TALON: Confidence-Aware Speculative Decoding with Adaptive Token Trees

Anonymous ACL submission

Abstract

Speculative decoding (SD) has become a standard technique for accelerating LLM inference without sacrificing output quality. Recent advances in speculative decoding have shifted from sequential chain-based drafting to tree-structured generation, where the draft model constructs a tree of candidate tokens to explore multiple possible drafts in parallel. However, existing tree-based SD methods typically build a **fixed-width, fixed-depth** draft tree, which fails to adapt to the varying difficulty of tokens and contexts. As a result, the draft model cannot dynamically adjust the tree structure to early stop on difficult tokens and extend generation for simple ones. To address these challenges, we introduce **TALON**, a ***training-free, budget-driven*** adaptive tree expansion framework that can be plugged into existing tree-based methods. Unlike static methods, **TALON** constructs the draft tree iteratively until a fixed token budget is met, using a hybrid expansion strategy that adaptively allocates the node budget to each layer of the draft tree. This framework naturally shapes the draft tree into a **"deep-and-narrow"** form for deterministic contexts and a **"shallow-and-wide"** form for uncertain branches, effectively optimizing the trade-off between exploration width and generation depth under a given budget. Extensive experiments across 5 models and 6 datasets demonstrate that **TALON** consistently outperforms state-of-the-art EAGLE-3, achieving up to 5.16× end-to-end speedup over auto-regressive decoding.

1 Introduction

While Large Language Models (LLMs) have achieved remarkable success in various benchmarks (OpenAI Team, 2025; Google DeepMind Team, 2025; Anthropic Team, 2025; QwenTeam et al., 2025; DeepSeek-AI et al., 2025; Team et al., 2025), their deployment is severely constrained by their auto-regressive token-by-token generation.

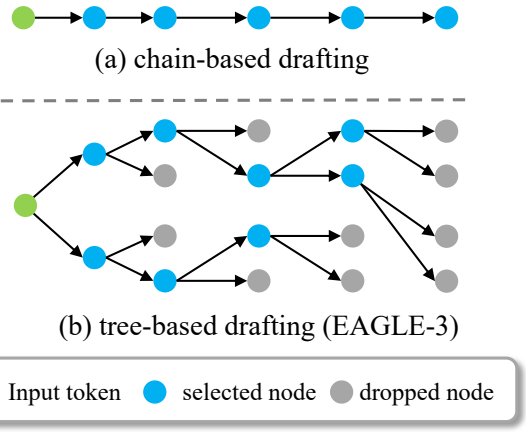


Figure 1: Illustration of chain-based drafting and tree-based drafting (EAGLE-3 (Li et al., 2025a)) with $K = 2$. At each step, EAGLE calls draft model forward on the selected K parent nodes of last step, selects top-K child nodes for each parent, and filters $K \times K$ child nodes. Then EAGLE employs an additional top-K operation to choose K nodes as parents for next step.

This sequential dependency prevents models from predicting multiple tokens (Gloeckle et al., 2024) in a single step, causing inference latency to scale linearly with output length (Pope et al., 2022) and making real-time interaction computationally expensive.

To alleviate this limitation, *Speculative Decoding* (SD) (Leviathan et al., 2023; Chen et al., 2023) has emerged as a promising paradigm to break the strict sequential dependency (Liu et al., 2025a). SD decouples each decoding step into two subprocedures: efficient drafting and parallel verification. A lightweight draft model first proposes a short candidate sequence, and the target LLM then validates all proposed tokens in parallel, accepting multiple tokens with a single target model forward.

However, early speculative decoding methods typically employ a *chain-based* drafting strategy, which is inherently vulnerable. A single rejection at the beginning of the draft sequence invalidates all

