Decoupling Task-Solving and Output Formatting in LLM Generation

Haikang Deng, Po-Nien Kung, Nanyun Peng
University of California, Los Angeles
{haikang, ponienkung, violetpeng}@cs.ucla.edu

Abstract

Large language models (LLMs) are increasingly adept at solving complex problems, such as mathematical reasoning and automatic evaluation. However, performance often degrades when prompts intertwine task instructions with rigid formatting requirements. This entanglement creates competing goals for the model, hindering its reasoning capabilities. To address this, we introduce DECO-G, a decoding framework that explicitly decouples format adherence from problem solving. DECO-G delegates format adherence to a separate Format Estimation Module (FEM), which performs probabilistic lookahead to estimate future format compliance rate and reweighs token probabilities, allowing the LLM to focus solely on task resolution. To make this approach both practical and efficient, we introduce three key innovations: instruction-aware distillation, a flexible trie-building algorithm, and HMM state pruning. Experiments across mathematical reasoning, event argument extraction, and LLM-as-a-judge demonstrate that DECO-G constantly gains over prompting or structured generation baselines, with guaranteed format compliance.

1 Introduction

Instruction fine-tuning (Wei et al., 2021; Chung et al., 2024) empowers LLMs to solve complex tasks, often enhanced by reasoning strategies like Chain-of-Thought (Wei et al., 2022) and Tree-of-Thought (Yao et al., 2023). However, emerging evidence suggests that combining problem-solving instructions with strict output-format requirements in a single prompt negatively impacts performance (Tam et al., 2024; Long et al., 2025; He et al., 2024). For instance, Long et al. (2025) demonstrate that output structure significantly affects accuracy on benchmarks like MMLU (Hendrycks et al., 2020), while Tam et al. (2024) observe that stricter constraints correlate with greater reasoning degradation. This suggests that the current paradigm of intertwining task and format instructions (as shown in Figure 1) may hurt LLMs' reasoning capabilities.

To mitigate this, recent works have explored relaxing format strictness (Tam et al., 2024) or adopting more intuitive schema (Long et al., 2025; He et al., 2024). Yet, these adjustments still impose constraints that distract LLMs from reasoning. Alternatively, constrained decoding frameworks (Beurer-Kellner et al., 2024; guidance-ai, 2024; Willard and Louf, 2023) ensure compliance by strictly enforcing token transitions. However, this rigid intervention does not consider the model's internal reasoning flow, resulting in abrupt cut-offs or incoherent outputs. This trade-off highlights the critical need for a framework that seamlessly decouples format constraints from task solving to unlock the full potential of LLMs.

In this paper, we introduce DECO-G, a framework that explicitly decouples format adherence from task reasoning, allowing LLMs to focus solely on problem-solving while introducing an auxiliary module to guarantee format adherence. Specifically, it leverages the modularity of existing controllable text generation methods (e.g. GeLaTo (Zhang et al., 2023), Ctrl-G (Zhang et al., 2024)) and delegates format adherence to a dedicated Format Estimation Module (FEM) that continuously estimates the final format compliance and reweighs each next-token distribution to guide generation.

While this work builds upon GeLaTo (Zhang et al., 2023) and Ctrl-G (Zhang et al., 2024), it is, to our knowledge, the first framework that introduces explicit separation of task solving and format adherence to preserve LLMs' full potential. Moreover, naively applying existing methods leads to degraded performance, due to their limited compatibility with instruction-tuned LLMs and computational bottlenecks when handling complex templates. To this end, we introduce three innovations: (1) instruction-aware distillation to capture task-

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 16764–16781

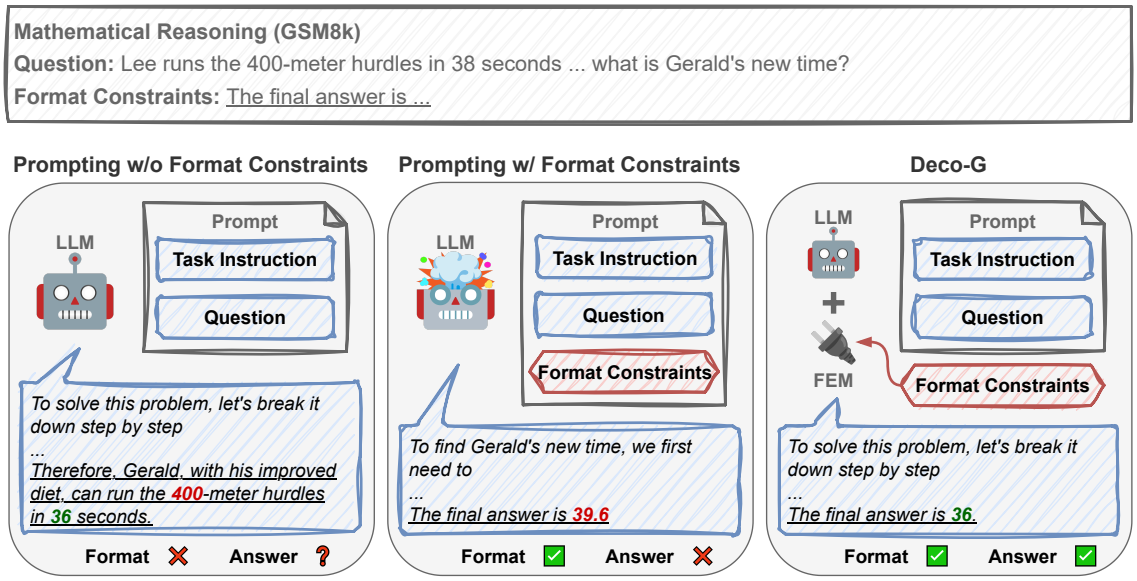


Figure 1: GSM8k examples. Unconstrained prompting yields correct answers but fails formatting, while adding constraints degrades reasoning accuracy. DECO-G decouples formatting via a Format Estimation Module (FEM), achieving both strict compliance and correct answer.

oriented behaviors; (2) a flexible trie algorithm for efficient automata construction; and (3) HMM state pruning to accelerate inference.

To assess DECO-G's effectiveness across diverse constraint types, we evaluate it on three distinct tasks: mathematical reasoning, event argument extraction, and LLM-as-a-judge. Experiments demonstrate that DECO-G consistently improves overall task performance, driven by: (1) guaranteeing 100% format compliance; (2) enabling more natural, context-aware format integration; and (3) freeing the LLM to concentrate on reasoning without the cognitive burden of formatting.

Our contributions are as follows:

- We propose a novel decoding framework that uses a Format Estimation Module to explicitly decouple format adherence from task solving, preserving the LLM's reasoning capabilities while enforcing strict constraints.
- We introduce three technical innovations—instruction-aware distillation, flexible trie construction, and HMM state pruning—that induce minimal computational overhead and render the DECO-G framework practical for real-world deployment.
- We demonstrate consistent performance gains across diverse benchmarks, supported by a detailed analysis of the steering mechanism and entropy-based control dynamics.

2 Preliminaries

This section reviews prior work on controllable generation that our approach builds upon, and highlights how these methods, while instrumental, struggle to achieve our goal of efficient task–format disentanglement. Our formulation and technical contributions are introduced in the next section.

2.1 Generation with Attribute Control

We follow prior work to formulate controllable text generation given a desired attribute α as

$$P(x_{1:n}|\alpha) = \prod_t P(x_t|x_{<t}, \alpha),$$

where the objective is to optimize the probability of sequences that exhibits the attribute α (e.g., contain certain keywords, sentiment, etc.). At each generation step t, the target distribution for producing text with the desired attribute is $P(x_t|x_{<t}, \alpha)$. Using Bayes' rule, we can rewrite this as:

$$P(x_t|x_{<t}, \alpha) = P_{\text{LM}}(x_t|x_{<t}) \frac{P_{\text{LM}}(\alpha|x_t, x_{<t})}{P_{\text{LM}}(\alpha|x_{<t})} \quad (1)$$

The first term $P_{\text{LM}}(x_t|x_{<t})$ is the language model's next-token probability. The second term $\frac{P_{\text{LM}}(\alpha|x_t,x_{<t})}{P_{\text{LM}}(\alpha|x_{<t})}$ acts as a control signal. It quantifies how the choice of the current token x_t influences the probability that the complete sequence will satisfy attribute α. However, directly calculating this ratio is intractable, as it requires marginalizing over

all possible future sequences. A key challenge in controllable generation is to find a tractable approximation for this term.

