RheoSampling: Resolving the One-Hot Dilemma in Stochastic Dynamic-Tree Speculative Decoding

Qiao Hu[1*] **Yepeng Weng**[2,3*†] **Bo Zhang**[4,5] **Takehisa Yairi**[2]
[1] National Center for Mathematics and Interdisciplinary Sciences (NCMIS), AMSS, CAS
[2] The University of Tokyo [3]Lenovo AI Technology Center
[4] SKLMS and AMSS, Chinese Academy of Sciences
[5] School of Mathematical Sciences, University of Chinese Academy of Sciences
huqiao2020@amss.ac.cn, yweng@g.ecc.u-tokyo.ac.jp

Abstract

Speculative decoding accelerates LLM inference by drafting multiple tokens in parallel, with tree-based methods further improving efficiency through structured hierarchies. Dynamic-tree methods such as EAGLE-3 achieve excellent performance under greedy decoding via deterministic top-K expansion and global pruning. However, in stochastic decoding ($T > 0$), this mechanism collapses the draft distribution into one-hot probabilities, causing a severe drop in acceptance rate. This exposes an apparent dilemma: dynamic-tree methods sacrifice stochastic sampling to preserve context-aware topology, while static-tree methods preserve stochastic sampling with context-agnostic structures. The issue arises because the same probability distribution is used for two conflicting tasks: constructing the tree and verifying the tokens. This coupling makes direct injection of randomness extremely challenging, as we are faced with a complex stochastic process. We resolve this by decoupling these two roles: RheoSampling assigns a token sampled from the draft distribution a *proxy probability* (for tree expansion and pruning) alongside its *true sampling probability* (for verification). Specifically, we inject a sampled token among the deterministic top-K slots and treat it with different probabilities in the construction and verification process, making RheoSampling the first dynamic-tree method with both context-aware top-K construction and stochastic sampling while maintaining losslessness. We establish the lossless guarantee through a novel equivalence-class analysis that compresses the stochastic tree space into tractable classes. An OT-based verification strategy and a sparse draft mechanism ensure that theoretical gains translate into practical efficiency. Experiments across diverse LLMs and benchmarks demonstrate consistent improvements in acceptance rate and speedup over state-of-the-art dynamic tree methods. This framework may provide a template for analyzing other complex stochastic tree structures.

1 Introduction

Large Language Models (LLMs) [17, 18, 6, 24] have demonstrated remarkable capabilities across diverse tasks, yet their autoregressive decoding nature incurs high inference latency. Speculative decoding [11, 2] mitigates this bottleneck by employing a lightweight draft model to predict multiple future tokens in parallel, which are then verified by the target model in a single forward pass. Early approaches draft tokens in a single chain. To further increase the probability of matching the target distribution per step, recent works have shifted towards tree-based speculative decoding [15, 4, 23, 1, 13, 12, 14], which organizes multiple candidate sequences as a tree.

*Equal Contribution.
†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

While initial tree methods employed static or fixed tree structures, state-of-the-art approaches such as EAGLE-2/3 [12, 14] adopt dynamic tree construction, where the draft topology is contextually adapted at each step. These methods typically follow an *expand-then-rerank* paradigm: they greedily expand the draft tree by selecting tokens with the highest path probabilities, then rerank and prune candidates to fit a global verification budget. Notably, this entire pipeline is essentially *deterministic*: once the draft distribution is computed, the tree topology and selected candidates are fully determined, with no randomness involved. Under greedy decoding ($T = 0$), this deterministic paradigm is highly effective. However, under stochastic decoding ($T > 0$), this mechanism collapses the draft distribution into degenerate one-hot probabilities during verification, causing the under-exploration of tail distribution and a severe acceptance-rate drop. This exposes an apparent dilemma: dynamic-tree methods sacrifice stochastic sampling to preserve context-aware topology, while static-tree methods preserve stochastic sampling with context-agnostic structures.

In this paper, we propose **RheoSampling** (Rheostat Sampling), a hybrid paradigm that resolves this challenge. The root difficulty is not merely introducing randomness, but handling a complex stochastic process: the tree topology and the sampled token identities are tightly coupled, yielding a probability space too vast for direct analysis. We overcome this through two synergistic innovations: a *dual-identity* probability design that decouples tree construction from verification, and a novel equivalence-class analysis that compresses the dynamic tree space into tractable classes. Specifically, we insert a single sampled token into the top-K candidate pool, immediately following the top-m deterministic tokens. We then apply a *dual treatment* to the sampled token: during dynamic tree construction, it is assigned a *proxy probability* to compete with deterministic candidates for expansion and survival; during verification, it is evaluated using its *true sampling probability*, guaranteeing losslessness. By decoupling the probabilities for tree construction and verification, RheoSampling injects stochasticity into dynamic drafting and materializes it in the final tree.

To further improve the acceptance rate of RheoSampling, we design a simple yet efficient verification algorithm based on Optimal Transport (OT) and prove its losslessness. Furthermore, we introduce a sparse draft distribution strategy to reduce computational overhead. Extensive experiments across diverse LLMs and benchmarks demonstrate that RheoSampling consistently outperforms top-K-based dynamic trees in both acceptance rate and wall-clock speedup in standard stochastic decoding.

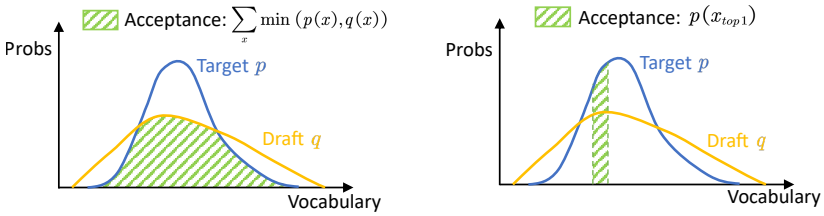
Our contributions are summarized as follows:

- **Dual-identity decoupling for the one-hot dilemma.** RheoSampling gives a stochastically sampled token two independent probability identities: a *proxy probability* for tree construction and a *true sampling probability* for verification. This decoupling resolves the conflict between dynamic top-K construction and stochastic sampling, making RheoSampling the first method to achieve both with rigorous losslessness.
- **Equivalence-class analysis for losslessness.** We develop an equivalence-class methodology that reduces the stochastic tree space to a tractable form, yielding the first rigorous proof of losslessness for stochastic dynamic trees. This analytical framework may provide a template for analyzing other complex stochastic tree structures.
- **Efficient algorithmic design.** We design a verification algorithm for RheoSampling based on Optimal Transport, and introduce a sparse draft distribution that enables efficient sampling over large vocabularies, translating theoretical gains directly into wall-clock speedup.
- **Comprehensive empirical validation.** Extensive experiments on various LLMs and tasks show that RheoSampling consistently achieves higher acceptance rates and speedups over the EAGLE-3 baseline, validating its robustness and generality.

2 Preliminaries

2.1 Tree-Based Speculative Decoding

Tree-based speculative decoding organizes multiple candidate sequences into a tree structure $\mathcal{T}$, allowing the target model to verify diverse drafting paths in parallel. Early approaches employed static tree topologies constructed heuristically (*e.g.,* SpecInfer [15], EAGLE-1 [13]). To further improve tree quality and hit rates, recent works have shifted to dynamic, context-aware construction (e.g., EAGLE-2/3 [12, 14]), where the tree topology adapts to the context. *Unless otherwise specified, we refer to EAGLE as the dynamic tree variant throughout this paper.*



(a) Sample-based Verification (b) Top-1-based Verification

Figure 1: Illustration of acceptance rates under different sampling strategies. (a) Sample-based verification: the acceptance rate equals the distribution overlap $\sum_x \min(p(x), q(x))$. (b) Top-1-based verification: the draft deterministically selects $x_{top1} = \arg\max_x q(x)$, yielding an acceptance rate of $p(x_{top1})$, which equals the probability mass the target model assigns to this single point.

EAGLE employs an *expand-then-rerank* paradigm for dynamic tree construction. At each decoding step, the draft model performs parallel forward passes on K selected parent nodes to obtain next-token distributions. For each parent, it selects the top-K tokens to form a candidate pool, resulting in $K \times K$ leaf nodes per layer. Each candidate's path score is the cumulative probability along its root-to-leaf path. Based on these scores, another top-K selection determines which nodes to expand in the next layer. This process continues until reaching depth D, after which the tree is pruned to satisfy a global budget N, keeping the nodes with the highest scores. A concrete example of this tree-construction procedure is provided in Appendix E.

This mechanism inherently collapses the stochastic draft distribution into a deterministic selection. Specifically, because tokens are exclusively chosen via top-K operations, the actual *proposal distribution* degenerates into a set of point masses. Consequently, the rich long-tail probability information in q is entirely discarded during tree expansion.

2.2 Sampling and Verification Strategies

To illustrate the differences between sampling strategies, consider the single-draft scenario (Fig. 1). For standard speculative sampling (*i.e.,* rejection sampling) [11, 2], the acceptance rate equals the overlap between the target distribution p and the draft distribution q, formally $\alpha^* = \sum_x \min(p(x), q(x))$. In contrast, under top-1 sampling, the draft deterministically selects the token with the highest $q(x)$, and the acceptance rate reduces to the target probability assigned to this single candidate, which is typically much smaller than α^* at high temperature.

These considerations also exist in multi-draft scenarios. In fact, pure top-K sampling suffers from an intrinsic limitation. Even when the draft model perfectly matches the target, top-K sampling cannot achieve 100% acceptance rate if the target assigns positive probability mass outside the candidate set, violating the *optimal transport property*, which is discussed in Sequoia [4]. Modern multi-draft strategies such as recursive rejection sampling [15, 25, 13, 9] and OT-based [20, 19, 8, 22] methods rely on the presence of *at least one stochastically sampled token* within the candidate set. In brief, even a single stochastic token in the candidate pool provides the distributional flexibility for advanced verification algorithms; on the contrary, pure top-K selection inherently lacks this operational space, precluding any non-trivial allocation strategy.

3 Method

3.1 The Incompatibility of Stochastic Sampling and Dynamic Tree Construction

Replacing top-K with direct sampling seems like the natural way to fix the one-hot collapse. However, this naive substitution fails structurally, because it destroys the foundation that dynamic tree construction relies upon.

Ranking by draft probability and its limits. Dynamic tree methods such as EAGLE-2/3 use draft probability $q(x)$ to rank tokens. Tokens with higher $q(x)$ are selected as parent nodes for expansion, and path scores are computed by multiplying $q(x)$ along the path. Under greedy decoding ($T = 0$),

this works well because the draft model's top choice is usually aligned with the target one. However, under non-greedy sampling ($T > 0$), whether a token is accepted depends on the ratio $p(x)/q(x)$, not on the size of $q(x)$ itself. A token with small $q(x)$ can still be accepted if $p(x)$ is large enough, and a high $q(x)$ does not necessarily guarantee a high acceptance rate. Thus, $q(x)$ is no longer a reliable indicator of acceptance. If we directly replace top-K selection with stochastic sampling and use $q(x)$ to rank the sampled tokens, the rules for expanding and pruning the tree become unclear.

The deeper trap: breaking losslessness. A more essential failure arises when sampled tokens are ranked by their raw probabilities. Because a token's value determines its own rank, its survival through pruning depends on its identity. Conditioning on survival distorts the token's conditional distribution away from the original sampling distribution. Once the verification probability no longer matches this distorted distribution, the lossless property is broken.

3.2 RheoSampling: Dual-Identity Decoupling for Dynamic Trees

The naive approach fails because it ties a token's survival in the tree to its own sampling probability. This coupling distorts the conditional distribution of surviving tokens, breaking losslessness. To resolve this, we propose **RheoSampling**, which decouples these two roles. Its core idea is to assign *two independent probability identities to a single stochastic token*: a proxy probability for tree construction (expansion, reranking and pruning), and its true sampling probability for verification.

