differently across decoding mechanisms and may benefit from different inference-time strategies.

2 Related Work

2.1 Structured Outputs and Constraints

Structured output generation is common in NLP, and constrained decoding is one promising solution. Geng et al. (2025) evaluate methods for generating and benchmarking structured outputs. Unlike that work, however, we do not benchmark JSON-schema engines; instead, we analyze architectural decoding dynamics and inference-time correction for related constraints such as section ordering and termination markers. Whereas Geng et al. (2025) evaluate framework compliance, we evaluate failure modes that appear in unconstrained settings and are especially relevant to scientific pipelines. This distinction matters because JSON schemas do not fully capture the section semantics and termination behavior that are particularly important in such pipelines.

Prior work has also evaluated grammar-constrained decoding for structured NLP tasks, such as Park et al. (2024). However, these methods target only a subset of grammar-checkable tasks. Our constraints instead target scientific-output reliability, where failure modes extend beyond syntactic correctness to include section-wise bullet counts and repetition. These are difficult to encode in formal grammars or depend on the rest of the output for evaluation. Unlike prior work, we also evaluate the effect of model architecture on these metrics to characterize how different architectures behave under identical constraints.

In Wang et al. (2025), training-time constraint verification and alignment are used to improve structural characteristics. We also use verifiers, but only implicitly through the iterative improvement system rather than during training. Wang et al. (2025) show that models can be adapted to improve structural characteristics, while we test whether constraint adherence can be improved at inference time without training-time alignment. This helps isolate whether failures are architectural or correctable post hoc.

2.2 Self-Revision Loops

Iterative improvement methods at inference time have been explored as a way to improve output quality by allowing models to critique their own outputs and then revise them using the generated feedback. Self-Refine (Madaan et al., 2023) is one example of this approach, where successive refinement loops improve semantic quality and correctness. While this method has shown strong benefits, we test whether similar gains hold for strict structural constraints and whether the benefits vary by model architecture.

3 Evaluation Framework

3.1 Structured Output Task

Models were prompted to produce multi-section structured outputs for a variety of scientific tasks. Each prompt consisted of a task-specific question followed by fixed structural specifications indicating the required section headings, their order, length constraints, and a termination marker.

Specifically, outputs were required to include a predefined sequence of headings (e.g., `Goal`,

Domain	#	Task Types
Molecular & Cellular Biology	16	CRISPR; RNA-Seq; ChIP-Seq; ATAC-Seq; Proteomics; Metabolomics; Epigenetics; scRNA-Seq
Imaging & Neuroscience	5	Microscopy; Time-Lapse; Brain Imaging; 3D Tissue Imaging; Synaptic Plasticity
Clinical & Animal Studies	6	Mouse Efficacy; Toxicity; RCT Design; Phase II Endpoints; Flow Cytometry; Gating
Statistics & Computational Biology	8	Survival Analysis; Statistical Tests; Bioinformatics; Simulation; Sensitivity Analysis; ML Evaluation
Physical & Materials Sciences	7	Mechanics; Thermal Conductivity; Enzyme Kinetics; Spectroscopy; Semiconductors; Binding Kinetics
Ecology & Genetics	4	Biodiversity; Population Dynamics; Linkage Analysis; Heritability
Scientific Systems & Review	4	Data Extraction; Literature Review; Meta-Analysis Bias; Research Synthesis

Table 1: Prompt domains and representative task types.

`Controls`, `Analysis`), adhere to bullet-count constraints, avoid repeated bullets, remain under a fixed token budget, and terminate with a designated end marker. Implementation details are provided in the project repository.[1]

3.2 Prompt Suite and Task Categories

The prompt suite contains 50 prompts. Each prompt was designed to reflect realistic scientific reasoning and planning tasks. The prompts span several scientific NLP domains, as shown in Table 1.

Although these topics require different forms of reasoning, all prompts share the same structural constraints. The tasks stress models because they require semantic understanding of the prompt while simultaneously organizing the response according to a fixed format. This increases the likelihood of structural failures. The prompts also simulate outputs expected from protocol generators, automated literature analysis tools, or experimental planning systems, all settings in which scientific NLP models may be deployed.

3.3 Constraint Definitions and Scoring

Each model output is evaluated against five binary criteria:

1. **Required Headings**: all listed headings appear in the output.
2. **Ordering**: headings are in the correct order.
3. **Explicit End Marker**: the output terminates with the specified token.
4. **Bullet Constraints**: each section meets the required number of bullets (2–4).
5. **Repetition Avoidance**: no repeated headings, sections, or bullets.