2.2 Estimating Attribute Likelihood

Recent controllable generation frameworks such as GeLaTo (Zhang et al., 2023) and Ctrl-G (Zhang et al., 2024) leverage a tractable probabilistic model (TPM) to efficiently estimate the marginal probability $P(\alpha|x_t, x_{<t})$, serving as a signal to steer an LLM's generation, following

$$P(x_t|x_{<t}, \alpha) \propto P_{\text{LM}}(x_t|x_{<t})P_{\text{TPM}}(\alpha|x_t, x_{<t}) \tag{2}$$

These approaches first distill a Hidden Markov Model (HMM) as a probabilistic approximation of the LLM and then encode logical constraints to formal structure.

Sequence Modeling. HMM is the specific type of TPM used in these frameworks, chosen for its ability to model sequential data tractably. The joint probability distribution over a sequence of observed variables (tokens, $x_{1:n}$) and a corresponding sequence of hidden state variables $z_{1:n}$, is modeled as

$$P_{\text{HMM}}(x_{\leq t}, z_{\leq t}) = P(z_1)P(x_1|z_1)\prod_{t=2}^{T} P(z_t|z_{t-1})P(x_t|z_t)$$

HMM's Markov property enables efficient probabilistic inference over future sequences, a task that is intractable for language models. In GeLaTo and Ctrl-G, the HMM is distilled from the LLM using samples drawn unconditionally from the LLM.

Formalizing Constraints. To enforce a constraint using the HMM, the constraints must be expressed in a formal language. Zhang et al. (2024) propose representing logical constraints as Deterministic Finite Automata (DFA). A DFA is an abstract state machine that recognizes patterns in sequences. Formally, a DFA is a 5-tuple $\mathcal{D} = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is the alphabet (the LLM's token vocabulary), $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of accept states. A sequence is "accepted" if it drives the machine from its initial state to an accept state. This formalism is capable of representing logical constraints including the presence of keyphrases and word counts.

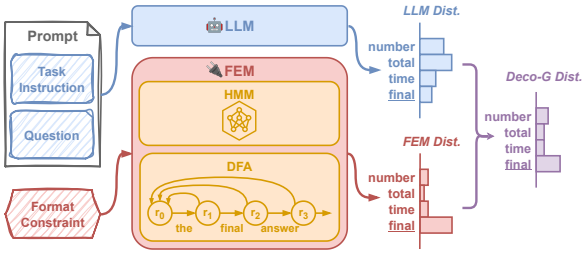


Figure 2: DECO-G decouples task and format by prompting LLM with task-only information and sending format constraints to FEM. DECO-G decodes from the posterior constructed by reweighing LLM token probabilities with the FEM estimated satisfaction rate.

Probabilistic reasoning over logical constraints. The core idea of these prior frameworks is to use the TPM to perform a *probabilistic lookahead* — i.e., to efficiently compute $P_{\text{TPM}}(\alpha|x_t, x_{<t})$, the probability that the full generated sequence will satisfy the constraint α. This is accomplished by marginalizing the joint HMM-DFA state space over all possible future sequences that reach an accepting state in the DFA. According to Zhang et al. (2024), this marginalization can be calculated efficiently using a backward recurrence relation. Refer to Section A for detailed derivation.

2.3 From Prior Work to DECO-G

While GeLaTo (Zhang et al., 2023) and Ctrl-G (Zhang et al., 2024) demonstrate the effectiveness of generic TPMs for logical constraints (e.g., keyphrases), they face significant limitations when adapted for task-format decoupling in instruction-tuned LLMs:

- **Domain Shift:** Prior methods distill HMMs from unconditional random generations. This fails to capture the *instruction-conditional* distribution required for tasks.
- **Template Complexity:** Real-world formats often interweave fixed text with variable-length fields. Representing these structures via standard DFA construction leads to state-space explosion, creating a computational bottleneck.
- **Inference Latency:** The computational cost of probabilistic lookahead scales poorly with the massive vocabulary size. Without optimization, this brings significant latency for real-time generation.

3 DECO-G

In this section, we present DECO-G, a framework that realizes the decoupling of task reasoning from output formatting. As shown in Figure 2, our

method separates the input prompt: the LLM receives only the task-specific information, while a dedicated Format Estimation Module (FEM) receives the format constraints. At each decoding step, the FEM estimates the likelihood of future compliance with the given format constraints α. This likelihood is then used to reweigh the LLM's original token probabilities, steering the generation towards a format-compliant output. Below, we describe the key components that enable this framework.

3.1 Instruction-Aware HMM Distillation

An HMM can approximate a large language model's (LLM) output distribution to guide controllable generation. The fidelity of this approximation is critical—ideally, an HMM that perfectly replicates the LLM's probabilities would yield an exact posterior for format-decoupled generation, per Equation (1). Our key insight is that for instruction-tuned LLMs, the output distribution is fundamentally different when conditioned on a prompt versus when generating text unconditionally. Prior methods (Zhang et al., 2023, 2024), however, distill their HMMs using text sampled unconditionally from a model, an approach that fails to capture the task-oriented behavior that emerges after instruction fine-tuning. This design renders methods like Ctrl-G ineffective, leading to the suboptimal control patterns we demonstrate in Section B.

To bridge this gap, we carry out instruction-aware distillation: conditioning an HMM on task-oriented behavior by training it exclusively on the LLM's instruction-response pairs. Specifically, we distill knowledge from over *one million* completions generated by an LLM prompted with *one thousand* unique instructions from the Natural-Instructions-v2 (Mishra et al., 2022) dataset. Following Zhang et al. (2023), we train the HMM using the Baum-Welch algorithm (Baum, 1972). This process yields a robust HMM that models the LLM's conditional, instruction-following behavior, enabling more precise control over generation across a wide spectrum of tasks.

3.2 Flexible Trie Building for Complex Constraints

To efficiently enforce complex formatting requirements, we formalize the constraint space as a *Template Language* $\mathcal{L}_\mathcal{T}$. Unlike standard trie constructions which branch on individual characters, our flexible trie algorithm constructs a Deterministic Finite Automaton (DFA) by branching on *segments*, comprised of fixed Pivots and variable Wildcards.

Template Segments. Let $\mathcal{V}$ be the vocabulary of the LLM. A template $T \in \mathcal{T}$ is defined as an ordered sequence of segments $S = (s_1, s_2, \ldots, s_n)$, where each segment s_i is either:

- **Pivot (P):** A deterministic sequence of tokens $x_{1:m} \in \mathcal{V}^m$ representing static text (e.g., formatting cues like "`The answer is:`"). The language accepted by a pivot is the singleton set:
$$L(P) = \{x_{1:m}\}$$
- **Wildcard (W):** A variable slot constrained only by length $[\ell_{min}, \ell_{max}]$. It accepts any token sequence x where the length satisfies the bounds. Its language is the union of all vocabulary permutations within the length range:
$$L(W) = \bigcup_{k=\ell_{min}}^{\ell_{max}} \mathcal{V}^k$$

The language recognized by a full template T is the concatenation of its segment languages: $L(T) = L(s_1) \cdot L(s_2) \cdots L(s_n)$.