Hybrid sampling mechanism. Consider a candidate pool with K slots at each expansion step. RheoSampling allocates these slots as follows: first, deterministically select the top-m tokens with highest draft probabilities (the lead tokens); second, sample *one* representative token x_s from the residual distribution $\tilde{q}$ (the tail beyond top-m); finally, fill the remaining $K - m - 1$ slots with the highest-ranked tokens from the remaining vocabulary (the fill tokens). This yields a mixed candidate set combining high-probability tokens with a single stochastic probe into the distribution tail.

Dual treatment of the sampled token. The critical innovation lies in how x_s is treated. During tree construction, it carries a **proxy probability** $q_{\text{proxy}}(x_s) = \min\{q(x_m), z\}$, where $z = 1 - \sum_{i=1}^{m} q(x_i)$ is the residual mass and $q(x_m)$ is the m-th deterministic probability. This construction places x_s ahead of all fill tokens within the candidate pool, ensuring its survival through global pruning is *independent of its realized identity*. This is an essential property for preserving the losslessness (see Section 4). In particular, when $m = 0$, we set the proxy probability to $q(x_1) + \epsilon$, anchoring the sampled token at the first slot; here $+\epsilon$ is purely symbolic and denotes that ties with $q(x_1)$ are broken in favor of the sampled slot. During verification, the token is evaluated using its **true sampling probability** $\tilde{q}(x_s)$, not the proxy. This dual treatment ensures the tree is built on controlled, principled estimates that preserve structural quality, while verification remains lossless by respecting the actual sampling distribution.

The rheostat parameter. By varying m, we control the proxy probability assigned to x_s and the number of deterministic lead tokens preceding it. A smaller m assigns a larger proxy, increasing the sampled token's survival probability and ensuring stochastic exploration materializes in the final tree, but reduces the deterministic backbone. Conversely, a larger m preserves high-quality tree topology yet risks pruning the sampled node, as its proxy weight places it at a lower rank. Thus, m acts as a rheostat, balancing stochastic survival against structural quality.

Efficient implementation via sparse distributions. To mitigate computational overhead from sampling over large vocabularies, we employ a sparse draft distribution strategy. We truncate logits to the top-128 entries prior to softmax, setting remaining logits to $-\infty$. This yields a sparse distribution that enables efficient sampling without iterating over the full vocabulary. Losslessness is preserved as long as verification employs the *same* truncated distribution used for drafting, and acceptance rate is maintained because top-128 entries capture most probability mass of the draft distribution (see Appendix B for more details).

3.3 RheoVerification: Better Acceptance via Optimal Transport

We develop an associated verification mechanism based on Optimal Transport (OT) for RheoSampling to further improve the acceptance rate, as shown in Algorithm 1. We provide a rigorous proof of

its losslessness in Section 4, together with an approximate analysis of its advantage over vanilla sequential rejection verification (Appendix D.3).

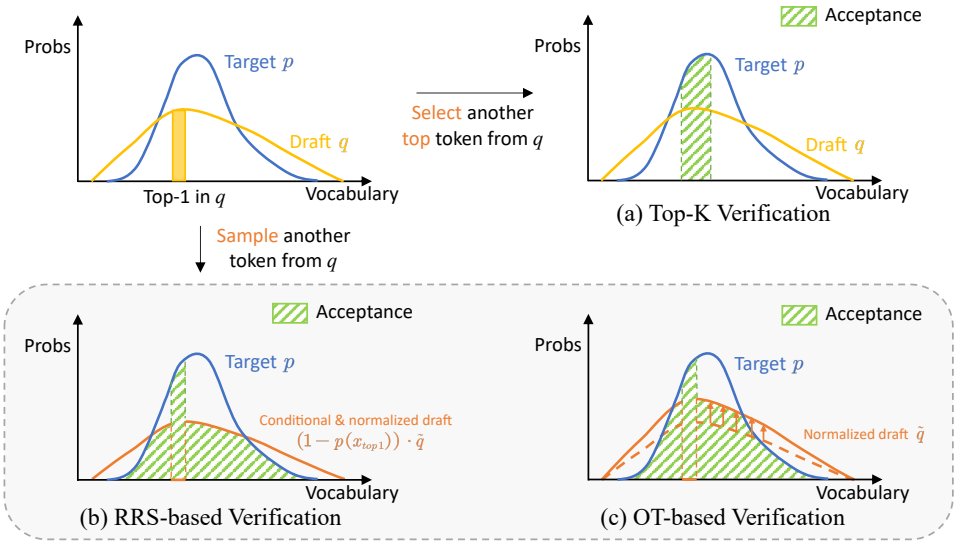


Figure 2: Illustrative comparison of multi-draft verification strategies in a simplified two-draft scenario. Starting with the initial top-1 token, the second candidate is either deterministically selected (a) or stochastically sampled (b, c). (a) Top-K: Covers only isolated point masses. (b) RRS-based: Explores the distribution tail but verifies sequentially. (c) OT-based (Ours): Globally reallocates the probability mass (using the normalized draft $\tilde{q}$, resulting in larger overlap with target p).

Algorithm overview. Given the pruned candidate set $\mathcal{C} = \{u_1, \ldots, u_n\}$ at a layer, if the sampled token u_s survives reranking ($s \neq -1$), we first verify it independently using its *true* sampling probability $\tilde{q}(u_s)$. With probability $\min(1, p(u_s)/\tilde{q}(u_s))$ the token is accepted and we descend into its subtree. If rejected, we compute the residual distribution $r = \text{norm}(\max(0, p - \tilde{q}))$ and perform recursive rejection over the remaining candidates under r. Should every candidate be rejected, we sample once from the final residual. When no sampled token is present ($s = -1$), the verification procedure degenerates to standard top-K verification and relies solely on the target p.

Relation to RRS. A naive alternative for verification is to apply Recursive Rejection Sampling without replacement (RRSw) over the candidates. While RRSw preserves losslessness, it verifies candidates sequentially without globally reallocating the probability mass. As illustrated in Figure 2, pure top-K covers only the isolated point masses. The advantage of our OT-based verification over RRSw lies precisely in achieving higher acceptance rate of the sampled token, effectively utilizing its stochastic flexibility to maximize the overall structural overlap with the target distribution. We provide empirical results in Appendix A.

Algorithm 1 RheoVerification

Require: Target distribution p; tail draft $\tilde{q}$; candidates $\mathcal{C} = \{u_1, \ldots, u_n\}$; sample index s (-1 if pruned)
Ensure: Next token u^*

if $s \neq -1$ **then**
  $u_s \leftarrow \mathcal{C}[s]$
  $\eta \sim \text{Uniform}(0, 1)$
  if $\eta \leq \min(1, p(u_s)/\tilde{q}(u_s))$ **then**
    return u_s
  end if
  $\mathcal{C} \leftarrow \mathcal{C} \setminus \{u_s\}$
  $r \leftarrow \text{norm}(\max(0, p - \tilde{q}))$
else
  $r \leftarrow p$
end if
for $u_i \in \mathcal{C}$ in arbitrary order **do**
  $\eta \sim \text{Uniform}(0, 1)$
  if $\eta \leq r(u_i)$ **then**
    return u_i
  end if
  $r(u_i) \leftarrow 0$; $r \leftarrow \text{norm}(r)$
end for
Sample $u^* \sim r$
return u^*

4 Theoretical Guarantees

The coupling challenge. Establishing losslessness for RheoSampling is substantially harder than for static-tree speculative decoding. In a dynamic tree, the stochastic token $Y \sim \tilde{q}$ not only determines its

own value, but also determines the identities of the subsequent fill tokens. These in turn affect the path scores of deeper nodes and thereby influence which branches survive global pruning. Consequently, the pruned tree T_R is inherently random: both its topology and the token fillings at each node are stochastic, and the two sources of randomness are tightly coupled. We overcome this difficulty via an equivalence-class analysis (full proof in Appendix C; numerical verification in Appendix C.5).

Theorem 1 (Losslessness of RheoSampling)**.** *For any pruning budget $R \geq 1$, any length $L \geq 1$, and any token sequence $\mathbf{Seq} = (v_1, \ldots, v_L)$, let T_R denote the random pruned tree produced by the RheoSampling draft mechanism followed by global top-R pruning, and p is the target model's distribution. Then*

$$\mathbb{E}_{T_R}\Big[\Pr\big(\mathrm{Rheo}(T_R)_{1:L} = \mathbf{Seq}\big)\Big] \;=\; p(\mathbf{Seq}), \tag{1}$$

where $\mathrm{Rheo}(T_R)$ is the output sequence of applying Algorithm 1 on T_R layer-by-layer; once the output token at some layer is a leaf of T_R or is not a node of T_R, the remaining tokens are drawn autoregressively from p (equivalently, by fresh lossless rounds).

Proof sketch. (i) The proxy of the sampled slot is fixed before sampling and dominates every fill token (Lemma 4), so whether the slot survives pruning is independent of the realized Y. (ii) We group unpruned trees into equivalence classes $[\mathcal{U}]_R$ that prune to the same T_R. (iii) We induct on R. Within a class $[\mathcal{U}]_k$, the $(k{+}1)$-th admitted node always occupies the same slot (Lemma 7). If this slot is deterministic, it merely adds a candidate to sequential rejection and leaves the output distribution unchanged; if it is the sampled slot, the class splits by the realized token v with proportions $\tilde{q}(v)$, and averaging Algorithm 1 over $v \sim \tilde{q}$ recovers exactly p at that layer. Each increment of R thus preserves the output distribution, and the base case $R = 1$ reduces to sampling from p.

A necessary bound on the proxy. Independence of the proxy from the realized sampled token is necessary but not sufficient: the proxy must also satisfy the ranking bound $q_{\mathrm{proxy}} \geq q(x_{m+1})$ (Lemma 4). Otherwise the survival of the sampled slot depends on the sampled token, and conditioning on survival distorts the conditional distribution away from $\tilde{q}$. Consider a vocabulary $\{A, B, C\}$ with draft distribution $\{0.5,\ 0.3,\ 0.2\}$. Table 1 shows the rankings under two proxy choices with $K = 3$, $m = 1$.

- Rheo-proxy probability: $q_{\mathrm{proxy}} = \min\{q(A), 1 - q(A)\} = 0.5$. The sampled token always sits at rank 2 regardless of the sampled token Y, so its survival is independent of Y.
- Adhoc-proxy probability: $q_{\mathrm{proxy}} = 0.25$. The rank of Y varies with the fill tokens. Conditioned on survival, Y is forced to be B with probability 1, rather than $\tilde{q}(B) = 0.6$. Algorithm 1 then evaluates Y against the wrong base probability, breaking losslessness.

Table 1: Candidate-pool rankings and survival under two proxy choices.

Proxy	**Sampled Y**	**Fill tokens**	**Pool ranking ($R = 2$)**	**Survival tokens**
Rheo: 0.5	$\tilde{q}(B) = 0.6$	C	$\boxed{A(0.5) \geq Y(0.5)} \geq C(0.2)$	$A, Y \leftarrow B$
	$\tilde{q}(C) = 0.4$	B	$\boxed{A(0.5) \geq Y(0.5)} \geq B(0.3)$	$A, Y \leftarrow C$
Adhoc: 0.25	$\tilde{q}(B) = 0.6$	C	$\boxed{A(0.5) \geq Y(0.25)} \geq C(0.2)$	$A, Y \leftarrow B$
	$\tilde{q}(C) = 0.4$	B	$\boxed{A(0.5) \geq B(0.3)} \geq Y(0.25)$	A, B

Remark 2. *Table 1 shows that **Rheo-proxy** design fixes the sampled slot at rank $m + 1$ within a single pool, making the local sibling order frozen. Yet across the full tree, the realized token Y still enters the path scores of all descendants, and this in turn determines which branches survive global pruning. Hence, the **pruned tree topology** remains an endogenous random variable coupled with the sampled tokens, and a rigorous proof must account for this compounding randomness (Appendix C).*

We derive a closed-form per-layer acceptance rate (Theorem 3); the proof and a comparison with the natural RRSw baseline (which places the sampled token later) are deferred to Appendices D.1–D.3. As shown in Appendix D.3, under the approximation of Eq. (19), Rheo dominates RRSw, which directly motivates the stochastic-first strategy of Algorithm 1; this dominance is not guaranteed for every pair (p, q) without the approximation (Appendix G).