No partial credit is given; each criterion is scored as pass or fail. The overall score is the sum of the five binary scores. All criteria are evaluated using deterministic string-based checks.

We use binary scoring because many downstream parsers and evaluators require exact validity, although this choice necessarily compresses near-miss cases that may still be partially useful to a human reader.

3.4 Iterative Improvement

Our iterative improvement system uses a three-step process:

1. **Initial Output**: the model is prompted with the same task prompt used for base generation.
2. **Critique**: the initial output is provided back to the model, along with instructions to critique the output with respect to the structural criteria.
3. **Revised Output**: the model is given the initial output, the critique, and a prompt to produce a revised final answer.

This process does not use any external tools or additional models. It provides a simple way to test whether some failures appear tightly tied to the decoding architecture or can be mitigated after initial inference.

4 Models and Systems

We evaluate autoregressive, diffusion-based, and hybrid language models at two parameter scales. At the 0.6B scale, we include an autoregressive baseline (Qwen3-0.6B), diffusion-based variants (Qwen3-0.6B-MDLM and Qwen3-0.6B-BD3LM from Tiny-A2D), and hybrid decoding pipelines

[1] https://github.com/R0g3rXYZ/ConstraintSatisfaction

that combine autoregressive and diffusion generation in both directions (AR→BD3LM and BD3LM→AR). Here, AR→Diffusion denotes a two-stage pipeline in which an autoregressive model first produces an initial draft or plan that is then refined by a diffusion model, whereas Diffusion→AR denotes the reverse ordering, with diffusion producing the initial output and autoregressive decoding performing the final rewrite. At the 8B scale, we evaluate an autoregressive baseline (Qwen3-8B), a diffusion-based model from the same family (WeDLM-8B; Liu et al., 2025), and corresponding hybrid decoding configurations (AR→WeDLM and WeDLM→AR).

To ensure fairness, each model is evaluated on the same prompt set, decoding settings are fixed within each model family, and iterative improvement is evaluated separately from base models. The maximum new token length is set to 256 for all models.

Hybrid models use two-stage decoding, where an initial model produces a structured outline and a second model refines the output for semantic completeness and formatting.

5 Experimental Protocol

5.1 Prompting and Output Budget

All experiments use the same prompt suite and a fixed structured suffix ("s1") that specifies: (i) the required headings `Goal/Controls/Readout/Analysis/Pitfalls`, (ii) per-section bullet-count constraints (2–4 bullets for all sections except `Goal`), (iii) a maximum output budget of 250 tokens, and (iv) an explicit termination marker `<END>` that must appear on its own line at the end of the answer. To ensure comparability across model families, we restrict each generation to at most 256 newly generated tokens.

When a model family provides a chat template, we use it to format prompts. We also disable explicit "thinking" modes and remove any leftover chain-of-thought content by stripping text that appears before a `</think>` token.

5.2 Autoregressive Decoding Settings

All autoregressive runs use the same output length cap (256 new tokens) and nucleus sampling with temperature 0.7 and nucleus threshold 0.95. Additional implementation details are provided in Appendix B.

5.3 Diffusion Decoding Settings

For all diffusion runs, we use the same output length cap (256 new tokens) and deterministic decoding (temperature 0.0). Implementation-specific hyperparameters (e.g., steps, block size, remasking, guidance) are provided in Appendix B.

5.4 Iterative Improvement Protocol

To test inference-time feedback as a controllability mechanism, we run the same fixed three-call draft–critique–revise loop for every base model:

1. **Draft**: generate an initial response to the task prompt.

2. **Critique**: condition on the task prompt and the draft, then request exactly three critique bullets targeting (i) structural and formatting errors, (ii) plausibility and consistency issues, and (iii) verbosity relative to the 250-token limit.

3. **Revise**: condition on the task prompt, draft, and critique; rewrite the answer to satisfy the same structural constraints and end with `<END>`.

To keep scoring consistent, we post-process each stage to enforce an explicit end marker: if `<END>` is missing, we append it on a new line; if trailing text appears after `<END>`, we truncate after the first `<END>` line.

To mitigate structural drift, we use conservative decoding configurations for both the critique and revision stages. For autoregressive models, the critique stage uses a low sampling temperature (0.1) with a nucleus threshold of 1.0 and an output limit of 128 newly generated tokens, while the revision stage uses a sampling temperature of 0.2 with the same nucleus threshold. For diffusion models, we reduce the number of diffusion steps and the allowed output length during critique while holding remasking and guidance fixed.

5.5 Hybrid Two-Stage Pipelines

We evaluate two hybrid decoding directions at each scale.