Trie Algorithm. We construct a minimal DFA $\mathcal{A} = (Q, \Sigma, \delta, q_{init}, F)$ that recognizes the union of all allowed templates. The core innovation is our State Convergence strategy, which prevents state-space explosion for multi-part templates. Let $q_{init}^{(i)}$ be the unique entry state for segment s_i. We construct the transition function such that for any valid string realization w of the current segment s_i, the automaton transitions to the unique entry state of the next segment s_{i+1}:

$$\forall w \in L(s_i), \quad \delta^*(q_{init}^{(i)}, w) = q_{init}^{(i+1)} \qquad (3)$$

where $\delta^* : Q \times \mathcal{V}^* \to Q$ is the extended transition function processing a string of tokens. Equation 3 mathematically guarantees that the state configuration for segment s_{i+1} is strictly independent of the path taken through s_i.

Complexity. Standard DFA construction treats every valid length permutation of a wildcard as a distinct path. For a template with N sequential wildcards of length variance K, this dependency chains forward, resulting in exponential complexity $O(K^N)$. Consider the event argument extraction template alternating between fixed pivots P and variable wildcards W:

$$T = P_1(\text{“}\texttt{Role A:}\text{”}) \cdot W_1 \cdot P_2(\text{“}\texttt{Role B:}\text{”}) \cdot W_2 \ldots$$

In standard construction, the state history of W_2 depends on the specific length chosen for W_1. In contrast, our algorithm enforces the convergence property (Equation 3): all valid paths for W_1 merge back to the single canonical node $q_{init}^{(P_2)}$. This decouples the wildcard histories, ensuring the total state space grows linearly as $O(N \times K)$.

3.3 Estimating Format Compliance

With a distilled HMM that simulates LLM distribution and a DFA that encodes format constraints α, we calculate the marginal probability over all sequences accepted by $\mathcal{D}(\alpha)$ as

$$P_{\text{FEM}}(\alpha|x_t, x_{<t}) = \frac{P(\mathcal{D}(\alpha) = 1, x_t, x_{<t})}{P(x_t, x_{<t})}$$

While the joint probability of format compliance and context sequence $P(\mathcal{D}(\alpha) = 1, x_t, x_{<t})$ is not readily available in the FEM, we follow Zhang et al. (2024)'s marginalization of HMM over DFA (Section A) to calculate this value. Finally, we use the FEM estimated compliance rate as likelihood to construct the DECO-G posterior:

$$P(x_t|x_{<t}, \alpha) \propto P_{\text{LM}}(x_t|x_{<t})[P_{\text{FEM}}(\alpha|x_{<t}, x_t)]^{\gamma}$$

The resulting scores are re-normalized over the vocabulary $\mathcal{V}$ (via softmax) to ensure a valid probability distribution. γ is a hyperparameter that controls the steering strength with a default value of 1.

3.4 HMM Hidden State Pruning

The Format Estimation Module (FEM) provides effective guidance but introduces a computational bottleneck during inference. The primary cost lies in the HMM's emission stage, which calculates the probability distribution over the entire vocabulary $\mathcal{V}$ from h hidden states. With $h = 4096$ and $|\mathcal{V}| > 100k$ for *Llama* and *Qwen*, this matrix-vector multiplication ($O(h|\mathcal{V}|)$) significantly impedes latency.

To mitigate this, we introduce **HMM hidden state pruning**. This technique exploits the observation that the probability mass of the hidden state distribution is highly concentrated within a small subset of states (see Figure 3). Instead of using the full state space, we approximate the emission probabilities by considering only the top-k most probable states. Empirically, selecting just the top 5% ($k = 200$) is sufficient to retain over 98% of the full model's performance. This strategy drastically improves efficiency by reducing the emission complexity from $O(h|\mathcal{V}|)$ to $O(k|\mathcal{V}| + h \log h)$, where the $O(h \log h)$ term accounts for selecting the top states. With $k \ll h$, this optimization achieves a $\sim 13\times$ reduction in FLOPs per decoding step, ensuring that the guidance overhead remains negligible while maintaining robust format compliance.

4 Experiment

Experimental Setup. We assess DECO-G's overall performance over three tasks: (1) math problem solving with reasoning, (2) event argument extraction as a generative task, and (3) LLM-as-a-judge for summary evaluation (see Section 4.3). We apply DECO-G on performant instruction models *Llama-3.1-8B-Instruct* (Grattafiori et al., 2024), *Qwen2.5-7B-Instruct* (Yang et al., 2025b), and *Qwen3-8B* (Yang et al., 2025a) to verify its effectiveness. The baselines we include for comparison are as follows:

- **Prompt-Only (NL/JSON):** Standard greedy decoding conditioned on task instructions and output constraints (natural language or JSON), without external interference.
- **Structured Generation (NL/JSON):** Constrained decoding using *Outlines* (Willard and Louf, 2023) to strictly enforce the target output format, either via a natural language template or a JSON schema.
- **Ctrl-G:** A controllable generation baseline that uses an HMM distilled from unconditional sampling to guide generation, as detailed in Zhang et al. (2024).

For the following experiments, we adopt greedy decoding to ensure fair comparison with baseline methods and evaluate zero-shot performance.

4.1 Mathematical Reasoning

In this task, we evaluate our framework on GSM8k (Cobbe et al., 2021), a collection of grade school math problems that take two to eight steps to solve. Models are expected to carry out step-by-step reasoning and arrive at the answer. Following Tam et al. (2024), a group of task instructions is adopted to prompt the model to first reason about the math problem and then yield an integer as its answer. For JSON format output, we prompt the model to output valid JSON blob with keys *"reason"* and *"answer."* For natural language output, format instructions are used to encourage the model to generate the template phrase *"The final answer is ..."* Meanwhile, this phrase is specified as a key phrase to appear in DECO-G's generation.

Model	Method	Format (%)	Acc. (%)
Llama-3.1-8B -Instruct	Prompt-Only (NL)	96.3	82.3
	Prompt-Only (JSON)	64.7	51.8
	Outlines (NL)	100	81.3
	Outlines (JSON)	100	75.7
	Ctrl-G	100	61.6
	DECO-G	100	**85.2**
Qwen2.5-7B -Instruct	Prompt-Only (NL)	98.0	83.6
	Prompt-Only (JSON)	93.3	74.8
	Outlines (NL)	99.9	82.7
	Outlines (JSON)	99.8	79.0
	Ctrl-G	100	84.9
	DECO-G$^{\gamma=2}$	100	**88.6**
Qwen3-8B	Prompt-Only (NL)	97.4	90.5
	Prompt-Only (JSON)	66.9	61.4
	Outlines (NL)	100	88.3
	Outlines (JSON)	99.2	90.6
	Ctrl-G	100	88.7
	DECO-G$^{\gamma=2}$	100	**91.7**

Table 1: GSM8k Results. DECO-G guarantees format satisfaction and consistently outperforms Prompt-Only and Outlines baselines across all models.

Evaluation Metrics. We measure *Format Compliance* as the rate to which the generated answer follows format requirement. In addition, we measure *Accuracy* as exact match of ground truth.

Results. As shown in Table 1, *Prompt-Only (NL)* offers decent performance, with *Llama* scoring 82.3%, *Qwen2.5* 83.6%, and *Qwen3* 90.5% on accuracy. However, unstructured generation methods completely rely on the LLM for following format constraints and thus suffer from low compliance rate. Structured generation (*Outlines*), on the contrary, guarantees format compliance, but its invasive intervention compromises task performance. While both Ctrl-G and DECO-G guarantee format compliance, DECO-G benefits from more accurate control signal provided by the instruction-aware HMM and achieves the best performance across all three models. In practice, we observe that *Qwen* models have more skewed token distribution. We thus raise the control factor γ to exert stronger control on the output.

4.2 Event Argument Extraction

The generative event argument extraction (EAE) task assesses a model's ability in identifying role-related arguments from source text. We evaluate on the ACE05-EN dataset (Doddington et al., 2004), in which a model is presented with an article, a trigger word, and a set of roles to determine whether arguments associated with the roles are present in the article. This is naturally a templated task as generative models have to specify which word is extracted for which role. Regarding JSON output, we ask model to generate a JSON blob with roles as keys and extracted arguments as values. For natural language output, we specify the template *"The <role$_i$> is ..."* for every relevant role. For DECO-G, we construct a flexible DFA that fuses the template phrases together with empty slots allowing LLM predict arguments spanning from 1 to 5 tokens.