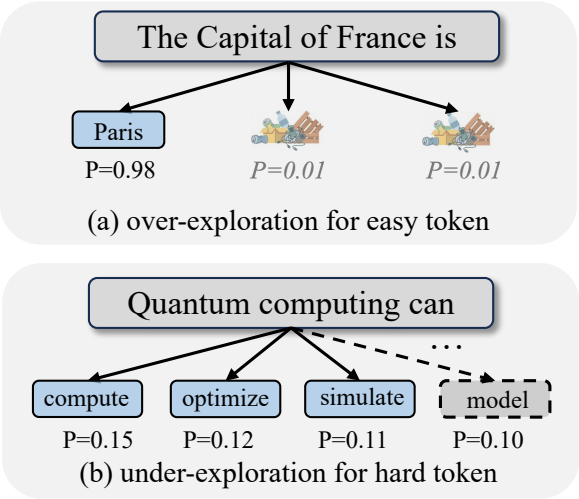


Figure 2: Limitations of tree-based drafting methods. (a) when the model is already confident to its prediction, the draft tree still grows k child nodes. (b) when the model is very confused and highly uncertain, the top-K draft tokens are still not sufficient.

subsequent tokens, leading to a substantial waste of computational resources (Miao et al., 2024). To further improve the overall acceptance rate, recent works have shifted from chain-based to *tree-based drafting* (Miao et al., 2024; Cai et al., 2024; Li et al., 2025b; Du et al., 2024; Li et al., 2024). Instead of predicting a single sequence, these approaches construct a token tree covering multiple plausible continuation paths. By leveraging *Tree Attention* (Yao et al., 2025), the target LLM can verify multiple draft token sequences in parallel within a single forward pass, significantly increasing the draft acceptance rate and overall acceleration.

State-of-the-art tree-based methods, such as EAGLE (Li et al., 2025b), construct the draft tree via a rigid layer-wise expansion mechanism. As illustrated in Figure 1, these approaches iteratively expand a fixed number of child nodes for every parent and select the top-K candidates to proceed to the next layer. However, this strategy enforces a *static tree structure* with pre-determined dimensions. Such rigid allocation fails to adapt to the model's dynamic confidence (output probabilities of its tokens) [1]: it still expands k child nodes even if the draft model is already confident (shown in Figure 2(a)), while it cannot allocate more node budget when the model is confused and highly uncertain (shown in Figure 2(b)). Moreover, it prematurely truncates high-confidence paths that could extend deeper, while squandering the computational budget on expanding low-probability branches in uncertain contexts.

To tackle these inefficiencies, we introduce **TALON**, a novel *training-free, budget-driven* adaptive tree expansion framework to op**T**imize dr**A**ft tree for faster specu**L**ative dec**O**di**N**g. Departing from the rigid fixed width and depth generation, **TALON** employs a dynamic growth algorithm that incrementally expands the draft tree until the number of nodes reaches a given budget N. Specifically, we design a hybrid expansion strategy to optimize the tree topology: at the first layer, we propose a fixed width initialization that utilizes a top-K operation to alleviate the early rejection phenomenon. At subsequent layers, we propose a confidence-gated expansion strategy to adaptively allocate the node budget. This allows the draft structure to evolve to **deep-and-narrow** for deterministic contexts to maximize draft length, or **shallow-and-wide** for uncertain ones to enhance hit rate.

To summarize, our contributions are:

(i) We identify a key limitation in existing tree-based speculative decoding methods: the strict, layer-wise static tree expansion fails to adapt to the model's varying confidence, leading to inefficient utilization of the computational budget.

(ii) We propose **TALON**, a budget-driven speculative decoding framework. By employing a dynamic tree growth algorithm with robust tree initialization and confidence-gated expansion, **TALON** constructs adaptive draft trees that dynamically adapts (deep-and-narrow or shallow-and-wide) based on context uncertainty.

(iii) Extensive experiments verify the effectiveness of **TALON**. It significantly outperforms state-of-the-art method EAGLE-3 in terms of draft efficiency and wall-clock speedup, particularly in scenarios with fluctuating generation difficulty.

2 Related Work

In this section, we review speculative decoding from the perspective of *drafting structures*: chain-based speculative decoding and tree-based specula-

[1] "Dynamic" in EAGLE means that the draft model dynamically select top-K nodes as next-layer parents, but it cannot adjust k to adapt for contexts. We provide an example in Figure 8 in Appendix to further illustrate the difference.

tive decoding. **We position TALON in the context of extensive related works in Appendix F.**