AR → Diffusion (plan → write). An autoregressive model first produces a structured JSON "plan" containing a short goal string and exactly three bullets per section (`Controls`, `Readout`, `Analysis`, `Pitfalls`). A diffusion model then renders the

final formatted answer using only the plan bullets, with minimal elaboration, and terminates with <END>. This tests whether diffusion decoding preserves the global schema better when conditioned on a discrete outline.

Diffusion → AR (draft → refine). A diffusion model first produces a free-form draft, and an autoregressive model then rewrites the draft to improve clarity and completeness. This tests whether AR refinement fixes diffusion failures (e.g., incomplete sections) or instead introduces structural drift.

5.6 Cross-Environment DAG for 8B Hybrids

For the 8B-scale hybrids, AR and diffusion inference run in separate runtime environments. To ensure clean and reproducible execution, we use a simple buffer-based directed acyclic graph (DAG): the stage-1 environment writes a JSONL buffer containing prompts and stage-1 outputs, and the stage-2 environment consumes this buffer to produce final outputs. This ensures that the exact same prompts and intermediate stage-1 generations are used regardless of which environment runs second.

5.7 Run Logging

For every model and pipeline, we log the prompt, output text, prompt and style metadata, measured wall-clock latency, approximate input and output token counts, and derived throughput (tokens/s). We then score these logs deterministically using the string-based constraint checks described in Section 3.3.

6 Results

6.1 Overall Structural Compliance

Figure 1 shows that compliance varies both by model and by constraint type. Rather than exhibiting uniformly strong or weak behavior, many models perform well on some constraints while struggling on others, which supports separating global structural constraints from local control constraints in the analysis.

6.2 Global vs. Local Controllability

To quantify the separation between global structure and local control, we compute a global score (required headings + ordering) and a local score (end marker + bullet constraints + repetition avoidance), and compare each base model to its corresponding iterative-improvement run. For each model, we compute the mean pass rate for each binary constraint across prompts and then sum those means: the global score sums Required Headings and Ordering (range 0–2), while the local score sums End Marker, Bullet Constraints, and Repetition Avoidance (range 0–3).

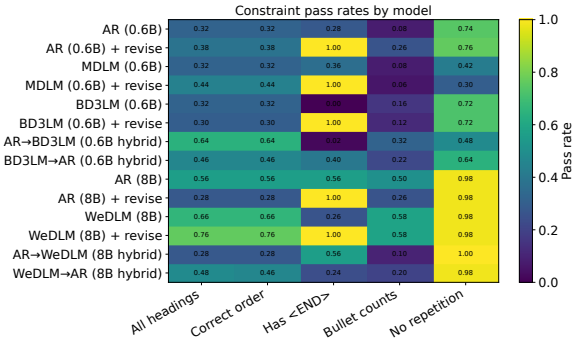

Figure 1: Per-constraint pass rates across evaluated models. Each cell shows the fraction of prompts for which a model satisfied the corresponding constraint, enabling direct comparison between global structural constraints (required headings and correct ordering) and local control constraints (explicit end marker, bullet-count validity, and repetition avoidance). Model labels indicate architecture, scale, hybrid decoding direction, and whether iterative revision is applied.

Model	Global Base	Local Base	Δ Global	Δ Local
wedlm_8b	1.320	1.820	0.200	0.740
ar_8b	1.120	2.040	-0.560	0.200
ar	0.640	1.100	0.120	0.920
bd3lm	0.640	0.880	-0.040	0.960
mdlm	0.640	0.860	0.240	0.500

Table 2: Global versus local controllability metrics (means) for each base model and the change after one iterative improvement cycle.

6.3 Iterative Improvement for Termination Control

Across the evaluated models, iterative improvement substantially increases pass rates on the end-marker criterion relative to the corresponding base model. This suggests that explicit termination is often not robustly maintained in base generation alone, but can be improved through critique-and-revision loops. In our setting, this pattern is more consistent with a local control problem than with a semantic one.

6.4 Diffusion Models

In our benchmark, diffusion models tend to preserve global structure, such as heading presence

and correct ordering, more consistently than autoregressive models. However, they also show weaker performance on explicit termination and bullet-count constraints. This pattern appears across the evaluated diffusion architectures.

6.5 Hybrid Systems

Hybridization produces more mixed outcomes than the base systems. In our experiments, hybrid models sometimes improve and sometimes reduce structural quality, with results depending in part on which decoding mechanism is used in the final stage.