Theorem 3 (Single-layer acceptance rate)**.** *Fix a parent prefix and a local pool size n that is determined before sampling (i.e., not conditioned on which candidates survive global pruning). Given the resulting candidate pool $\mathcal{C}$ with $|\mathcal{C}| = n$ and $s \neq -1$ in Algorithm 1, let Top_n be the n highest-q tokens and x_n the n-th. The per-layer acceptance rate, averaged over $Y \sim \tilde{q}$ and the randomness of Algorithm 1, is*

$$\begin{aligned}\mathcal{A}_{\textit{Rheo}} &= \sum_{v \in \mathrm{Top}_{n-1}} p(v) + \sum_{v \notin \mathrm{Top}_{n-1}} \min\big(p(v), \tilde{q}(v)\big) + r(x_n) \sum_{v \in \mathrm{Top}_{n-1}} \max\big(0, \tilde{q}(v) - p(v)\big) \\ &\leq \sum_{v \in \mathrm{Top}_n} p(v) + \sum_{v \notin \mathrm{Top}_n} \min\big(p(v), \tilde{q}(v)\big), \end{aligned} \tag{2}$$

where $r = \mathrm{norm}(\max(0, p - \tilde{q}))$. In addition, $\mathcal{A}_{\textit{Rheo}} = \sum_{v \in \mathrm{Top}_n} p(v)$ for $s = -1$.

5 Experiments

5.1 Experimental Setup

Datasets and Models. We evaluate RheoSampling across six diverse benchmarks: Alpaca [21], GSM8K [5], HumanEval [3], MT-bench [26], Natural Questions [10], and CNN/DailyMail [16]. Each dataset consists of 80 questions, spanning instruction following, mathematical reasoning, code generation, multi-turn dialogue, question answering, and summarization, ensuring a comprehensive assessment of generation quality and efficiency. For target models, we employ Llama-3.1-8B-Instruct [6], Vicuna-13B-v1.3 [26], and DeepSeek-R1-Distill-Llama-8B [7]. All draft models use the officially released EAGLE-3 checkpoints [14] without further fine-tuning.

Metrics and Implementation. We report two primary metrics: *average acceptance length* (τ), defined as the average number of tokens accepted per drafting-verification cycle, and *actual speedup*, measured as the wall-clock time reduction relative to autoregressive decoding. Our implementation builds upon the open source EAGLE codebase [13, 12, 14]. The default size of the decoding tree is 60 and the draft depth is 8, following the original EAGLE-3 setting. All experiments are conducted on a single NVIDIA A6000 GPU over three independent runs with different random seeds. Unless otherwise specified, we adopt the default temperature $T = 1.0$ and the rheostat parameter $m = 1$.

5.2 Main Results

Table 2 presents the end-to-end evaluation of RheoSampling against the vanilla Top-K baseline across six benchmarks and three target models. RheoSampling achieves consistent improvements in mean acceptance length (τ) across all configurations, with mean gains (averaged over the six benchmarks) ranging from $+0.14$ (Vicuna-13B) to $+0.22$ (DeepSeek-R1-Distill-8B). The absolute improvement varies by task and model, reflecting differences in draft-target alignment and distribution tail mass. Nevertheless, the average gains over pure top-K sampling are consistently observed across all three random seeds.

These improvements in acceptance length translate directly into wall-clock speedup, though the relative speedup improvement is slightly smaller than the τ gain. For instance, on Llama-3.1-8B, τ improves by 4.2% ($5.04 \to 5.25$) while speedup increases by 3.2% ($2.84\times \to 2.93\times$). This stems from the additional overhead of sampling and verifying the stochastic probe, which is not present in the pure Top-K baseline. Nevertheless, the net speedup is positive across all models, confirming that the theoretical benefits of stochastic exploration outweigh its marginal computational cost.

5.3 Ablation Study on Rheostat Parameter

As shown in Figure 3, $m = 1$ achieves the best or near-best performance in most settings. $m = 0$ and $m = 2$ remain viable and outperform the Top-K baseline, but $m \geq 3$ suffers from diminishing proxy probability, so the sampled token is pruned during reranking, yielding marginal gains.

Verification perspective. When $m = 0$, the residual distribution $\tilde{q}$ degenerates to the full draft distribution q, reducing the verification of the sampled token to standard rejection sampling. As a result, the OT-based probability reallocation, which benefits from the exclusion of top-m mass to raise the normalized residual probability, is not fully exploited. As m increases, the top-m mass

Table 2: Main results with EAGLE-3 on six different tasks. We report the mean acceptance length (mean$_{\pm\text{std}}$ over 3 runs) and the end-to-end speedup. V-13B, L31-8B and DSL-8B denote Vicuna-13B-v1.3, Llama-3.1-8B-Instruct and DeepSeek-R1-Distill-Llama-8B, respectively.

Model	Sampling	Tasks						Average	
		Alpaca	Math	Code	MT	QA	Sum	τ	Speedup
V-13B	Top-K	5.59$_{\pm0.09}$	5.85$_{\pm0.09}$	6.66$_{\pm0.13}$	5.69$_{\pm0.06}$	4.91$_{\pm0.08}$	5.80$_{\pm0.03}$	5.75$_{\pm0.03}$	3.37×
	Rheo	**5.92**$_{\pm0.05}$	**5.87**$_{\pm0.05}$	**6.72**$_{\pm0.06}$	**5.88**$_{\pm0.09}$	**5.00**$_{\pm0.04}$	**5.94**$_{\pm0.07}$	**5.89**$_{\pm0.03}$	**3.43**×
L31-8B	Top-K	5.68$_{\pm0.02}$	5.50$_{\pm0.03}$	6.11$_{\pm0.09}$	4.60$_{\pm0.06}$	3.89$_{\pm0.04}$	4.49$_{\pm0.07}$	5.04$_{\pm0.02}$	2.84×
	Rheo	**6.03**$_{\pm0.12}$	**5.60**$_{\pm0.08}$	**6.23**$_{\pm0.03}$	**4.84**$_{\pm0.14}$	**4.14**$_{\pm0.04}$	**4.66**$_{\pm0.07}$	**5.25**$_{\pm0.03}$	**2.93**×
DSL-8B	Top-K	4.48$_{\pm0.04}$	6.70$_{\pm0.05}$	5.43$_{\pm0.09}$	4.93$_{\pm0.02}$	4.08$_{\pm0.05}$	4.32$_{\pm0.06}$	4.99$_{\pm0.01}$	2.89×
	Rheo	**4.79**$_{\pm0.06}$	**6.71**$_{\pm0.07}$	**5.79**$_{\pm0.05}$	**5.16**$_{\pm0.05}$	**4.35**$_{\pm0.05}$	**4.47**$_{\pm0.06}$	**5.21**$_{\pm0.02}$	**2.99**×

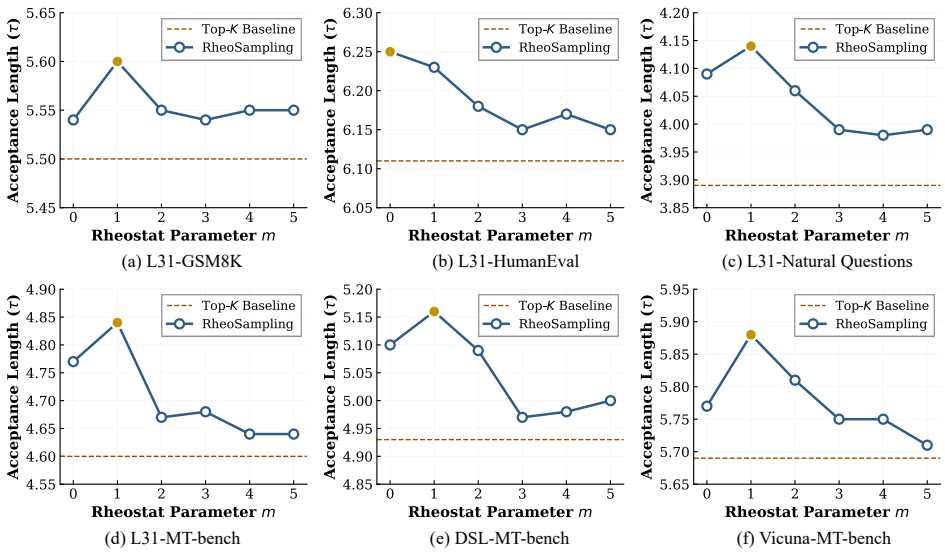
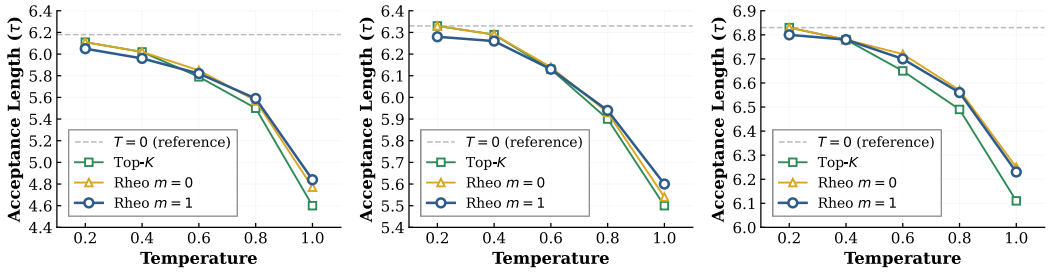


Figure 3: Ablation on the rheostat parameter m. **Top row:** Llama-3.1-8B-Instruct across tasks (GSM8K, HumanEval, Natural Questions). **Bottom row:** Results on MT-bench across different target models (Llama-3.1-8B-Instruct, DeepSeek-R1-Distill-Llama-8B, Vicuna-13B-v1.3).

enters the normalization denominator, enabling higher theoretical acceptance bounds under the OT framework; however, excessively large m suppresses the proxy probability below the survival threshold, preventing the sampled token from realizing these gains in the final tree.

Tree structure perspective. The impact of the sampled token on tree topology depends on its assigned slot. Under $m = 0$, the proxy equals $q(x_1)$ with ties broken in favor of the sampled slot, anchoring the token at the first slot. Therefore, if the sampled token coincides with the original top-1 candidate, the behavior of tree expansion and pruning at this node is effectively identical to pure Top-K; otherwise, the tree will alter, introducing unpredictable structural deviation. In this sense, $m = 0$ is an exception: it may preserve the original structure by chance, or perturb it significantly. By contrast, for any $m > 0$, the proxy injection definitely affects the tree structure, though the impact becomes increasingly modest as m grows and more deterministic tokens precede the sampled one.

The sweet spot of rheostat. Under standard stochastic decoding conditions, $m = 1$ strikes a favorable balance: it anchors the sampled token at the second slot, ensuring sufficient proxy mass to survive reranking while fully leveraging the OT-based verification advantage. The structural impact is moderate and stable, unlike the all-or-nothing behavior of $m = 0$. In fact, this trade-off is temperature-dependent. We analyze this interaction in Section 5.4.



(a) MT-bench (b) GSM8K (c) HumanEval

Figure 4: Temperature ablation on Llama-3.1-8B-Instruct across MT-bench, GSM8K, and HumanEval. The dashed grey line marks the $T = 0$ (greedy decoding) reference.