Chain-based speculative decoding. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) introduces a lossless *draft-and-verify* (Zhang et al., 2024a) principle: a cheap drafter proposes a short continuation and a target model verifies it in parallel, accepting the longest matched prefix. This paradigm suffers from a clear efficiency bottleneck—*early rejection* (Sun et al., 2025)—since a mismatch at an early position invalidates all downstream drafted tokens. Follow-up work improves chain-based SD by (i) producing better proposals with minimal overhead (e.g., lightweight head-based drafting that reuses target representations(Cai et al., 2024; Du et al., 2024; Li et al., 2025b)), and (ii) reducing wasted draft computation via adaptive lookahead (Huang et al., 2025) or pipelined scheduling (Liu et al., 2025a). **Nevertheless, the speedup is fundamentally sensitive to the hardest tokens, because one rejection discards the entire suffix.**

Tree-based speculative decoding. Tree-based SD generalizes the draft from a single chain to a *token tree*, allowing the verifier to choose among multiple candidate branches within one forward and thus mitigating early rejection. SpecInfer (Miao et al., 2024) is an early representative that organizes candidates as a token tree and verifies them in parallel with tree attention. More recent work strengthens the tree-based pipeline from two angles: (i) improving tree proposals via well-calibrated drafters (Zhang et al., 2024b; Huo et al., 2025; Gao et al., 2025) and context-aware dynamic draft trees (Xiong et al., 2024), and (ii) optimizing the draft-tree structure under a node budget, including dynamic-programming-based and search-based construction (e.g., Sequoia (Chen et al., 2025) and OPT-Tree (Wang et al., 2024)). However, existing methods typically rely on **rigid or heuristic expansion patterns**, failing to **explicitly allocate a token budget based on real-time confidence.** This gap motivates **TALON**, a training-free, budget-driven framework that constructs adaptive token trees (deep-and-narrow vs. shallow-and-wide).

3 Background

We first formulate *chain-based* drafting and *tree-based* drafting for better understanding.

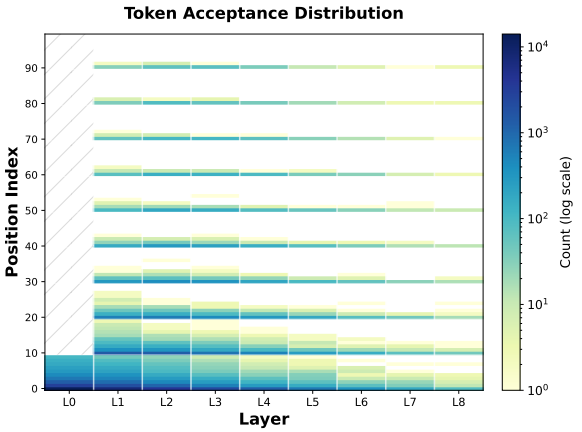


Figure 3: Visualization of token acceptance frequency within a static draft tree ($K = 10, D = 8$). The heatmap reveals an **"Acceptance Funnel"** effect: while the acceptance frequency of the first layer is relatively uniform, the acceptance in subsequent layers ($d \geq 1$) shows a funnel trend that the accepted tokens concentrate more on high confidence regions (e.g., top-1 and top-2), rendering the wide static expansion computationally wasteful. Note that the first layer only has K nodes, while its subsequent layers have $K \times K$ nodes.

3.1 Chain-based Drafting

In standard speculative decoding, a smaller draft model $\mathcal{M}_d$ generates a single sequence of γ tokens (a chain) as a speculation for the target model $\mathcal{M}_t$. Given the prefix $x_{\leq t}$, the drafting process generates a sequence $x_{t+1}, \ldots, x_{t+\gamma}$ auto-regressively:

$$x_{t+i} \sim P_{\mathcal{M}_d}(\cdot|x_{<t+i}), \quad 1 \leq i \leq \gamma \qquad (1)$$

The target model $\mathcal{M}_t$ then verifies this sequence in parallel. The key limitation of this approach is the *sequential dependency* during verification: if the token at position i is rejected, all subsequent tokens $x_{>i}$ are discarded regardless of their correctness, resulting in wasted computation.