Evaluation Metrics. We measure performance by calculating the *f1-score* comparing the extracted tuples and the ground truth tuples for the following categories: Argument Id (AI), Argument Class (AC), Argument-attached Id (AI+), Argument-attached Class (AC+).

Results. The *f1-scores* reported in Table 2 suggest that EAE remains a challenging task for generative models. LLMs suffer from identifying correct relations in the article and presenting valid predictions that exist in the original text. Baseline methods show inconsistent trends across models, indicating LLMs' lack of robustness in event argument extraction. Employing DECO-G enhances overall extraction quality for *Llama* and *Qwen3*, while mainly improving over AI and AI+ for *Qwen2.5*. DECO-G's gain on AI and AI+ is more evident than its improvement on AC and AC+, suggesting that DECO-G can further benefit from a tighter association between roles and extracted arguments.

Model	Method	AI	AC	AI+	AC+
Llama-3.1-8B -Instruct	Prompt-Only (NL)	36.8	27.3	34.8	25.5
	Prompt-Only (JSON)	35.2	26.2	33.4	24.6
	Outlines (NL)	37.1	27.8	35.1	26.0
	Outlines (JSON)	33.6	25.2	31.6	23.6
	Ctrl-G	39.1	27.5	**37.0**	26.1
	DECO-G	**39.4**	**28.7**	**37.0**	**26.8**
Qwen2.5-7B -Instruct	Prompt-Only (NL)	32.6	25.5	31.2	24.4
	Prompt-Only (JSON)	31.9	24.1	30.5	22.9
	Outlines (NL)	33.2	24.9	31.1	23.7
	Outlines (JSON)	34.1	**26.1**	32.5	**24.7**
	Ctrl-G	34.7	24.5	33.1	23.9
	DECO-G$^{\gamma=2}$	**35.2**	25.9	**33.4**	24.5
Qwen3-8B	Prompt-Only (NL)	33.2	**24.6**	31.5	**23.1**
	Prompt-Only (JSON)	31.7	23.4	30.1	21.8
	Outlines (NL)	33.3	24.2	31.5	22.6
	Outlines (JSON)	32.5	23.0	30.8	21.5
	Ctrl-G	33.7	23.3	31.6	22.4
	DECO-G$^{\gamma=2}$	**34.0**	**24.6**	**32.1**	**23.1**

Table 2: Generative EAE results on ACE05.

Model	Method	Coherence		Consistency		Fluency		Relevance		Avg		
		ρ	τ	ρ	τ	ρ	τ	ρ	τ	Format	ρ	τ
Llama-3.1-8B -Instruct	Prompt-Only (NL)	0.381	0.311	0.383	0.351	0.321	0.291	0.405	0.337	95.8	0.372	0.322
	Prompt-Only (JSON)	0.449	0.368	0.446	0.415	0.326	0.296	0.424	0.358	99.8	0.411	0.359
	Outlines (NL)	0.376	0.308	0.375	0.343	0.316	0.287	0.435	0.364	100	0.376	0.325
	Outlines (JSON)	0.450	0.369	**0.447**	**0.416**	**0.334**	**0.302**	0.424	0.358	100	0.414	0.361
	Ctrl-G	0.449	0.368	0.440	0.403	0.327	0.296	0.437	0.369	100	0.413	0.359
	DECO-G	**0.458**	**0.379**	0.439	0.404	0.331	0.298	**0.441**	**0.371**	100	**0.418**	**0.363**
Qwen2.5-7B -Instruct	Prompt-Only (NL)	0.407	0.339	0.442	0.407	0.291	0.265	0.399	0.340	100	0.385	0.338
	Prompt-Only (JSON)	0.411	0.334	0.488	0.455	0.305	0.280	0.383	0.326	99.7	0.396	0.349
	Outlines (NL)	0.403	0.337	0.448	0.412	0.279	0.254	0.408	0.347	100	0.384	0.338
	Outlines (JSON)	**0.412**	**0.335**	0.489	0.457	0.309	0.284	0.387	0.330	100	0.399	0.351
	Ctrl-G	0.347	0.286	0.481	0.450	0.325	0.293	0.441	0.372	100	0.399	0.350
	DECO-G$^{\gamma=2}$	0.327	0.271	**0.506**	**0.470**	**0.348**	**0.311**	**0.452**	**0.380**	100	**0.408**	**0.358**
Qwen3-8B	Prompt-Only (NL)	**0.510**	**0.416**	0.540	0.504	0.479	0.441	0.464	0.392	100	0.498	0.439
	Prompt-Only (JSON)	0.504	0.409	0.491	0.459	0.444	0.410	0.450	0.382	99.8	0.472	0.415
	Outlines (NL)	0.507	0.413	0.544	0.509	0.477	0.439	0.468	0.396	100	0.499	0.439
	Outlines (JSON)	0.486	0.393	0.486	0.454	0.406	0.376	0.441	0.374	100	0.455	0.399
	Ctrl-G	0.484	0.391	**0.550**	**0.519**	0.476	0.436	0.491	0.412	100	0.500	0.440
	DECO-G$^{\gamma=2}$	0.490	0.395	0.546	0.516	**0.499**	**0.456**	**0.494**	**0.414**	100	**0.507**	**0.445**

Table 3: SummEval results, measured over Coherence, Consistency, Fluency, and Relevance.

4.3 LLM-as-a-judge Evaluation

We then use LLMs as judges to evaluate the quality of summaries and assess how well it aligns with human annotation. This evaluation is performed on the SummEval (Fabbri et al., 2021) which consists 1600 machine-generated summaries for 100 news articles and human annotated scores over four dimensions: Coherence, Consistency, Fluency, and Relevance. The models are asked to analyze the summary and assign a score from 1 to 5 based on the given criteria suggested by ChatGPT (OpenAI, 2025). We use the format "*The rating is ...*" for natural language output and "*rating*" as the key for harnessing JSON output.

Evaluation Metrics. Following Liu et al. (2023), we adopt the summary-level Spearman and Kendall-Tau correlation to gauge the performance of each method. Higher number indicates better alignment with human annotated scores.

Results. For this task, *Qwen* models perform well in following the output format in unstructured settings, securing over 99.7% compliance rate. This may attribute to a less intensive reasoning phase compared to mathematical reasoning. As indicated in Table 3, DECO-G demonstrates strongest average correlation with human annotator, enhancing over Consistency, Fluency, and Relevance when applied to *Qwen* models. A qualitative inspection reveals that DECO-G enables the flexible integration of scoring phrases at arbitrary positions, whereas unstructured and structured baselines consistently append ratings only after the analysis concludes.

5 Analysis

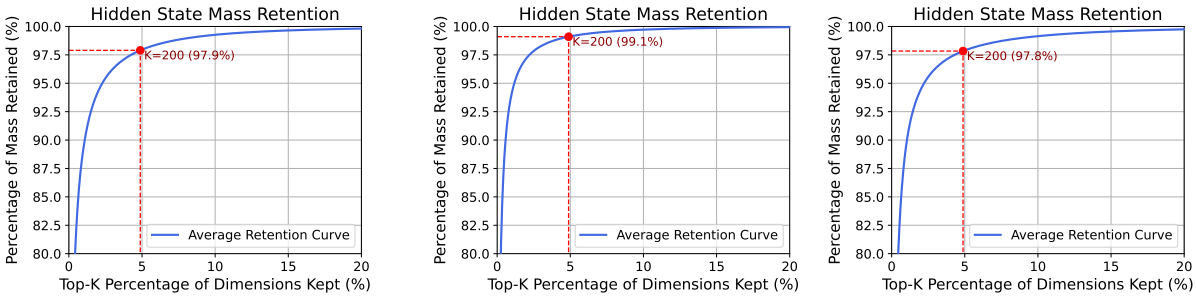
5.1 Complexity and Efficiency

We analyze the efficiency of DECO-G from both theoretical complexity and empirical latency perspectives, highlighting improvements brought by our HMM pruning (Section 3.4) and flexible trie construction (Section 3.2).