5.4 Impact of Temperature

Decoupling tree construction from temperature. An important implementation detail is that tree construction (expansion and reranking) operates on the *original* draft probabilities without temperature scaling, consistent with standard EAGLE practice. Temperature only affects the sampling and verification stages. This decoupling is necessary because low temperatures can distort the relative ranking of tokens and destroy the ordinal information.

Acceptance Overview. Figure 4 demonstrates the impact of temperature. As temperature decreases, the distribution becomes increasingly sharp, causing any probability-based sampling to degenerate toward deterministic top-token selection. Mathematically, the verification strategies such as Top-K, RRS-based or our OT-based scheme become nearly equivalent in this limit, and their difference in acceptance rate narrows. All methods exhibit rising acceptance lengths as T drops, though their respective structural behaviors remain distinct.

Degeneracy asymmetry. When $T \to 0$, RheoSampling degenerates along two axes: *sampling degeneracy* (the stochastic probe converges to deterministic selection) and *structural degeneracy* (the tree topology matches pure Top-K). Yet $m = 0$ and $m = 1$ degenerate differently. Under $m = 0$, the proxy is $q(x_1) + \epsilon$. At low temperature, the residual concentrates on the true top-1 candidate, which is almost surely sampled. Since the proxy closely matches the original top-1 probability used for tree construction, the resulting topology is identical to pure Top-K. Thus, $m = 0$ achieves *dual degeneracy*: both sampling and structure collapse to the Top-K baseline. Under $m = 1$, only sampling degenerates. At $T \to 0$, the residual concentrates on the true second-ranked token, but its proxy $\min\{q(x_1), z\}$ does not equal its original probability $q(x_2)$. While verification evaluates the token using its true sampling probability, tree construction still operates on a mismatched proxy, perturbing path scores. Consequently, $m = 1$ exhibits only *single degeneracy*: sampling collapses to Top-K, but the tree structure remains distinct. This structural discrepancy causes the acceptance gap between $m = 1$ and the Top-K / $m = 0$ baselines as $T \to 0$.

Remarks on proxy design. As discussed in Section 4 and Lemma 4, any proxy probability that does not depend on the realized sampled token and is no smaller than $q(x_{m+1})$ (with ties broken in favor of the sampled slot) preserves the losslessness guarantee. Setting the proxy to $q(x_{m+1}) + \epsilon$ would achieve the "dual degeneracy" for arbitrary m at low temperature, since the sampled token would carry its true original ranking probability into tree construction. Conversely, our choice of $\min\{q(x_m), z\}$ prioritizes the survival probability of the sampled token during reranking by granting it a larger proxy mass. *Both proxy designs are valid instantiations within the RheoSampling framework; the choice of proxy is a matter of empirical tuning, not theoretical necessity.*

6 Related Work

Speculative decoding and tree structures. Speculative decoding introduces a drafting-verification paradigm that accelerates LLM inference without sacrificing generation quality [11, 2]. Early works primarily employed static tree structures for parallel verification, such as the manually designed trees

in Medusa [1] and SpecInfer [15], as well as EAGLE-1 [13]. Subsequent research shifted towards dynamic, context-aware tree construction. EAGLE-2 [12] and EAGLE-3 [14] employ an expand-then-rerank paradigm that adaptively selects candidates based on path probabilities. Opt-Tree [23] shares the motivation for context-aware dynamic construction, employing a slightly different optimization-based approach. Sequoia [4] adopts a hardware-aware tree topology via dynamic programming.

Multi-draft verification strategies. Verification strategies have evolved from chain-based to tree-based settings. SpecInfer [15] introduced Recursive Rejection Sampling (RRS) for multi-draft scenarios, later refined into RRS without replacement [13, 4, 25, 9] to prevent repeat sampling of identical tokens. From an optimal transport (OT) perspective, SpecTr [20] first formulated multi-draft verification as OT problem. Subsequent works such as SpecHub [19] and Greedy method [8] explored hybrid drafting strategies that combine deterministically selected drafts with sampled tokens. However, these approaches are designed for static candidate pools, whereas RheoSampling integrates hybrid stochasticity into *dynamic* tree construction, enabling context-aware expansion with lossless OT-based verification.

7 Conclusion

We present **RheoSampling**, resolving the one-hot dilemma in dynamic-tree speculative decoding where deterministic expansion inherently conflicts with stochastic generation. By assigning a sampled token a *proxy probability* for context-aware tree construction and its *true sampling probability* for OT-based verification, our dual-identity framework successfully decouples topology from randomness. Crucially, we establish the first rigorous proof of losslessness for stochastic dynamic trees through a novel equivalence-class analysis that compresses the otherwise intractable probability space. Coupled with a sparse draft mechanism, RheoSampling translates these theoretical guarantees into consistent empirical speedups and superior acceptance rates over state-of-the-art baselines. By unifying dynamic topologies with mathematically sound stochastic sampling, this work provides a foundational template for adapting advanced verification algorithms to complex tree structures.

Acknowledgments and Disclosure of Funding

We thank the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, The University of Tokyo, and Lenovo AI Technology Center for their support of this work.

References

[1] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Proceedings of the International Conference on Machine Learning*.

[2] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

[3] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

[4] Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. 2024. Sequoia: Scalable and robust speculative decoding. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

[5] 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. *arXiv preprint arXiv:2110.14168*.

[6] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

[7] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

[8] Zhengmian Hu, Tong Zheng, Vignesh Viswanathan, Ziyi Chen, Ryan A. Rossi, Yihan Wu, Dinesh Manocha, and Heng Huang. 2025. Towards optimal multi-draft speculative decoding. In *Proceedings of the International Conference on Learning Representations*.

[9] Wonseok Jeon, Mukul Gagrani, Raghavv Goel, Junyoung Park, Mingu Lee, and Christopher Lott. 2024. Recursive speculative decoding: Accelerating LLM inference via sampling without replacement. *arXiv preprint arXiv:2402.14160*.

[10] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

[11] Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*.

[12] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024. EAGLE-2: faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*.

[13] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024. EAGLE: speculative sampling requires rethinking feature uncertainty. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*.

[14] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Annual Conference on Neural Information Processing Systems*.

[15] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 2024. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on*

Architectural Support for Programming Languages and Operating Systems, Volume 3, ASPLOS 2024, La Jolla, CA, USA, 27 April 2024- 1 May 2024.

[16] Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*.

[17] OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

[18] OpenAI. 2026. Openai GPT-5 system card. *arXiv preprint arXiv:2601.03267*.

[19] Ryan Sun, Tianyi Zhou, Xun Chen, and Lichao Sun. 2024. Spechub: Provable acceleration to multi-draft speculative decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*.

[20] Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix X. Yu. 2023. Spectr: Fast speculative decoding via optimal transport. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

[21] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

[22] Rahul Krishna Thomas and Arka Pal. 2026. Global resolution: Optimal multi-draft speculative sampling via convex optimization. In *The Fourteenth International Conference on Learning Representations*.

[23] Jikai Wang, Yi Su, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, and Min Zhang. 2025. Opt-tree: Speculative decoding with adaptive draft tree structure. *Trans. Assoc. Comput. Linguistics*.

[24] 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, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

[25] Sen Yang, Shujian Huang, Xinyu Dai, and Jiajun Chen. 2024. Multi-candidate speculative decoding. *arXiv preprint arXiv:2401.06706*.

[26] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

A Comparison on Verification Strategies

While RheoSampling's hybrid draft strategy inherently improves acceptance by covering the distribution tail, the verification algorithm itself also contributes to the final rate. As we have mentioned in 3.3, we design RheoVerification from an Optimal Transport perspective, achieving a higher acceptance rate for the sampled token while maintaining losslessness. In contrast, Recursive Rejection Sampling without replacement (RRSw) processes candidates sequentially without such global optimization.

Table 3 presents the end-to-end performance across multiple benchmarks. RheoSampling paired with RRSw already outperforms the vanilla Top-K baseline, confirming the benefit of injecting stochasticity into the draft tree. Replacing RRSw with RheoVerification yields further consistent gains, validating that the RheoVerification captures additional acceptance probability beyond sequential rejection.

To isolate the pure effect of the verification strategy, we conduct a controlled demo experiment on MT-bench with fixed settings: draft depth is restricted to one layer, and the three candidate slots

Table 3: End-to-end performance comparison across benchmarks. RheoSampling combined with RheoVerification consistently outperforms both the RRSw variant and the vanilla Top-K baseline.

Sampling	Verification	Alpaca	GSM8K	HumanEval	MT-bench	Natural Q.	CNN/DM	Average
Top-K	Vanilla	5.68	5.50	6.11	4.60	3.89	4.49	5.04
Rheo	RRSw	5.88	5.47	6.18	4.67	4.09	4.58	5.15
	Rheo	**6.03**	**5.60**	**6.23**	**4.84**	**4.14**	**4.66**	**5.25**

are set to top-1, sampled token, and the highest-ranked token outside the first two. Under this configuration, the difference between the two Rheo rows stems solely from the verification algorithm. As shown in Table 4, RheoVerification achieves 85.4% acceptance, compared to 83.5% under RRSw and 80.0% under vanilla Top-K. The 3.5% gap between RRSw and Top-K reflects the gain from hybrid sampling alone, while the additional 1.9% gap between RheoVerification and RRSw precisely reflects the gain from OT-based mass allocation.

Table 4: Acceptance rate comparison under controlled settings (MT-bench, draft depth=1, fixed candidate slots: top-1, sampled-1, and top-1 of remaining).

Sampling	Verification	Acceptance
Top-K	Vanilla	80.0%
Rheo	RRSw	83.5%
	Rheo	**85.4%**

B Discussion on Sparse Draft Probability

To validate the practical impact of sparse draft distributions, we conduct ablation studies on MT-bench, measuring draft coverage, acceptance length (τ), and drafting latency across varying support sizes. Results are summarized in Table 5.

Impact on acceptance rate. As shown in Table 5, the primary purpose of the *Full* configuration is to provide an empirical reference value (4.89) using the dense draft distribution. Sparse support sizes from 128 to 1024 achieve comparable τ values (4.78–4.85), confirming that truncation does not systematically degrade acceptance performance. This is expected for two reasons: first, even at support size 128, the coverage of the original draft distribution exceeds 91% on MT-bench (even higher on other datasets such as $\sim$96% on Alpaca and $\sim$95% on HumanEval), meaning the sparse and dense distributions are nearly identical for practical purposes. Second, the sparse operation only affects the single stochastically sampled token, leaving all deterministic top-K candidates untouched. The minor fluctuations in τ across support sizes are well within statistical noise, and notably, the 128-support configuration actually outperforms the 512-support variant, suggesting that sparse approximation does not introduce consistent penalty.

Latency. For support sizes below 1024, drafting latency remains stable (within $\pm$0.2 ms of the 128 baseline), indicating that sparse truncation introduces negligible overhead to the overall tree construction pipeline. We adopt 128 as the default to avoid over-engineering hyperparameters.

C Losslessness of RheoSampling

C.1 Setup and Notation

Let p denote the target next-token distribution at a given decoding layer, conditioned on the prefix generated so far. Let q denote the original draft distribution over the full/sparse vocabulary $\mathcal{V}$.

Table 5: Effect of sparse support size on draft coverage and average acceptance length (τ). The *Full* row provides an empirical reference using the dense vocabulary distribution. Drafting latency is reported for completeness; the Full configuration uses native PyTorch operators such as concatenation and multinomial, and is not optimized for speed.

Sampling	Support Size	Coverage of q	τ	Drafting Latency
Rheo	128	91.0%	4.84	13.1 ms
	256	91.8%	4.85	13.2 ms
	512	92.2%	4.78	13.3 ms
	1024	94.1%	4.83	13.3 ms
	Full	100.0%	4.89	18.8 ms[†]
Top-K	N/A	N/A	4.60	12.9 ms

[†]PyTorch native implementation over the full vocabulary and not optimized for speed; included as a τ reference only.