7 Analysis

7.1 Structure vs. Control

To analyze how different controllability axes contribute to constraint satisfaction, we group the evaluated criteria according to the aspects of reasoning they probe. In our framework, *structural control* governs *global* properties of the output schema, while *decoding dynamics* and *iterative correction* primarily influence *local* generation behavior such as termination, repetition, and fine-grained constraint adherence.

Required headings and correct ordering reflect *structural control*, since they measure a model's ability to follow a prescribed global schema. In contrast, explicit end markers, per-section bullet constraints, and repetition avoidance probe *decoding-time stability and correction behavior*, capturing whether generation reliably terminates, respects local constraints, and avoids degeneration.

Across models, constraints related to decoding stability are most consistently improved through inference-time iterative correction, suggesting that feedback loops can serve as a useful control mechanism for local and termination-related constraints. Conversely, constraints associated with global structure are more often satisfied by diffusion-based and hybrid decoding approaches, which is consistent with the idea that sequence-level refinement may help preserve global structural coherence.

7.2 Iterative Improvement Reasoning

Iterative improvement is highly effective for strengthening local control constraints, but it can also degrade heading preservation and ordering. This pattern is consistent with the hypothesis that critique-and-revision loops improve isolated local failures while sometimes disrupting global structural organization.

8 Discussion

8.1 Practical Implications

For practitioners, the main takeaway is that compliance with machine-checkable constraints breaks down into two distinct properties: (i) *global schema preservation* (e.g., headings and ordering) and (ii) *local control* (e.g., termination, bullet counts, repetition). In practice, these behave differently and should be evaluated separately. Diffusion-style decoding is often a good default when global structure matters, but explicit checks are still needed for termination and other local constraints.

At the same time, these results should be interpreted as evidence about structural reliability under machine-checkable constraints rather than as a complete evaluation of scientific answer quality.

If strict compliance is required, inference-time feedback through a simple draft–critique–revise loop is a model-agnostic way to repair many local failures without fine-tuning. However, hybrid pipelines should be evaluated end to end using the same downstream verifiers, and feedback loops are best applied selectively, such as only when fast checks flag a failure, in order to control latency and cost.

8.2 Future Work

There are several natural extensions to this work. First, our current evaluation focuses on machine-checkable structure and control, but does not include semantic correctness. Adding task-specific semantic metrics or lightweight human evaluation would help determine whether improvements in structural compliance correspond to improvements in scientific usefulness, and would allow analysis of trade-offs between semantic quality and structural scores.

Second, our prompt suite uses a fixed heading schema across diverse tasks. Future work could analyze different schemas better matched to particular task families, such as experimental protocols versus literature synthesis, and study how schema choice changes the separation between global structure and local control.

Third, we evaluate only a single iterative improvement cycle. Evaluating multiple cycles, potentially with early stopping based on verifier failures, would help determine whether local control

constraints continue to improve, whether global structure degrades over successive revisions, and where diminishing returns occur.

Finally, our hybrid systems are relatively simple two-stage pipelines. More robust hybrid approaches could incorporate explicit structure-preserving constraints between stages, such as freezing headings, section boundaries, and bullet counts while allowing semantic refinement within sections, potentially improving end-to-end controllability without sacrificing useful edits.

9 Conclusion

Scientific NLP systems often serve as interfaces to downstream automated pipelines that require outputs to satisfy strict, machine-checkable structural and termination constraints. Current language models do not always meet these requirements reliably, even when their semantic content appears strong. Understanding how generation mechanisms influence constraint satisfaction is therefore important for reliable deployment in scientific workflows.

In this paper, we presented an initial empirical study of structural controllability under parser-facing constraints. In our benchmark, diffusion models tended to better preserve global structure, while iterative improvement substantially improved explicit termination and other local control constraints, sometimes at the cost of global organization. These findings suggest that structural validity should not be treated as a single capability: global structure and local control appear to behave differently across decoding mechanisms and may benefit from different inference-time strategies. More broadly, our results support evaluating structural constraints separately from semantic quality when analyzing the reliability of language models in scientific pipelines.

Acknowledgments

This research is supported by an Institutional Development Award (IDeA) from the National Institute of General Medical Sciences of the National Institutes of Health under grant number P20GM103451.

Limitations

Our study has several limitations. First, our evaluation isolates structure and control under machine-checkable constraints, but does not directly measure semantic correctness or scientific usefulness. A model can therefore satisfy all five constraints while still producing a weak, generic, or incomplete answer. Second, our binary scoring framework matches exact parser-facing validity, but it also compresses near-miss cases that may still be partially useful to a human reader. Third, using fixed constraints across all prompts may obscure task-specific differences in what counts as a good structured output. Fourth, we evaluate only a single round of iterative revision, leaving open the question of whether additional iterations would further improve local control or instead increase structural drift. Finally, our baseline set is intentionally focused on architectural decoding differences and a simple inference-time repair mechanism; broader comparisons with additional baselines would strengthen future versions of the study.