Method	Acc (%)
Llama-3.1-8B-Instruct	
DECO-G w/o Pruning	85.4
DECO-G	85.2 (Δ=-0.2)
Qwen2.5-7B-Instruct	
DECO-G w/o Pruning	89.8
DECO-G	88.6 (Δ=-1.2)
Qwen3-8B	
DECO-G w/o Pruning	90.8
DECO-G	91.7 (Δ=+0.9)

Table 4: DECO-G performance on GSM8k with and without pruning.

Hidden State Sparsity. Our pruning strategy is empirically justified by the highly concentrated nature of the hidden state distributions (Figure 3). For *Llama*- and *Qwen*-distilled HMMs, the top 5% of hidden states ($k = 200$) capture over 97.8% of



(a) *Llama-3.1-8B-Instruct* (b) *Qwen2.5-7B-Instruct* (c) *Qwen3-8B*

Figure 3: Average retention rate (of total mass) over top-k HMM hidden states on GSM8k dataset.

the total probability mass on average. Crucially, this approximation preserves task performance. As shown in Table 4, the accuracy degradation on GSM8k is minimal when pruning is applied (e.g., -0.2% for *Llama*, +0.9% for *Qwen3*). This confirms that the pruned HMM retains sufficient guidance information to steer generation effectively.

Computational Complexity. The primary bottleneck in HMM-guided generation is the emission probability calculation, which typically scales with $O(h|\mathcal{V}|)$. By exploiting the sparsity established above, we reduce this complexity to $O(k|\mathcal{V}|)$ where $k \ll h$. With $k = 200$, this yields a $\sim 13\times$ reduction in guidance FLOPs (from 1.08 down to 0.08 GFLOPs for an 8B model). Consequently, the total computational cost of the FEM constitutes only $\sim 0.53\%$ of the LLM's main forward pass (Kaplan et al., 2020), rendering the overhead negligible.

Empirical Latency. We measure the wall-clock latency of DECO-G against standard unconstrained generation, structured generation baselines (Outlines), and prior HMM-based control (Ctrl-G). As shown in Table 5, DECO-G incurs minimal latency overhead while guaranteeing format compliance.

Method	Latency *(ms/token)* ↓	Overhead *(latency)* ↓	Format	Task Perf.
Prompt-Only	3.07	1.00×		
Structured (Outlines)	8.09	2.64×	✓	
Ctrl-G	10.70	3.49×	✓	
DECO-G *(Full HMM)*	9.45	3.08×	✓	✓
DECO-G	**5.12**	**1.67**×	✓	✓

Table 5: Inference latency comparison on GSM8k with single NVIDIA A100.

Scalability with Constraint Complexity. Beyond raw speed, DECO-G demonstrates superior scalability when handling complex constraints compared to baselines:

- **vs. Structured Generation:** Frameworks like Outlines (Willard and Louf, 2023) and Guidance (guidance-ai, 2024) rely on intersecting FSMs with the LLM vocabulary. This becomes computationally prohibitive for "floating" constraints at *arbitrary positions* (e.g., after free-form reasoning), as modeling such unconstrained spans over long sequences triggers state-space explosion in the automaton. In contrast, DECO-G naturally handles these "floating" constraints, allowing the LLM to generate free-form text until the probabilistic guidance steers it into the target format, without requiring expensive pre-computation of all possible prefix permutations.
- **vs. Ctrl-G:** Ctrl-G (Zhang et al., 2024) fails on templates with chained variable-length segments, such as EAE extraction slots (e.g., "`Role A: [span] Role B: [span]`"). Standard DFA construction creates unique paths for every length permutation, causing exponential state explosion ($O(K^N)$ for N slots). Our *Flexible Trie* (Section 3.2) avoids this by merging wildcard paths at each fixed pivot, achieving linear complexity ($O(N \times K)$).

5.2 The Steering Process

DECO-G takes advantage of HMM to estimate the future format satisfaction rate and adjust token probabilities based on the estimation. To better understand this steering process, we examine the control signals produced by the FEM and visualize the control for a span of decoding step. We track the original LLM distribution, FEM distribution, and their composed distribution, which DECO-G decodes from. Figure 5 provides an illustration of DECO-G encouraging the generation of key phrase after step by step reasoning. While the LLM tends to conclude its response with "*The total number*

of ... is ..." DECO-G assigns high probabilities to the token "*final*," steering the LLM generation to conform with format constraints.

As LLMs are trained to provide clear and concise response, they tend to avoid repeating themselves when presenting the final answer. DECO-G captures this intricacy and replaces LLM's intended conclusive phrase with the format phrase "*The final answer is*" to reduce repetition. We consider this format integration to be more natural than forcing LLM to generate certain phrases as in regex-structured generation.

6 Related work

In the paper, we explore a controllable text generation (CTG) method to decouple task solving from format adherence. There are two branches in CTG that provide avenues for achieving this format-task decoupling—content-wise hard control and attribute-wise soft control. **Content-wise methods** (Willard and Louf, 2023; guidance-ai, 2024) enforce strict adherence to predefined templates, ensuring reliability but often exerting invasive control that leads to abrupt, incoherent cutoffs. **Attribute-wise soft control** offers greater flexibility, either by updating model weights—via retraining (Keskar et al., 2019; Arora et al., 2022), fine-tuning (Wei et al., 2021; Zeldes et al., 2020; Li and Liang, 2021; Lester et al., 2021), or reinforcement learning (Ouyang et al., 2022; Stiennon et al., 2020; Zeng et al., 2024; Dai et al., 2024)—or through *weighted decoding* (Dathathri et al., 2019; Yang and Klein, 2021; Krause et al., 2021; Schick et al., 2021; Liu et al., 2021; Khandelwal et al., 2021; Sitdikov et al., 2022; Wen et al., 2023; Deng and Raffel, 2023; Loula et al., 2025; Lipkin et al., 2025). The latter steers generation at inference time using lightweight auxiliary models (Bayes' rule) without the high cost of retraining. DECO-G adopts this weighted decoding paradigm to compute the posterior probability of format constraints as a decoupled attribute.

7 Conclusion

We present DECO-G, a novel decoding framework designed to decouple the responsibilities of task reasoning and format adherence. It achieves this responsibility separation by employing an auxiliary Format Estimation Module to estimate future format satisfaction and modify token probabilities, thus allowing the LLM to concentrate solely on problem-solving. Experiments on mathematical reasoning, event argument extraction, and LLM-as-a-judge evaluation demonstrate this decoupling approach leads to overall performance gain, attributed to improved format compliance and more natural format integration in the response.

Limitations

While we show DECO-G enhances LLM task performance in various tasks, a few limitations should be taken into consideration when using DECO-G. Firstly, the HMM used to estimate format satisfaction rate is specific to an LLM, meaning that one has to distill a new HMM when switching to a different LLM. Although, in practice, we find that an HMM can be applied to larger LLMs in the same family, it is not an accurate representation of their token distributions. Secondly, similar to other CTG methods that includes additional module for attribute modeling, DECO-G introduces additional computation overhead during decoding. As suggested by (Zhang et al., 2024), the probabilistic traversal of future generation courses using HMM has a complexity that is linear to the number of edges in DFA and quadratic to the number of hidden states in HMM. Complex format constraints, converted to larger DFA, are thus likely to increase generation runtime. Finally, finding the optimal hyperparameter γ for LLMs with highly peaked token distributions may require empirical explorations. Such distributions require increased γ to ensure robust format compliance, yet an excessive value may adversely affect the output quality.

Ethical considerations

We conduct experiments on mathematical reasoning, event argument extraction, and LLM-as-a-judge evaluation. The score assigned by an LLM should not be considered an accurate reflection of quality of the summary. In addition, the LLM responses to the GSM8k questions should not be referenced for math instruction as they may include hallucination.