Hybrid candidate pool. For a fixed parent node, the top-m deterministic tokens $\{x_1, \ldots, x_m\}$ with probabilities $\{q(x_i)\}_{i=1}^m$, and the residual mass

$$z \;=\; 1 - \sum_{i=1}^{m} q(x_i).$$

The stochastic token Y is sampled from the tail distribution

$$\tilde{q}(Y) \;=\; \frac{q(Y)}{z}, \qquad \forall Y \notin \{x_1, \ldots, x_m\}.$$

The remaining $K - m - 1$ slots are filled with the highest-ranked tokens from $\mathcal{V} \setminus \{x_1, \ldots, x_m, Y\}$ without replacement, denoted $\{X_{m+2}, \ldots, X_K\}$. Each node in the candidate pool carries a *proxy probability* for tree construction:

- Top-m tokens: $q_{\text{proxy}}(x_i) = q(x_i), i = 1, \ldots, m$;
- Sampled token: $q_{\text{proxy}}(Y) = \min\{q(x_m), z\}$;
- Fill tokens: $q_{\text{proxy}}(X_j) = q(X_j), j = m + 2, \ldots, K$.

Special case $m = 0$. Here x_m is undefined and $z = 1$. We set $q_{\text{proxy}}(Y) = q(x_1) + \epsilon$ as in Section 3, where $+\epsilon$ is symbolic: numerically $q_{\text{proxy}}(Y) = q(x_1)$, and any tie with a fill token is resolved in favor of the sampled slot, which occupies the first (leftmost) slot and is therefore handled by the sibling-order tie-breaking rule below. Throughout this appendix, $\min\{q(x_m), z\}$ is read as this convention when $m = 0$.

Path scores and global pruning. For any node u in the unpruned tree $\mathcal{U}$, its path proxy score is the product of proxy probabilities along the root-to-u path. The global pruning operator $\text{Prune}_R(\mathcal{U})$ retains exactly the R nodes with highest path scores (ties broken arbitrarily but deterministically). Given $\mathcal{U}$, the pruned tree $T_R = \text{Prune}_R(\mathcal{U})$ is fully deterministic.

Tie-breaking rule. Global top-R pruning ranks nodes first by path proxy score in descending order. When scores are equal, we break ties by depth (shallower nodes prioritized over deeper nodes) and then by left-to-right sibling order. Under this rule, the pruned dynamic tree is always ancestor-closed and connected.

Verification input. At a fixed layer ℓ of the pruned tree T_R, let $\mathcal{C} = \{u_1, \ldots, u_n\}$ be the set of candidate tokens at that layer, and let $s \in \{-1, 1, \ldots, n\}$ indicate the index of the sampled token ($s = -1$ if the sampled node was pruned away).

C.2 Structural Properties of the Proxy Scores

The first lemma states that, within any single candidate pool, the sampled token is always ranked above every fill token, regardless of its realized identity.

Lemma 4 (Local ranking preservation)**.** *The proxy probability of the sampled token satisfies*

$$q_{\text{proxy}}(Y) \;=\; \min\{q(x_m),\, z\} \;\geq\; q(x_{m+1}),$$

where $q(x_{m+1})$ is the $(m+1)$-th largest probability in the draft distribution q. Consequently, the proxy probability of every fill token X_j satisfies $q(X_j) \leq q_{\text{proxy}}(Y)$.

Proof. By sorting, $q(x_{m+1}) \leq q(x_m)$. Moreover $z = \sum_{i=m+1}^{|\mathcal{V}|} q(x_i) \geq q(x_{m+1})$. Hence $q(x_{m+1}) \leq \min\{q(x_m), z\} = q_{\text{proxy}}(Y)$. For $m = 0$, $q_{\text{proxy}}(Y) = q(x_1)$ numerically, and the bound holds with equality. Since every fill token is drawn from $\mathcal{V} \setminus \{x_1, \ldots, x_m, Y\}$, its proxy probability is at most $q(x_{m+1})$. □

Lemma 4 guarantees that, although the *identities* of the fill tokens X_j depend on the realized Y (due to sampling without replacement), their *scores* can never surpass the sampled token's proxy score. This fixes the sampled node at rank $m+1$ inside its sibling group.

The second lemma captures the ancestor-closed nature of global pruning.

Lemma 5 (Ancestor monotonicity)**.** *For any node u in an unpruned dynamic tree $\mathcal{U}$, let $S(u)$ denote its path proxy score (the product of proxy probabilities along the root-to-u path). If v is a child of u, then $S(v) \leq S(u)$.*

Proof. $S(v) = S(u) \cdot q_{\text{proxy}}(v)$ and $q_{\text{proxy}}(v) \leq 1$ by construction. □

C.3 Proof of Theorem 1 via Equivalence Classes

We now turn to the main result. For any unpruned tree $\mathcal{U}$ and any positive integer R, the pruned tree $T_R = \text{Prune}_R(\mathcal{U})$ is a deterministic function of $\mathcal{U}$. We define the equivalence class

$$[\mathcal{U}]_R \;=\; \big\{\mathcal{U}' \mid \text{Prune}_R(\mathcal{U}') = T_R\big\},$$

which collects all unpruned trees that prune to the same R-node tree. Let $\text{Rheo}([\mathcal{U}]_R)$ denote the random output sequence obtained by running Algorithm 1 layer-by-layer on T_R. The randomness of $\text{Rheo}([\mathcal{U}]_R)$ is fully from the internal coin flips of Algorithm 1. As in Theorem 1, once the output leaves T_R (the emitted token is a leaf or not a node of T_R), the output is continued autoregressively from p, so $\text{Rheo}([\mathcal{U}]_R)$ is defined for every length; throughout, $\mathbf{Seq} = (v_1, \ldots, v_L)$ has an arbitrary fixed length L and the event $\{\text{Rheo} = \mathbf{Seq}\}$ refers to the first L output tokens.

Our goal is to prove that, for any sequence $\mathbf{Seq}$ and any $R \geq 1$,

$$\mathbb{E}_{[\mathcal{U}]_R}\Big[\Pr\big(\text{Rheo}([\mathcal{U}]_R) = \mathbf{Seq}\big)\Big] \;=\; p(\mathbf{Seq}), \tag{3}$$

where $p(\mathbf{Seq})$ is the probability assigned to $\mathbf{Seq}$ by standard autoregressive sampling from the target model.

Theorem 6 (Losslessness)**.** *Equation* (3) *holds for every positive integer R and every sequence* $\mathbf{Seq}$*.*

Proof. We proceed by mathematical induction on R.

Base case ($R = 1$). The pruned tree T_1 contains exactly one node, which must reside at depth 1 by Lemma 5. Two sub-cases arise:

Case $m = 0$. The retained node is the sampled token. Its proxy probability is $q(x_1) + \epsilon$, so it is ranked first among the depth-1 candidates: no fill token exceeds $q(x_1)$, and any tie is broken in favor of the sampled slot. Consequently, Algorithm 1 performs standard rejection sampling with proposal $\tilde{q}$ and target p. The losslessness is trivial.

Case $m > 0$. The retained node is the top-1 deterministic candidate. Algorithm 1 sets $s = -1$ and $r \leftarrow p$, so the output is sampled directly from p.

In both cases, the expectation $\mathbb{E}_{[\mathcal{U}]_1}\left[\Pr(\text{Rheo}([\mathcal{U}]_1) = \mathbf{Seq})\right] = p(\mathbf{Seq})$.

Inductive hypothesis. Assume that for $R = k$ and all sequences $\mathbf{Seq}$,

$$\mathbb{E}_{[\mathcal{U}]_R}\Big[\Pr\big(\text{Rheo}([\mathcal{U}]_R) = \mathbf{Seq}\big)\Big] \;=\; p(\mathbf{Seq}). \tag{4}$$

Inductive step ($R = k + 1$). Fix an arbitrary equivalence class $[\mathcal{U}]_k$ and consider all unpruned trees $\mathcal{U}' \in [\mathcal{U}]_k$. For each such $\mathcal{U}'$, let u' be the unique node in $\mathrm{Prune}_{k+1}(\mathcal{U}') \setminus \mathrm{Prune}_k(\mathcal{U}')$ (the newly admitted $(k+1)$-th node). The following structural fact is essential and its proof is deferred to Appendix C.4.

Lemma 7 (Slot determinism)**.** *For any equivalence class $[\mathcal{U}]_k$, if the node $u' = \mathrm{Prune}_{k+1}(\mathcal{U}') \setminus \mathrm{Prune}_k(\mathcal{U}')$ exists for some $\mathcal{U}' \in [\mathcal{U}]_k$, then*

1. ***(Topological invariance)** The slot of u' are identical across all $\mathcal{U}' \in [\mathcal{U}]_k$, which means that $\mathrm{Prune}_{k+1}(\mathcal{U}')$ have the same topological structure for all $\mathcal{U}' \in [\mathcal{U}]_k$.*
2. ***(Deterministic consistency)** If the slot type of u' is deterministic (top-m slot or fill slot), then $\mathrm{Prune}_{k+1}(\mathcal{U}')$ is identical for every $\mathcal{U}' \in [\mathcal{U}]_k$, which means $[\mathcal{U}]_{k+1} \subset [\mathcal{U}]_k$.*
3. ***(Stochastic fidelity)** If the slot type of u' is sampled, then $[\mathcal{U}]_k = \bigsqcup_v [\mathcal{U}]_k^{(v)}$ are partitioned into disjoint subclasses according to the realized token v of u', and the proportion of each subclass equals $\tilde{q}(v)$.*

By Lemma 5 and the tie-breaking rule, u' is a leaf in $T_{k+1} = \mathrm{Prune}_{k+1}(\mathcal{U}')$, so it affects only the single layer ℓ where it resides. Let $T_k = \mathrm{Prune}_k(\mathcal{U}')$ (which is constant for all $\mathcal{U}' \in [\mathcal{U}]_k$ by definition) and let C be the candidate set of T_k at layer ℓ.

We now split the analysis according to the slot type of u'.

Case 1: deterministic slot (top-m or fill). By Lemma 7(2), the pruned tree T_{k+1} is identical for every $\mathcal{U}' \in [\mathcal{U}]_k$. During RheoSampling, the verification on T_k and T_{k+1} proceeds identically until layer ℓ. Algorithm 1 processes u' as an additional point mass in the sequential-rejection pool. Whether u' is accepted directly or bypassed, the total probability mass allocated to each token remains governed by the target distribution r (or p). The presence of u' alters only the acceptance length, not the marginal output distribution at layer ℓ. Moreover, since u' is a leaf of T_{k+1}, by the same continuation argument as in Case 2 below, the conditional law of the tokens after layer ℓ given the emitted token is identical for T_k and T_{k+1}. Hence, for every $\mathcal{U}' \in [\mathcal{U}]_k$,

$$\Pr\big(\text{Rheo}([\mathcal{U}']_{k+1}) = \mathbf{Seq}\big) \;=\; \Pr\big(\text{Rheo}([\mathcal{U}]_k) = \mathbf{Seq}\big). \tag{5}$$

Case 2: sampled slot. By Lemma 7(3), the equivalence class $[\mathcal{U}]_k$ is partitioned into subclasses $[\mathcal{U}]_k^{(v)}$ indexed by the realized token v of u', with $\Pr(\mathcal{U}' \in [\mathcal{U}]_k^{(v)}) = \tilde{q}(v)$. For each v, the pruned tree is $T_{k+1}^{(v)} = T_k \cup \{u'_v\}$, where u'_v denotes the sampled node filled with token v. At layer ℓ, T_k has $s = -1$ (the sampled slot was previously empty), while $T_{k+1}^{(v)}$ has $s \neq -1$ pointing to u'_v with true sampling probability $\tilde{q}(v)$.