References

Kim Feldhoff, Hajo Wiemer, Philip Träger, Robert Kühne, Martina Zimmermann, and Steffen Ihlenfeldt. 2025. Automatic information extraction from scientific publications based on the use case of additive manufacturing. *Applied Sciences*, 15(17):9331.

Saibo Geng, Hudson Cooper, Michał Moskal, Samuel Jenkins, Julian Berman, Nathan Ranchin, Robert West, Eric Horvitz, and Harsha Nori. 2025. Generating structured outputs from language models: Benchmark and studies. *arXiv preprint arXiv:2501.10868*. Introduces JSONSchemaBench, a benchmark for evaluating constrained decoding across efficiency, coverage, and output quality.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851.

Aiwei Liu, Minghua He, Shaoxun Zeng, Sijun Zhang, Linhao Zhang, Chuhan Wu, Wei Jia, Yuan Liu, Xiao Zhou, and Jie Zhou. 2025. WeDLM: Reconciling diffusion language models with standard causal attention for fast inference. *arXiv preprint arXiv:2512.22737*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Sean Welleck, Shashank Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *NeurIPS 2023 Workshop on Large-Scale Language Models*. Introduces an iterative self-feedback loop in which a language model generates output, critiques it, and refines it using its own feedback.

Niels Mündler, Jasper Dekoninck, and Martin Vechev. 2025. Constrained decoding of diffusion LLMs with context-free grammars. *arXiv preprint arXiv:2508.10111*.

Nikita Neveditsin, Pawan Lingras, and Vijay Kumar Mago. 2025. Evaluating structured output robustness of small language models for open attribute-value extraction from clinical notes. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 286–296, Vienna, Austria. Association for Computational Linguistics.

Kanghee Park, Jiayu Wang, Taylor Berg-Kirkpatrick, Nadia Polikarpova, and Loris D'Antoni. 2024. Grammar-aligned decoding. *arXiv preprint arXiv:2405.21047*. Formalizes grammar-aligned decoding and proposes adaptive sampling techniques for constrained decoding.

Lena Schmidt, Ailbhe N. Finnerty Mutlu, Rebecca Elmore, Babatunde K. Olorisade, James Thomas, and Julian P. T. Higgins. 2025. Data extraction methods for systematic review (semi)automation: Update of a living systematic review. *F1000Research*, 10:401. Living systematic review highlighting the emergence of LLM-based data extraction methods in evidence synthesis tasks.

Fei Wang, Chao Shang, Sarthak Jain, Shuai Wang, Qiang Ning, Bonan Min, Vittorio Castelli, Yassine Benajiba, and Dan Roth. 2025. Aligning to constraints for data-efficient language model customization. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5310–5325. Association for Computational Linguistics. Proposes ACT, which uses automatic constraint verifiers to generate supervision signals and improve language model constraint adherence.

Yuanhao Zou, Yunsu Kim, and Jinyeong Kang. 2023. A survey of diffusion models in natural language processing. *arXiv preprint arXiv:2305.14671*.

A Reproducibility Details

All experiments are reproducible using the scripts and prompts in the project repository. The repository contains all command-line arguments and environment files needed to regenerate the data. The repository is available at:

https://github.com/R0g3rXYZ/ConstraintSatisfaction

B Full Decoding Settings

Autoregressive (AR). Nucleus sampling with `temperature=0.7`, `top_p=0.95`, and `max_new_tokens=256`.

Diffusion (MDLM). Blockwise masked denoising with `steps=256`, `block_size=16`, low-confidence remasking, `cfg_scale=0.0`, `temperature=0.0`, and `max_new_tokens=256`.

Diffusion (BD3LM). Staircase attention with `steps=256`, `block_size=32`, low-confidence remasking, `cfg_scale=0.0`, `temperature=0.0`, and `max_new_tokens=256`.

Diffusion (WeDLM). Native WeDLM inference with `max_new_tokens=256` and `temperature=0.0`.

Iterative improvement (critique/revision). For autoregressive models, critique uses `temperature=0.1`, `top_p=1.0`, and `max_new_tokens=128`, while revision uses `temperature=0.2` with `top_p=1.0`. For diffusion models, critique uses fewer diffusion steps and a shorter output budget than the main generation, while keeping remasking and guidance settings fixed.