We acknowledge the use of AI assistants for improving the manuscript's prose, generating tables in LaTeX format, figure design, and assisting with code implementation for the analysis of HMM hidden state pruning. All generated content, particularly the data in tables, was manually verified for accuracy against our experimental results.

References

Kushal Arora, Kurt Shuster, Sainbayar Sukhbaatar, and Jason Weston. 2022. Director: Generator-classifiers for supervised language modeling. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 512–526.

Leonard E Baum. 1972. An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3(1):1–8.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. 2024. Guiding LLMs the right way: Fast, non-invasive constrained generation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 3658–3673. PMLR.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. 2024. Safe rlhf: Safe reinforcement learning from human feedback. In *The Twelfth International Conference on Learning Representations*.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*.

Haikang Deng and Colin Raffel. 2023. Reward-augmented decoding: Efficient controlled text generation with a unidirectional reward model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11781–11791.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Lrec*, volume 2, pages 837–840. Lisbon.

Alexander R Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. Summeval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

guidance-ai. 2024. Guidance: A guidance language for controlling large language models. `https://github.com/guidance-ai/guidance`.

Jia He, Mukund Rungta, David Koleczek, Arshdeep Sekhon, Franklin X Wang, and Sadid Hasan. 2024. Does prompt formatting have any impact on llm performance? *Preprint*, arXiv:2411.10541.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. Gedi: Generative discriminator guided sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4929–4952.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

Benjamin Lipkin, Benjamin LeBrun, Jacob Hoover Vigly, João Loula, David R. MacIver, Li Du, Jason Eisner, Ryan Cotterell, Vikash Mansinghka, Timothy J. O'Donnell, Alexander K. Lew, and Tim Vieira. 2025. Fast controlled generation from language models with adaptive weighted rejection sampling. *Preprint*, arXiv:2504.05410.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A Smith, and Yejin Choi. 2021. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*.

Do Xuan Long, Ngoc-Hai Nguyen, Tiviatis Sim, Hieu Dao, Shafiq Joty, Kenji Kawaguchi, Nancy F. Chen, and Min-Yen Kan. 2025. LLMs are biased towards output formats! systematically evaluating and mitigating output format bias of LLMs. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 299–330, Albuquerque, New Mexico. Association for Computational Linguistics.

João Loula, Benjamin LeBrun, Li Du, Ben Lipkin, Clemente Pasti, Gabriel Grand, Tianyu Liu, Yahya Emara, Marjorie Freedman, Jason Eisner, Ryan Cotterell, Vikash Mansinghka, Alexander K. Lew, Tim Vieira, and Timothy J. O'Donnell. 2025. Syntactic and semantic control of large language models via sequential monte carlo. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *ACL*.

OpenAI. 2025. ChatGPT. Large language model. Accessed May 19, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll 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 F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Yi Ren and Danica J. Sutherland. 2025. Learning dynamics of llm finetuning. *Preprint*, arXiv:2407.10490.

Timo Schick, Sahana Udupa, and Hinrich Schütze. 2021. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in nlp. *Transactions of the Association for Computational Linguistics*, 9:1408–1424.

Askhat Sitdikov, Nikita Balagansky, Daniil Gavrilov, and Alexander Markov. 2022. Classifiers are better experts for controllable text generation. *arXiv preprint arXiv:2205.07276*.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung yi Lee, and Yun-Nung Chen. 2024. Let me speak freely? a study on the impact of format restrictions on performance of large language models. *Preprint*, arXiv:2408.02442.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Zhihua Wen, Zhiliang Tian, Zhen Huang, Yuxin Yang, Zexin Jian, Changjian Wang, and Dongsheng Li. 2023. Grace: gradient-guided controllable retrieval for augmenting attribute-based text generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8377–8398.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Kevin Yang and Dan Klein. 2021. Fudge: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3511–3535.

Qwen: An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan

Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2025b. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Yoel Zeldes, Dan Padnos, Or Sharir, and Barak Peleg. 2020. Technical report: Auxiliary tuning and its application to conditional text generation. *arXiv preprint arXiv:2006.16823*.

Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. 2024. Token-level direct preference optimization. In *Proceedings of the 41st International Conference on Machine Learning*, pages 58348–58365.

Honghua Zhang, Meihua Dang, Nanyun Peng, and Guy Van den Broeck. 2023. Tractable control for autoregressive language generation. *Preprint*, arXiv:2304.07438.

Honghua Zhang, Po-Nien Kung, Masahiro Yoshida, Guy Van den Broeck, and Nanyun Peng. 2024. Adaptable logical control for large language models. *Preprint*, arXiv:2406.13892.

A Probabilistic Reasoning over Logical Constraints

Zhang et al. (2023) and Zhang et al. (2024) use a TPM to perform a probabilistic lookahead—that is, to efficiently compute $P_{\text{TPM}}(\alpha|x_{\leq t})$, the probability that the full generated sequence will satisfy the constraint α. The constraint is encoded as a DFA, and compliance is denoted by the event $\mathcal{D}(\alpha) = 1$. Then, the marginal probability over all sequences accepted by $\mathcal{D}(\alpha)$ is expressed as

$$P_{\text{TPM}}(\alpha|x_{\leq t}) = \frac{P_{\text{TPM}}(\mathcal{D}(\alpha) = 1, x_{\leq t})}{P_{\text{HMM}}(x_{\leq t})}$$

where the numerator—likelihood of satisfying the constraint given a prefix $x_{\leq t}$—is found by marginalizing over the HMM hidden states z_t and the DFA states s_t

$$P_{\text{TPM}}(\mathcal{D}(\alpha) = 1, x_{\leq t}) = \sum_{z_t} P_{\text{TPM}}(\mathcal{D}(\alpha) = 1 \mid z_t, s_t) P_{\text{HMM}}(z_t, x_{\leq t})$$

The conditional compliance probability $P_{\text{TPM}}(\mathcal{D}(\alpha) = 1|z_t, s_t)$, as shown in Zhang et al. (2024), is calculated using a backward recurrence relation. This sums the probabilities of all valid transitions from step t to $t + 1$, weighted by the HMM's transition and emission probabilities

$$P(\mathcal{D}(\alpha) = 1 \mid z_t, s_t) = \sum_{z_{t+1}} P(z_{t+1} \mid z_t) \sum_{s_{t+1}} P(\mathcal{D}(\alpha) = 1 \mid z_{t+1}, s_{t+1}) \sum_{\delta(s_t, x_{t+1}) = s_{t+1}} P(x_{t+1} \mid z_{t+1})$$

The computed probability then serves as the tractable approximation of the format compliance likelihood, which is used to guide the LLM's next-token generation as shown in Equation 2.

B Suboptimal Control from Unconditioned HMM Distillation

When an instruction-tuned model is prompted with no specific user input for unconditional sampling, it often defaults to generic conversational phrases like, "*Is there something I can help you with?*" This behavior is a byproduct of its safety and helpfulness training. An HMM distilled from corpora containing such non-substantive responses learns a token distribution that is unrepresentative of the model's capabilities in actual problem-solving scenarios.

Consequently, when this HMM is applied to a complex reasoning task, it produce suboptimal control signal that can disrupt the reasoning process of the LLM. For example, with *Llama*, when we apply Ctrl-G to the GSM8k dataset, its control mechanism prematurely forces the model to generate the required format phrase ("*The final answer is ...*"), suppressing the step-by-step reasoning necessary to solve the problem. This results in an accuracy of 61.6%, even worse than the Prompt-Only (NL) baseline.

Table 6 provides an example of this failure mode on a GSM8k problem, contrasting Ctrl-G's flawed output with the coherent response from DECO-G, which uses an instruction-aware HMM.