Algorithm 1 first tests u'_v with acceptance probability $\min(1, p(v)/\tilde{q}(v))$. If accepted, the output at layer ℓ is v; if rejected, the residual distribution

$$r \;=\; \mathrm{norm}\big(\max(0,\, p - \tilde{q})\big)$$

is verified over the deterministic candidate set C. Let $h_{\mathrm{res}}(C, v_\ell)$ denote the probability that Algorithm 1, operating on the deterministic set C under target distribution r, outputs token v_ℓ (the token appearing in $\mathbf{Seq}$ at layer ℓ). A standard sequential-rejection argument shows $h_{\mathrm{res}}(C, v_\ell) = r(v_\ell)$.

Averaging over the realized token $v \sim \tilde{q}$, the probability of emitting v_ℓ at layer ℓ is

$$\begin{aligned}
&\sum_v \tilde{q}(v)\Big[\mathbb{I}(v = v_\ell)\min\Big(1, \frac{p(v)}{\tilde{q}(v)}\Big) \;+\; \Big(1 - \min\Big(1, \frac{p(v)}{\tilde{q}(v)}\Big)\Big)\, h_{\mathrm{res}}(C, v_\ell)\Big] \\
&=\; \min\big(p(v_\ell), \tilde{q}(v_\ell)\big) \;+\; h_{\mathrm{res}}(C, v_\ell) \sum_v \max\big(0, \tilde{q}(v) - p(v)\big) \\
&=\; \min\big(p(v_\ell), \tilde{q}(v_\ell)\big) \;+\; \frac{\max\big(0, p(v_\ell) - \tilde{q}(v_\ell)\big)}{\sum_t \max(0, p(t) - \tilde{q}(t))} \cdot \sum_v \max\big(0, p(v) - \tilde{q}(v)\big) \\
&=\; \min\big(p(v_\ell), \tilde{q}(v_\ell)\big) \;+\; \max\big(0, p(v_\ell) - \tilde{q}(v_\ell)\big) \;=\; p(v_\ell).
\end{aligned} \tag{6}$$

Meanwhile, T_k at layer ℓ (with $s = -1$ and $r \leftarrow p$) also emits v_ℓ with probability $p(v_\ell)$.

It remains to check that the continuation after layer ℓ is also unchanged. The two trees T_k and $T_{k+1}^{(v)}$ coincide everywhere except for the leaf u'_v, so the verification before layer ℓ is identical. Conditional on emitting a token w at layer ℓ: if $w \in C$, both trees descend into the same subtree of w (which does not contain u'_v); otherwise w is either the leaf u'_v of $T_{k+1}^{(v)}$ or not a node of either tree (note that $v \notin C$, since the sampled token is distinct from its deterministic siblings), and in both trees the remaining tokens are drawn autoregressively from $p(\cdot \mid v_{1:\ell-1}, w)$. Hence the conditional law of the tokens after layer ℓ given $(v_{1:\ell-1}, w)$ is identical for T_k and $T_{k+1}^{(v)}$, and combining it with the equality of the layer-ℓ marginals above, we obtain

$$\mathbb{E}_{\mathcal{U}' \in [\mathcal{U}]_k}\Big[\Pr\big(\text{Rheo}([\mathcal{U}']_{k+1}) = \mathbf{Seq}\big)\Big] \;=\; \Pr\big(\text{Rheo}([\mathcal{U}]_k) = \mathbf{Seq}\big). \tag{7}$$

Completing the induction. Combining Eq. (5) of Case 1 and Eq. (7) of Case 2, for every equivalence class $[\mathcal{U}]_k$ we have

$$\mathbb{E}_{\mathcal{U}' \in [\mathcal{U}]_k}\Big[\Pr\big(\text{Rheo}([\mathcal{U}']_{k+1}) = \mathbf{Seq}\big)\Big] \;=\; \Pr\big(\text{Rheo}([\mathcal{U}]_k) = \mathbf{Seq}\big). \tag{8}$$

Taking expectation over all $[\mathcal{U}]_k$,

$$\begin{aligned}
\mathbb{E}_{[\mathcal{U}]_{k+1}}\Big[\Pr\big(\text{Rheo}([\mathcal{U}]_{k+1}) = \mathbf{Seq}\big)\Big] \;&=\; \mathbb{E}_{[\mathcal{U}]_k}\Big[\mathbb{E}_{\mathcal{U}' \in [\mathcal{U}]_k}\big[\Pr(\text{Rheo}([\mathcal{U}']_{k+1}) = \mathbf{Seq})\big]\Big] \\
&=\; \mathbb{E}_{[\mathcal{U}]_k}\Big[\Pr\big(\text{Rheo}([\mathcal{U}]_k) = \mathbf{Seq}\big)\Big] \\
&=\; p(\mathbf{Seq}),
\end{aligned} \tag{9}$$

where the last equality is the induction hypothesis (4). Thus Eq. (3) holds for $R = k + 1$.

By mathematical induction, Theorem 6 holds for every positive integer R. Further, Theorem 1 holds and RheoSampling is lossless. □

C.4 Deferred Proof of Lemma 7 (Slot Determinism)

Proof of Lemma 7. We prove the three statements in order.

Step 1: Existence of the $(k+1)$-th slot in every $\mathcal{U} \in [\mathcal{U}]_k$. Fix $\mathcal{U}' \in [\mathcal{U}]_k$ and let $u' = \mathrm{Prune}_{k+1}(\mathcal{U}') \setminus \mathrm{Prune}_k(\mathcal{U}')$. Let P' be the parent of u'. If P' is the root, it is expanded in every tree and the slot of u' trivially exists; otherwise $P' \in \mathrm{Prune}_k(\mathcal{U}') = T_k$ by Lemma 5. Because every $\mathcal{U} \in [\mathcal{U}]_k$ shares the same T_k, it follows that $P' \in \mathrm{Prune}_k(\mathcal{U})$ for all $\mathcal{U} \in [\mathcal{U}]_k$.

We claim that P' is expanded in every $\mathcal{U} \in [\mathcal{U}]_k$. Suppose not: there exists some $\mathcal{U} \in [\mathcal{U}]_k$ in which P' is *not expanded*. At depth $\ell = \mathrm{depth}(P')$, the tree-construction policy expands exactly the K nodes with highest path scores. Since P' is expanded in $\mathcal{U}'$ (because its child u' exists), it belongs to the top-K at depth ℓ in $\mathcal{U}'$. Since P' is *not expanded* in $\mathcal{U}$, there must be K other nodes at depth ℓ in $\mathcal{U}$ which are higher priority than P'. Because $P' \in T_k$ and T_k collects the globally highest k nodes in $\mathcal{U}$, any node with higher priority than P' must also belong to T_k. Thus these K nodes all lie in $T_k = \mathrm{Prune}_k(\mathcal{U}')$, and therefore appear at depth ℓ in $\mathcal{U}'$ as well, with the same higher priority. Consequently, in $\mathcal{U}'$ the node P' is ranked below at least K nodes at depth ℓ and cannot be expanded. This creates a contradiction since P' has a child u' in $\mathcal{U}'$. Hence P' is expanded in every $\mathcal{U} \in [\mathcal{U}]_k$, and the child slot occupied by u' exists in all of them.

Step 2: Score invariance and uniqueness. Fix any $\mathcal{U} \in [\mathcal{U}]_k$, let u be its $(k+1)$-th node, with parent P, which is either the root or a node of T_k. In both cases, its path score $S_{T_k}(P)$ is frozen by T_k (the root has score 1) and is therefore identical across $[\mathcal{U}]_k$. Let $\mathrm{slot}(u)$ be the corresponding slot of u in $\mathcal{U}$, then we inspect the three possible slot types of $\mathrm{slot}(u)$ to prove its path score $S_{\mathcal{U}}(\mathrm{slot}(u))$ is also frozen by T_k:

Top-m slot. The token $x_i \to \mathrm{slot}(u)$ and its draft probability $q(x_i)$ are deterministic functions of the parent. Hence the slot score $S_{\mathcal{U}}(\mathrm{slot}(u)) = S_{T_k}(P) \cdot q(x_i)$ is a constant over $[\mathcal{U}]_k$.

Sampled slot. The proxy probability $q_{\mathrm{proxy}}(Y) = \min\{q(x_m), z\}$ depends only on the top-m probabilities and the residual mass, all of which are fixed before sampling. Thus the slot score

$S_{\mathcal{U}}(\mathrm{slot}(u)) = S_{T_k}(P) \cdot q_{\mathrm{proxy}}(Y)$ is likewise constant over all $\mathcal{U} \in [\mathcal{U}]_k$, independent of the realized token Y.

Fill slot. By Lemma 4, the sampled sibling always has proxy score at least $q(x_{m+1})$, dominating every fill token including u. Therefore the sampled sibling of u must already belong to T_k, which freezes its realized token and hence the remaining vocabulary is deterministic. Consequently the fill token and its slot score $S_{\mathcal{U}}(\mathrm{slot}(u))$ are identical across the whole equivalence class $\mathcal{U} \in [\mathcal{U}]_k$.

Now fix $\mathcal{U}, \mathcal{U}' \in [\mathcal{U}]_k$ and let u, u' be their respective $(k+1)$-th nodes. By Step 1, the slot of u exists in $\mathcal{U}'$ and the slot of u' exists in $\mathcal{U}$. Let σ and σ' denote these two slots, respectively. Then we create a particular tree $\mathcal{T} = T_k \cup \{\sigma, \sigma'\}$ with tokens $u \to \sigma$ and $u' \to \sigma'$. In $\mathcal{U}$, since u is the $(k+1)$-th node, slot σ' cannot outrank σ; hence path score $S_{\mathcal{U}}(\sigma') \le S_{\mathcal{U}}(\sigma)$ and then $S_{\mathcal{T}}(\sigma') \le S_{\mathcal{T}}(\sigma)$. Symmetrically, in $\mathcal{U}'$, we have $S_{\mathcal{U}'}(\sigma) \le S_{\mathcal{U}'}(\sigma')$ and then $S_{\mathcal{T}}(\sigma) \le S_{\mathcal{T}}(\sigma')$. Therefore $S_{\mathcal{T}}(\sigma') = S_{\mathcal{T}}(\sigma)$ in tree $\mathcal{T}$. Because the global tie-breaking rule (depth first, then left-to-right) is deterministic and fixed, the same slot wins the $(k+1)$-th rank in these three trees $\mathcal{U}, \mathcal{U}'$ and $\mathcal{T}$. Hence $\sigma = \sigma'$, establishing topological invariance (1).

Step 3: Deterministic consistency (2). If the slot type of u' is top-m or fill, Step 2 shows that its token and score are both frozen by T_k. Hence $\mathrm{Prune}_{k+1}(\mathcal{U}')$ is identical for every $\mathcal{U}' \in [\mathcal{U}]_k$.

Step 4: Stochastic fidelity (3). If the slot of u' is sampled, its proxy score is constant (Step 2), while the realized token Y is drawn from $\tilde{q}$ during tree construction. Moreover, T_k contains no node whose token or score depends on Y: such nodes are the fill siblings of this slot and the descendants of this slot, all of which rank below the slot itself (Lemma 4, Lemma 5, and the tie-breaking rule) and hence lie outside T_k; and, as in Step 1, whether a node of T_k is expanded is determined only by higher-priority nodes, which also lie in T_k. Thus membership in $[\mathcal{U}]_k$ does not depend on Y, and since Y is sampled independently of the remaining randomness that determines T_k, conditioning on $[\mathcal{U}]_k$ leaves $Y \sim \tilde{q}$. Because the survival of this slot in the global top-$(k+1)$ depends only on the constant proxy score and not on realized sample token u', the equivalence class $[\mathcal{U}]_k$ is partitioned into subclasses $[\mathcal{U}]_k^{(v)} = \{\mathcal{U}' \in [\mathcal{U}]_k \mid u' = v\}$ whose proportions are exactly $\tilde{q}(v)$. □

C.5 Numerical Verification on an Enumerable Tree

Downstream accuracy parity is not a direct test of distributional fidelity, and sequence-level divergences cannot be estimated reliably on real LLM outputs. We therefore complement the proof with a controlled finite-state experiment in which both the exact target sequence distribution and every stochastic tree realization can be enumerated.

Setup. We use a vocabulary $\{A, B, C, D, E\}$, $K = 3$, $m = 1$, draft depth 2, and a global pruning budget of $R = 6$ out of the $3 + 3 \times 3 = 12$ unpruned nodes. The draft and target distributions at the root and after each first token are listed in Table 6. At every parent, the candidate pool consists of the top-1 token, one token sampled from $\tilde{q}$, and the highest-probability remaining fill token; the sampled slot receives $q_{\mathrm{proxy}} = \min\{q(x_1), z\}$. All three first-layer nodes are expanded by the same rule, with independent tail samples, after which global top-6 pruning is applied with the tie-breaking rule of Appendix C.1. Algorithm 1 is then executed layer by layer. If the first-layer output is not a node of the pruned tree, or the accepted node has no surviving child, the second token is drawn from $p(\cdot \mid \text{first token})$, which is equivalent to starting a fresh lossless round at that prefix.

Table 6: Draft (q) and target (p) distributions of the toy experiment. The first row is the root distribution; each subsequent row is conditioned on the first token.

Prefix	q					p				
	A	B	C	D	E	A	B	C	D	E
$\varnothing$	0.40	0.25	0.15	0.12	0.08	0.20	0.10	0.30	0.25	0.15
A	0.42	0.24	0.16	0.11	0.07	0.18	0.12	0.30	0.25	0.15
B	0.38	0.27	0.14	0.13	0.08	0.25	0.10	0.20	0.30	0.15
C	0.34	0.29	0.18	0.11	0.08	0.12	0.28	0.18	0.22	0.20
D	0.46	0.20	0.15	0.12	0.07	0.30	0.08	0.24	0.18	0.20
E	0.36	0.23	0.19	0.13	0.09	0.16	0.20	0.14	0.28	0.22

Procedure. Each pool admits 4 possible tail samples, so the four pools (root and three first-layer nodes) yield $4^4 = 256$ tree realizations, each weighted by the product of its $\tilde{q}$ probabilities. For every realization we integrate the outcomes of Algorithm 1 analytically in exact rational arithmetic, which gives the exact induced distribution $\hat{p}$ over all 25 length-two sequences; we compare it with the autoregressive target $p(a)\,p(b \mid a)$. As a diagnostic control, we repeat the computation with the invalid identity-dependent proxy $q_{\mathrm{proxy}} = q(Y)$. For reference, we also run 10^5 Monte Carlo trials of both the full RheoSampling pipeline and vanilla target sampling, with fixed seeds 20261024 (vanilla) and 20261025 (RheoSampling); these Monte Carlo rows vary at the 10^{-3} level across seeds. A reference implementation is provided in the supplementary material.

Table 7: Sequence-level fidelity on the enumerable toy tree ($R = 6$).

Method	TV	$\mathrm{KL}(\hat{p} \,\|\, p)$	Max error
Exact RheoSampling	0	0	0
Monte Carlo, vanilla target sampling	8.09×10^{-3}	2.08×10^{-4}	2.06×10^{-3}
Monte Carlo, RheoSampling	7.82×10^{-3}	2.05×10^{-4}	1.76×10^{-3}
Exact, invalid proxy $q(Y)$	3.52×10^{-2}	1.41×10^{-2}	2.11×10^{-2}

Results. As shown in Table 7, the exact distribution produced by RheoSampling coincides with the target as rational numbers on all 25 sequences, and its Monte Carlo deviation is comparable to that of vanilla target sampling under the same budget. The same exact equality holds for every budget $R \in \{1, \ldots, 12\}$. In contrast, the invalid proxy $q(Y)$ incurs a clear systematic bias for every $2 \le R \le 11$; the bias vanishes only in the degenerate cases $R = 1$ (only the top-1 node survives) and $R = 12$ (no pruning), where survival is trivially independent of Y. This confirms that the experiment is sensitive to violations of the losslessness condition, and that the identity-independent proxy of Lemma 4 is what preserves the target distribution under global pruning.

D The Superiority of RheoVerification

D.1 Single-layer Rheo-acceptance Rate (Proof of Theorem 3)

Setup. Let $x_1, x_2, \ldots$ be the vocabulary sorted by draft probability q in descending order. A candidate pool of size n consists of:

- top-m deterministic tokens $\{x_1, \ldots, x_m\}$;
- one sampled token $Y \sim \tilde{q}$, where $\tilde{q}(v) = q(v)/z$ for $v \notin \{x_1, \ldots, x_m\}$ and $z = 1 - \sum_{i=1}^{m} q(x_i)$;
- $n - m - 1$ fill tokens, i.e. the highest-q tokens from $\mathcal{V} \setminus \{x_1, \ldots, x_m, Y\}$.

Throughout this section (including Appendix D.2), $\Pr[\text{output} = v]$ denotes the probability that v is output as an *accepted candidate*, excluding the final fallback sample of Algorithm 1; the acceptance rate is the sum of these probabilities over v.

Always-present tokens. The first $n-1$ tokens $\mathrm{Top}_{n-1} = \{x_1, \ldots, x_{n-1}\}$ appear in *every* realization of the pool:

- $x_1, \ldots, x_m$ are deterministic;
- each x_j ($m + 1 \le j \le n - 1$) is either the sampled token (if $Y = x_j$) or a fill token (if $Y \ne x_j$).

Output probability of $v \in \mathrm{Top}_{n-1}$. For any $v \in \mathrm{Top}_{n-1}$, Algorithm 1 outputs v via two paths:

$$\begin{aligned}\Pr[\text{output} = v] &= \underbrace{\tilde{q}(v)\min\Big(1, \frac{p(v)}{\tilde{q}(v)}\Big)}_{Y=v,\text{ direct accept}} \;+\; \underbrace{r(v)\sum_{t\neq v}\tilde{q}(t)\max\Big(0, 1-\frac{p(t)}{\tilde{q}(t)}\Big)}_{Y\neq v,\text{ reject } Y \text{ then hit } v \text{ under } r} \\ &= \min\big(p(v), \tilde{q}(v)\big) \;+\; \frac{\max\big(0, p(v)-\tilde{q}(v)\big)}{Z}\cdot\big(Z - \max(0, \tilde{q}(v)-p(v))\big) \\ &= p(v), \end{aligned} \tag{10}$$

where $Z = \sum_w \max(0, \tilde{q}(w)-p(w)) = \sum_w \max(0, p(w)-\tilde{q}(w))$ and $r = \mathrm{norm}(\max(0, p-\tilde{q}))$. The last equality follows from the identity $\min(a,b) + \max(0, a-b) = a$.

Summing Eq. (10) over $v \in \mathrm{Top}_{n-1}$ yields the first term of Theorem 3.

Output probability of $v \notin \mathrm{Top}_{n-1}$. For such v, two cases arise according to whether $v = x_n$.

Case 3a: $v = x_n$. This token enters the pool as a fill token *iff* $Y \in \{x_{m+1}, \ldots, x_{n-1}\}$ (as the last fill token), and as the sampled token iff $Y = x_n$; otherwise it is absent. Its output probability therefore decomposes as

$$\begin{aligned}\Pr[\text{output} = x_n] &= \underbrace{\tilde{q}(x_n)\min\Big(1, \frac{p(x_n)}{\tilde{q}(x_n)}\Big)}_{\text{sampled path}} \;+\; \underbrace{r(x_n)\sum_{j=m+1}^{n-1}\tilde{q}(x_j)\max\Big(0, 1-\frac{p(x_j)}{\tilde{q}(x_j)}\Big)}_{\text{fill path}} \\ &= \min\big(p(x_n), \tilde{q}(x_n)\big) \;+\; r(x_n)\sum_{v\in\mathrm{Top}_{n-1}}\max\big(0, \tilde{q}(v)-p(v)\big). \end{aligned} \tag{11}$$

Case 3b: $v \notin \mathrm{Top}_n$. Such a token can only appear as the sampled token $Y = v$, and it never survives into the fill set because the fill slots are exhausted by $\mathrm{Top}_{n-1} \cup \{x_n\} \setminus \{Y\}$. Hence

$$\Pr[\text{output} = v] = \tilde{q}(v)\min\Big(1, \frac{p(v)}{\tilde{q}(v)}\Big) = \min\big(p(v), \tilde{q}(v)\big). \tag{12}$$

Summation. Adding Eqs. (10), (11), and (12) over their respective token sets gives

$$\mathcal{A}_{\text{Rheo}} \;=\; \sum_{v\in\mathrm{Top}_{n-1}} p(v) \;+\; \sum_{v\notin\mathrm{Top}_{n-1}} \min\big(p(v), \tilde{q}(v)\big) \;+\; r(x_n)\sum_{v\in\mathrm{Top}_{n-1}}\max\big(0, \tilde{q}(v)-p(v)\big), \tag{13}$$

which establishes the equality of (2). The inequality of (2) is trivial since

$$\sum_{v\in\mathrm{Top}_{n-1}}\max\big(0, \tilde{q}(v)-p(v)\big) \le Z \quad \text{and} \quad p(x_n) = \min\big(p(x_n), \tilde{q}(x_n)\big) + r(x_n)\cdot Z.$$

□

D.2 Single-layer RRSw-acceptance Rate

Setup (RRSw baseline). The candidate pool is identical to that in Theorem 3, but the verification order is changed. The sampled token $Y \sim \tilde{q}$ is evaluated *after* the deterministic top-m tokens and *before* fill tokens.

Let $\tilde{p}$ be the renormalized restriction of p to the complement of $\mathrm{Top}_m = \{x_1, \ldots, x_m\}$. The verifier first performs sequential rejection over Top_m under p; if all are rejected, it tests Y under $\tilde{p}$; if Y is also rejected, the remaining fill token x_n is tested under $r_{\mathrm{RRSw}} = \mathrm{norm}(\max(0, \tilde{p}-\tilde{q}))$. Trivially, sequential rejection over $\{x_1, \ldots, x_m\}$ under the original target p recovers

$$A := \sum_{i=1}^{m} p(x_i) = \sum_{v\in\mathrm{Top}_m} p(v). \tag{14}$$

Always-present tail tokens. For each $v \in \{x_{m+1}, \ldots, x_{n-1}\}$, two paths lead to acceptance:

$$\begin{aligned}\Pr[\text{output} = v] &= (1-A) \cdot \Bigg[\underbrace{\tilde{q}(v) \min\Big(1, \frac{\tilde{p}(v)}{\tilde{q}(v)}\Big)}_{Y=v,\ \text{direct accept}} + \underbrace{r_{\mathrm{RRSw}}(v) \sum_{t \neq v} \tilde{q}(t) \max\Big(0, 1 - \frac{\tilde{p}(t)}{\tilde{q}(t)}\Big)}_{Y \neq v,\ \text{reject } Y \text{ then hit } v \text{ under } r_{\mathrm{RRSw}}} \Bigg] \\ &= (1-A) \left[\min\big(\tilde{p}(v), \tilde{q}(v)\big) + \frac{\max\big(0, \tilde{p}(v) - \tilde{q}(v)\big)}{\tilde{Z}} \big(\tilde{Z} - \max\big(0, \tilde{q}(v) - \tilde{p}(v)\big)\big) \right] \\ &= (1-A) \cdot \tilde{p}(v) = p(v), \end{aligned} \tag{15}$$

where $\tilde{Z} = \sum_w \max(0, \tilde{q}(w) - \tilde{p}(w)) = \sum_w \max(0, \tilde{p}(w) - \tilde{q}(w))$.