C Token Entropy and Steering Strength

In the experiments, we report DECO-G's results with hyperparameter $\gamma = 2$ for controlling *Qwen* models, as $\gamma = 1$ doesn't provide enough power to steer the model away from its own generation course. We hypothesize that *Qwen* models' token distributions are more skewed than *Llama*'s, making it difficult for the control signal to actually make an impact on the distribution. To verify this, we draw 100 examples from GSM8k responses and measure the average step-wise entropy of LLM token distribution. As shown in Figure 4, *Llama*'s entropy is significantly higher than those of *Qwen2.5* and *Qwen3*, suggesting that *Llama*'s token probabilities are more spread out and diverse, whereas *Qwen* models' token distributions are more peaky. This increased peakiness could be a consequence of the distribution squeezing induced by more intensive fine-tuning and preference optimization of the LLM (Ren and Sutherland, 2025). It is thus intuitive to amplify DECO-G's control strength for LLM with more skewed distribution to guarantee format compliance.

Within the same model, structured generation methods have slightly higher entropy than their unstructured counterparts. This may attribute to imposed template tokens provoking more uncertainty in future token prediction. Meanwhile, DECO-G produces lowest LLM entropy, indicating that an absence of format constraint in task solving may lead to LLM providing the most confident response.

Method	Output
Ctrl-G	The final answer is 0.36 **(INCORRECT)**
DECO-G	To find the probability that both tickets are winners, we need to multiply the probabilities of each ticket winning. 1. The probability of the first ticket winning is 20% or 0.2. 2. The probability of the second ticket winning is three times more likely, so it's $3 \times 0.2 = 0.6$. 3. The probability of both tickets winning is the product of their individual probabilities: $0.2 \times 0.6 = 0.12$. 4. To express this as a percentage, we multiply by 100: $0.12 \times 100 = 12\%$. The final answer is 12. **(CORRECT)**

Table 6: Example of Ctrl-G's failure mode on GSM8k reasoning task.

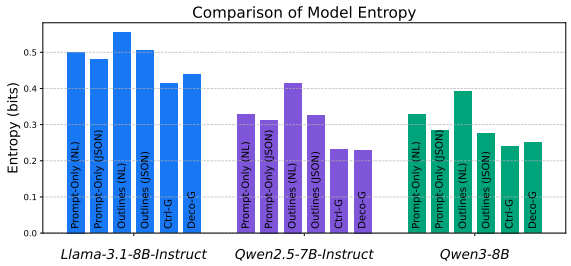


Figure 4: LLM's token-level entropy for different models and methods. *Llama* has a more flexible token distribution as compared to *Qwen*.

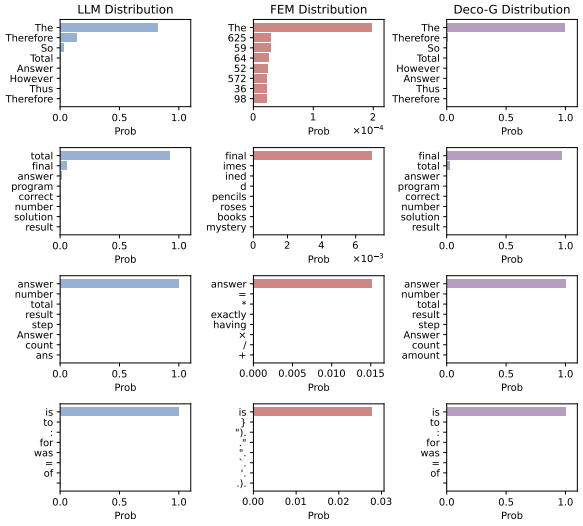


Figure 5: DECO-G steers *Llama* to generate predefined template "The final answer is ..." by boosting probabilities of template tokens.

D HMM Pruning and Efficiency

The primary benefit of HMM hidden state pruning is a substantial improvement in computational efficiency. By reducing the computation of the HMM emission stage, pruning achieves a 13x reduction in the FLOPs required by the HMM forward function at each decoding step (from approx. 1.08 GFLOPs to 0.08 GFLOPs for *Llama*, calculation see below). When compared to the LLM's own forward pass, which requires approximately 16 GFLOPs per token (Kaplan et al., 2020), the pruned FEM's computational cost constitutes only about 0.53% of the main inference workload. This optimization renders the guidance overhead practically insignificant, thereby enhancing the viability of the DECO-G framework.

HMM Forward Computation Cost. Below, we specify the computational cost (in FLOPs) of the HMM's forward pass. The calculation uses the HMM parameters for the *Llama* model: hidden states h=4096, vocabulary size $|\mathcal{V}|$=128k, and top-k states for pruning k=200.

Before Pruning. The total cost is the sum of the state transition cost ($2h^2$) and the emission cost ($2h|\mathcal{V}|$).

$$\begin{aligned}\text{Total FLOPs} &= (2 \times 4096^2) \\ &\quad + (2 \times 4096 \times 128{,}000) \\ &= (3.36 \times 10^7) + (1.05 \times 10^9) \\ &\approx \textbf{1.08 GFLOPs}\end{aligned}$$

After Pruning. The cost is the sum of the state transition cost and the pruned emission cost ($2k|\mathcal{V}|$).

$$\begin{aligned}\text{Total FLOPs} &= (2 \times 4096^2) \\ &\quad + (2 \times 200 \times 128{,}000) \\ &= (3.36 \times 10^7) + (5.12 \times 10^7) \\ &\approx \textbf{0.08 GFLOPs}\end{aligned}$$

This optimization reduces the HMM's computational overhead from 1.08 GFLOPs to 0.08 GFLOPs, a $\sim$**13x reduction** per decoding step.

E Reproducibility Statement

A detailed description of our experimental setup, including the specific models used, HMM training parameters, and decoding strategy, is provided in the introductory paragraph of Section 4. To allow for replication of our experiments, the full prompts used for the mathematical reasoning (GSM8k), event argument extraction (ACE05), and LLM-as-a-judge (SummEval) tasks are detailed in Section F. Regarding computational overhead, a breakdown of the FLOPs required for the HMM forward pass, both with and without pruning, is presented in Section D.

F Prompt Construction

We present the set of prompts used in the experiments. For GSM8k (Table 8), we sample from a set of task instructions and a set of format instructions to construct prompts for baseline methods. For ACE05 (Table 9), an event description is appended to the task instructions which further explains the event of interest. For SummEval (Table 10), we include domain specific scoring criteria in the task instructions to help LLM align better with human annotations for all methods.

G More EAE Results

In Table 2, we report the *f1-scores* for each method. In Table 7, we present the full results for our event argument extraction experiment.

H HMM Distillation and Usage

For *Llama* and *Qwen*, we distill their HMMs on the LLM continuation only, since the instructions from *Natural-Instructions* are human authored and should not be considered reflecting LLM distribution. We remove the special chat tokens (e.g. <|system|>, <|user|>, etc.) from the responses for HMM to capture the natural language distribution.

We tried different inputs to the HMM, including 1) regular prompt (with chat template), 2) cleaned text prompt (without chat template), and 3) no prompt (empty string). In practice, their results are almost identical. Nonetheless, in accordance with the distillation objective, we report scores yielded from using empty input to the HMM.