Together with Eq. (14), the first $n-1$ tokens contribute $\sum_{v \in \mathrm{Top}_{n-1}} p(v)$.

Output probability of $v \notin \mathrm{Top}_{n-1}$. For such v, two cases arise according to whether $v = x_n$.

Case 3a: $v = x_n$. This token enters the pool as a fill token iff $Y \in \{x_{m+1}, \ldots, x_{n-1}\}$ (as the last fill token), and as the sampled token iff $Y = x_n$. Conditioned on the prefix Top_m being rejected (probability $1 - A$), its output probability therefore decomposes as

$$\begin{aligned}\Pr[\text{output} = x_n] &= (1-A) \left[\underbrace{\min\Big(\tilde{q}(x_n), \tilde{p}(x_n)\Big)}_{\text{sampled path}} + \underbrace{r_{\mathrm{RRSw}}(x_n) \sum_{j=m+1}^{n-1} \tilde{q}(x_j) \max\Big(0, 1 - \frac{\tilde{p}(x_j)}{\tilde{q}(x_j)}\Big)}_{\text{fill path}} \right] \\ &= (1-A) \left[\min\big(\tilde{p}(x_n), \tilde{q}(x_n)\big) \ + \ r_{\mathrm{RRSw}}(x_n) \sum_{v \in \mathrm{Top}_{n-1}} \max\big(0, \tilde{q}(v) - \tilde{p}(v)\big) \right]. \end{aligned} \tag{16}$$

Case 3b: $v \notin \mathrm{Top}_n$. Such a token can only appear as the sampled token $Y = v$. Therefore

$$\Pr[\text{output} = v] = (1-A)\tilde{q}(v) \min\Big(1, \frac{\tilde{p}(v)}{\tilde{q}(v)}\Big) = (1-A) \min\big(\tilde{p}(v), \tilde{q}(v)\big). \tag{17}$$

Summation. Adding the contributions from Eqs. (14), (15), (16) and (17) yields

$$\begin{aligned}\mathcal{A}_{\mathrm{RRSw}} = &\sum_{v \in \mathrm{Top}_{n-1}} p(v) + \sum_{v \notin \mathrm{Top}_{n-1}} \min\big(p(v), (1-A)\tilde{q}(v)\big) \\ &+ (1-A) \cdot r_{\mathrm{RRSw}}(x_n) \sum_{v \in \mathrm{Top}_{n-1}} \max\big(0, \tilde{q}(v) - \tilde{p}(v)\big), \end{aligned} \tag{18}$$

where $r_{\mathrm{RRSw}} = \mathrm{norm}(\max(0, \tilde{p} - \tilde{q}))$ and $A = \sum_{v \in \mathrm{Top}_m} p(v)$.

D.3 Approximate Comparison: Rheo vs. RRSw

Approximation assumption. In both acceptance-rate formulas, the margin-token terms involve a finite sum over Top_{n-1}:

$$S_{\mathrm{Rheo}} = \sum_{v \in \mathrm{Top}_{n-1}} \max\big(0, \tilde{q}(v) - p(v)\big), \qquad S_{\mathrm{RRSw}} = \sum_{v \in \mathrm{Top}_{n-1}} \max\big(0, \tilde{q}(v) - \tilde{p}(v)\big).$$

Because $\tilde{q}$ is dominated by its top-ranked entries (it is a renormalized tail of the draft distribution), the omitted tail contribution is negligible. We therefore approximate

$$S_{\mathrm{Rheo}} \approx Z := \sum_v \max\big(0, \tilde{q}(v) - p(v)\big), \qquad S_{\mathrm{RRSw}} \approx \tilde{Z} := \sum_v \max\big(0, \tilde{q}(v) - \tilde{p}(v)\big). \tag{19}$$

Case 3a under approximation. Plugging Eq. (19) into the margin-token terms:

$$\begin{aligned}
\text{Rheo:}\quad & \min\big(p(x_n), \tilde{q}(x_n)\big) + r(x_n) S_{\text{Rheo}} \\
& \approx \min\big(p(x_n), \tilde{q}(x_n)\big) + \max\big(0, p(x_n) - \tilde{q}(x_n)\big) \;=\; p(x_n), \\
\text{RRSw:}\quad & (1-A)\Big[\min\big(\tilde{p}(x_n), \tilde{q}(x_n)\big) + r_{\text{RRSw}}(x_n) S_{\text{RRSw}}\Big] \\
& \approx (1-A)\Big[\min\big(\tilde{p}(x_n), \tilde{q}(x_n)\big) + \max\big(0, \tilde{p}(x_n) - \tilde{q}(x_n)\big)\Big] \;=\; p(x_n).
\end{aligned}$$

Thus, under the approximation, the margin-token contributions are equal.

Case 3b (strict inequality). For every $v \notin \text{Top}_n$, the tail-overlap terms satisfy

$$\min\big(p(v), \tilde{q}(v)\big) \;\geq\; \min\big(p(v), (1-A)\tilde{q}(v)\big),$$

with strict inequality whenever $(1-A)\tilde{q}(v) < p(v) < \tilde{q}(v)$.

Net difference. Collecting the two cases, the approximate acceptance rates become

$$\begin{aligned}
\mathcal{A}_{\text{Rheo}} &\approx \sum_{v \in \text{Top}_{n-1}} p(v) + p(x_n) + \sum_{v \notin \text{Top}_n} \min\big(p(v), \tilde{q}(v)\big), \\
\mathcal{A}_{\text{RRSw}} &\approx \sum_{v \in \text{Top}_{n-1}} p(v) + p(x_n) + \sum_{v \notin \text{Top}_n} \min\big(p(v), (1-A)\tilde{q}(v)\big).
\end{aligned}$$

Hence

$$\mathcal{A}_{\text{Rheo}} - \mathcal{A}_{\text{RRSw}} \;\approx\; \sum_{v \notin \text{Top}_n} \Big[\min\big(p(v), \tilde{q}(v)\big) - \min\big(p(v), (1-A)\tilde{q}(v)\big)\Big] \;\geq\; 0.$$

The gap is strictly positive as soon as there exists any tail token v with $(1-A)\tilde{q}(v) < p(v) < \tilde{q}(v)$. In other words, **under this approximation, stochastic-first improves acceptance by preserving the full tail-draft overlap**, whereas sequential RRSw compresses the overlap by the prefix mass A. When the approximation is loose (i.e., $\tilde{q}$ places substantial mass outside Top_{n-1}), the margin-token terms no longer cancel, and the dominance is not guaranteed.

E Example of Dynamic Tree Construction

We illustrate the EAGLE-style expand-then-rerank procedure of Section 2.1 with expansion width $K = 10$, depth $D = 8$, and global budget $N = 60$ (the default setting in Section 5). The root first produces 10 candidate tokens at the first layer. Each of these nodes is expanded into 10 children, yielding 100 candidates at the second layer, among which the 10 with the highest path scores are selected for further expansion; here the path score of a node is the product of draft probabilities along its root-to-node path. This process repeats until the eighth layer, producing $10 + 7 \times 100 = 710$ candidate nodes in total. Finally, global reranking retains the 60 highest-scoring nodes (including the root) across all layers; by Lemma 5, the retained set is ancestor-closed and hence forms a valid tree.

Selection for expansion is distinct from final retention: a node that is not expanded may still survive global reranking, and an expanded node may still have all its children discarded. Thus, the candidates generated during local expansion do not yet constitute the final verification tree. In RheoSampling, each local candidate pool consists of the lead, sampled, and fill tokens. The fill tokens are deterministic candidates rather than additional samples from the tail distribution $\tilde{q}$, and the sampled token enters the path scores through its proxy probability q_{proxy} rather than its draft probability.

F Broader Impact

This paper proposes RheoSampling, a method that injects stochasticity into dynamic-tree speculative decoding to improve acceptance rates and speedup. Our contribution mainly lies in the decoding algorithm, which may reduce computational cost and energy consumption of LLM inference, facilitating deployment on resource-constrained devices. LLMs themselves may lead to various societal impacts, which this work does not directly contribute to.

On the technical side, the equivalence-class analysis developed in this work provides a principled proof of losslessness for stochastic dynamic trees. The technique of decoupling randomness via frozen proxy scores may serve as a useful perspective for analyzing other problems with complex coupled randomness, though its broader applicability remains to be examined.

G Limitations

While RheoSampling successfully injects stochasticity into dynamic-tree speculative decoding and provides rigorous lossless guarantees, several limitations merit discussion.

Engineering overhead at low temperatures. When the sampling temperature T approaches zero, the target distribution becomes increasingly sharp, and the practical benefit of stochastic exploration diminishes. Even with $m = 0$, RheoSampling still pays the overhead of sampling, proxy assignment, and OT-based verification that pure deterministic top-K expansion avoids. In greedy or low-temperature decoding, where the acceptance-length advantage becomes negligible, this extra cost is wasted. For such settings, standard top-K dynamic trees remain the more efficient choice, and RheoSampling is not a universal replacement.

Sensitivity to the rheostat parameter m. The hyperparameter m controls the number of deterministic lead tokens and critically determines whether the stochastically sampled token survives global pruning. While $m = 0$ and $m = 1$ were empirically identified as two attractive operating points, the optimal choice depends on the draft-target alignment and the temperature, and a universally optimal setting does not exist. This means m may require light tuning when deploying RheoSampling on a new model or task, which limits plug-and-play generality. An adaptive strategy for selecting m would help address this in future work.

Analytical gap in dominance over RRSw. In Appendix D.3, we provide an approximate comparison showing that RheoVerification dominates the recursive rejection sampling without replacement (RRSw) baseline. This derivation relies on the approximation that the finite sum over the top-$(n-1)$ tokens captures nearly all the residual mass, and the dominance is not guaranteed when this approximation is loose. Although the empirical evidence (Table 4 and the end-to-end results) consistently favors RheoVerification in realistic settings, a precise characterization of the conditions under which RheoVerification dominates RRSw remains an open question.

H Use of LLMs

The authors use AI assistants to polish and assist the implementation of the methods. All content generated by AI is carefully reviewed by the authors. The authors are responsible for all AI generated content, including codes and texts.

I Licenses

This paper utilizes various models, datasets, and open-source codes. For models, Vicuna operates under a non-commercial research license (derived from LLaMA-1), Llama 3.1 is subject to the Llama 3.1 Community License, and DeepSeek-R1-Distill-Llama is released under the MIT license while remaining subject to Llama 3.1 derivative terms. Regarding datasets, Alpaca is licensed under CC-BY-NC-4.0, MT-bench under CC-BY-4.0, while HumanEval and GSM8K are under the MIT license. Natural Questions follows the CC-BY-SA license (inheriting from Wikipedia), and CNN/DM consists of news articles utilized for research purposes. The foundational codebase is built upon the EAGLE repository, which is licensed under Apache-2.0.

J Computing Resources

All experiments were conducted using NVIDIA RTX A6000 (48GB) GPUs. Each independent evaluation run across the six datasets requires approximately 30 to 60 minutes on a single GPU. We estimate the total computational budget for all experiments, including ablations and baselines, to be approximately 300 GPU hours.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We confirm that the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: See Section 4.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See Section 5 for experimental details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The code cannot be released due to corporate intellectual property restrictions. All datasets and models used are publicly available, and Section 5 and the appendix provide the algorithmic and experimental details needed to reproduce the main results. A standalone script reproducing the numerical verification in Appendix C.5 is provided in the supplementary material.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: See Section 5 for experimental setup.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We include the standard deviation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Section 5 and Appendix J for device information.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We are sure the research is conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Appendix F.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: See Appendix I.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new datasets, models, or codebases. The supplementary material only contains a small script for the toy numerical verification in Appendix C.5.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: See Section H.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.