Model	Method	AI			AC			AI+			AC+		
		Precision	Recall	f1	Precision	Recall	f1	Precision	Recall	f1	Precision	Recall	f1
Llama-3.1-8B-Instruct	Prompt-Only (NL)	33.7	40.5	36.8	24.2	31.4	27.3	30.5	40.5	34.8	21.7	31.0	25.5
	Prompt-Only (JSON)	30.5	41.8	35.2	21.5	33.3	26.2	27.8	41.7	33.4	19.6	33.0	24.6
	Outlines (NL)	33.7	41.3	37.1	24.4	32.3	27.8	30.5	41.3	35.1	21.8	32.1	26.0
	Outlines (JSON)	28.0	42.0	33.6	19.8	34.4	25.2	25.4	41.6	31.6	18.0	34.1	23.6
	Ctrl-G	38.3	39.9	39.1	25.6	29.8	27.5	34.7	39.6	37.0	23.4	29.5	26.1
	DECO-G	39.4	39.3	39.4	27.3	30.3	28.7	35.5	38.5	37.0	24.5	29.6	26.8
Qwen2.5-7B-Instruct	Prompt-Only (NL)	29.7	36.2	32.6	22.6	29.4	25.5	27.5	35.7	31.2	21.0	29.0	24.4
	Prompt-Only (JSON)	29.1	35.2	31.9	21.3	27.7	24.1	27.0	35.0	30.5	19.7	27.4	22.9
	Outlines (NL)	28.6	39.5	33.2	20.7	31.4	24.9	26.2	39.0	31.3	19.1	31.2	23.7
	Outlines (JSON)	32.2	36.2	34.1	23.9	28.7	26.1	29.6	36.1	32.5	21.9	28.4	24.7
	Ctrl-G	29.9	41.3	34.7	20.1	31.4	24.5	27.8	41.0	33.1	19.3	31.4	23.9
	DECO-G $^{\gamma=2}$	30.9	41.0	35.2	21.8	32.0	25.9	28.4	40.5	33.4	20.1	31.4	24.5
Qwen3-8B	Prompt-Only (NL)	28.0	40.9	33.2	19.5	33.3	24.6	25.6	40.9	31.5	17.8	33.0	23.1
	Prompt-Only (JSON)	27.0	38.4	31.7	18.6	31.7	23.4	24.7	38.4	30.1	16.7	31.4	21.8
	Outlines (NL)	27.9	41.4	33.3	18.9	33.6	24.2	25.4	41.3	31.5	17.1	33.2	22.6
	Outlines (JSON)	26.8	41.4	32.5	17.5	33.4	23.0	24.5	41.3	30.8	15.9	33.2	21.5
	Ctrl-G	27.6	43.2	33.7	17.5	35.0	23.3	24.8	43.5	31.6	16.5	34.8	22.4
	DECO-G $^{\gamma=2}$	27.9	43.6	34.0	18.9	35.1	24.6	25.5	43.4	32.1	17.3	34.8	23.1

Table 7: Full ACE05 Results.

GSM8k	
Task Instructions	1. Follow the instruction to complete the task:\nYou are a math tutor who helps students of all levels understand and solve mathematical problems. \nRead the last question carefully and think step by step before answering, the final answer must be only a number. 2. Follow the instruction to complete the task:\nRead the last question carefully and think step by step before answering, the final answer must be only a number. You are a math tutor who helps students of all levels understand and solve mathematical problems. 3. Follow the instruction to complete the task:\nMathematical problem-solving task:\n- Given: A mathematical question or problem\n- Required: A numerical answer only\n- Role: You are a math tutor assisting students of all levels\n- Process: Think step by step to solve the problem\nNote: Read the question carefully before beginning your analysis.
NL Format Instructions	1. Provide your output in the following text format:\n<think step by step>. The final answer is <answer> 2. Provide your output in the following text format:\nReasoning: <reasoning first>. Answer: The final answer is ...
JSON Format Instructions	Provide your output in the following valid JSON format:\n{“reason”: “<step by step reasoning>”,“answer": “<final answer>”}
Question Example	Janet’s ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers’ market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers’ market?

Table 8: GSM8k prompt construction and an example question.

ACE05	
Task Instructions	You are an argument extractor designed to check for the presence of arguments regarding specific roles for an event in a sentence. \nTask Description: Identify all arguments related to the role Attacker, Target, Instrument, Place, Agent in the sentence. These arguments should have the semantic role corresponding to the given event trigger by the word span between [t] and [/t]. The event of interest is Conflict:Attack. The event is related to conflict and some violent physical act. Roles of interest: Attacker, Target, Instrument, Place, Agent
NL Format Instructions	Provide your output in the following text format:\nThe <role_1> is: <extracted argument>\nThe <role_2> is: <extracted argument>\n...\nThe <role_n> is: <extracted argument>
JSON Format Instructions	Provide your output in the following valid JSON format:\n{"<role>": "<extracted argument>" for role in roles of interest}
Question Example	Text: Efforts were to continue at the United Nations Friday to find a breakthrough in the diplomatic stalemate on Iraq , with Washington warning it could bypass the Security Council and go to [t] war [/t] alone .

Table 9: ACE05 prompt construction and an example question.

SummEval	
Task Instructions (Coherence)	You will be provided with a summary written for a news article \nYour task is to rate the summary based on its coherence. \n\nPlease ensure you read and understand these instructions carefully. Keep this document open while reviewing, and refer to it as needed. \n\nEvaluation Criteria: \nCoherence (1-5): \n- 5: The summary is well-structured and organized, presenting information in a logical and seamless flow. \n- 4: The summary is mostly coherent, with minor lapses in organization or flow. \n- 3: The summary has noticeable organizational issues or lacks a smooth flow but is somewhat understandable. \n- 2: The summary is poorly structured, with significant difficulties in following its logic or flow. \n- 1: The summary is highly disjointed and lacks any meaningful structure or coherence. \nUse these criteria to assign a coherence score between 1 and 5 based on how well the summary organizes and presents information in a clear and logical manner.
Task Instructions (Consistency)	You will be provided with a news article and a summary written for this article. \nYour task is to rate the summary based on its consistency. \n\nPlease ensure you read and understand these instructions carefully. Keep this document open while reviewing, and refer to it as needed. \n\nEvaluation Criteria: \nConsistency (1-5): \n- 5: The summary is fully factually accurate and all its statements are directly supported by the source document. \n- 4: The summary is mostly factually accurate, with only minor errors or omissions. \n- 3: The summary contains noticeable factual errors or unsupported statements but retains some alignment with the source document. \n- 2: The summary has significant factual inaccuracies or includes multiple unsupported claims. \n- 1: The summary is largely inconsistent with the source, containing numerous factual inaccuracies or fabricated details. \nUse these criteria to assign a consistency score between 1 and 5 based on how well the summary aligns factually with the source article.
Task Instructions (Fluency)	You will be provided with a summary written for a news article. \nYour task is to rate the summary based on its fluency. \n\nPlease ensure you read and understand these instructions carefully. Keep this document open while reviewing, and refer to it as needed. \n\nEvaluation Criteria: \nFluency (1-5): \n- 5: The summary is clear and easy to read, with good grammar, spelling, and sentence structure. \n- 4: The summary is generally clear and fluent, with a few minor errors that don't interfere with understanding. \n- 3: The summary has some noticeable issues that might make it a little harder to read but still understandable overall. \n- 2: The summary has more noticeable problems that might make it challenging to follow in places. \n- 1: The summary has significant errors that make it difficult to read or understand in many parts. \nUse these criteria to assign a fluency score between 1 and 5 based on the quality of grammar, word choice, and sentence structure. \nImportant: When evaluating fluency, ignore punctuation and capitalization. Focus only on how natural and easy the language feels regardless of formatting.
Task Instructions (Relevance)	You will be provided with a summary written for a news article. \nYour task is to rate the summary based on its relevance. \n\nPlease ensure you read and understand these instructions carefully. Keep this document open while reviewing, and refer to it as needed. \n\nEvaluation Criteria: \nRelevance (1-5): \n- 5: The summary includes all the important information from the source document with no redundancies or irrelevant details. \n- 4: The summary is mostly relevant, with only minor omissions or slight redundancies. \n- 3: The summary includes some important information but misses key points or has noticeable redundancies. \n- 2: The summary contains limited relevant information, with significant omissions or excessive irrelevant content. \n- 1: The summary is largely irrelevant, failing to capture the main points of the source document. \nUse these criteria to assign a relevance score between 1 and 5 based on how well the summary captures the important content from the source without including excess or redundant information.
NL Format Instructions	Provide your output in the following text format: <analyze the summary>. The rating is <a number between 1 and 5>
JSON Format Instructions	Provide your output in the following valid JSON format:\n{"analysis": "<analyze the summary>","rating": <a number between 1 and 5>}

Table 10: SummEval prompts.