3.2 Tree-based Drafting

To mitigate the limitations of chain-based drafting, recent works (e.g., EAGLE) verify a token tree $\mathcal{T}$ to cover diverse paths. As formalized in Algorithm 1 and Figure 1, these methods employ a *static* construction strategy: at each depth d, the model takes a parallel forward on all parent nodes and outputs their next-token distributions. Then, it selects top-K entries from the output distribution of each parent node, acquiring $K \times K$ leaf nodes. After that, the model measures the path score of each layer node v:

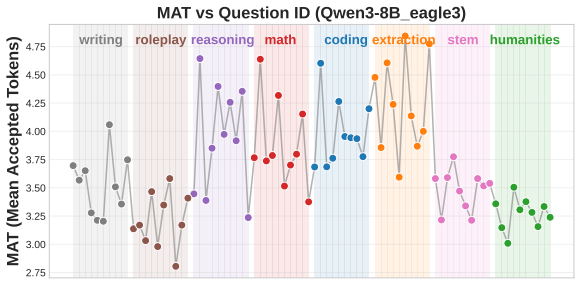


Figure 4: Real-World Mean Accepted Tokens (MAT) distribution across different queries in EAGLE. The results exhibit high volatility: even within the same task category (e.g., Math or Coding), the optimal generation length fluctuates significantly.

$$p(v) = \prod_{j \in \text{Path}(x_t, v)} P_{\mathcal{M}_d}(j \mid x_{<j}) \tag{2}$$

where $\text{Path}(x_t, v)$ represents the path from the root node x_t to the leaf node v, $x_{<j}$ denotes all prefix tokens of j. Then it uses another top-K operation based on the path score to select K leaf nodes as parent nodes of next layer. Finally, the generation ends at depth D, and the tree $\mathcal{T}$ will be pruned to meet a global budget N. While structured, this rigid "expand-then-shrink" mechanism often generates redundant nodes that are discarded during intermediate shrinking or final pruning, leading to suboptimal resource allocation.

4 Motivated Experiments

To empirically investigate the limitations of static tree-based speculative decoding, we conduct a pilot analysis using EAGLE (Li et al., 2025b) as a representative baseline. We employ a fixed tree topology with width $K = 10$ and depth $D = 8$ (official settings in EAGLE paper) with Qwen3-8B on MT-Bench (Zheng et al., 2023). We also conduct additional motivated experiments in Appendix C with Llama-series models and various datasets to demonstrate the generality of our motivation.

4.1 The "Acceptance Funnel" Phenomenon

We first investigate the acceptance frequency of each position in the draft tree. Detailed model configurations are provided in Table 2. Figure 3 visualizes the acceptance frequency of tokens at each position within the static tree structure. Two key observations emerge regarding the *distribution of effective speculation*:

Diminishing Returns of Width in Deep Layers. As illustrated in Figure 3, the acceptance distribution exhibits a funnel-like pattern. In the first initial layer ($d = 0$), the acceptance probability is relatively uniform across the top-K candidates. This suggests that due to the initial stochasticity and minor distributional divergence between the draft model $\mathcal{M}_d$ and the target model $\mathcal{M}_t$, a wider search breadth is necessary to capture the valid continuation. However, as the tree deepens ($d \geq 1$), the acceptance mass concentrates sharply on the high confidence regions (e.g. top-1 and top-2 candidates). We attribute this to the *cumulative error amplification* in auto-regressive drafting: for deep nodes, the draft model is either (1) confidently aligned with the target model (correct prediction), or (2) hallucinating a divergent path where even the top-K candidates fail to recover the target distribution. Consequently, maintaining a static width $K = 10$ at deeper layers yields negligible marginal utility. When the draft model is aligned, a much smaller K can find the correct draft tokens. When the draft model is highly uncertain, there needs a larger K to ensure the coverage and stops the generation to avoid wasteful draft model forward.

4.2 Variance in Real-World Accepted Length

While Section 4.1 reveals the structural redundancy within fixed width, we further investigate the weakness of fixed depth. We plot the distribution of Mean Accepted Tokens (MAT) for various queries in Figure 4. (Same settings with Figure 3)

The Static Depth Dilemma. As shown in Figure 4, the accepted tokens fluctuate drastically even within the same task category (e.g., Math or Coding), exposing the limitation of a fixed-depth drafting policy. In low-entropy contexts where the draft model is well-aligned with the target, the potential acceptance length often exceeds the pre-defined depth D. However, a static fixed depth D prevents $\mathcal{M}_d$ from generating more draft tokens for maximum speedup. Conversely, in high-entropy scenarios where prediction becomes ambiguous, the draft model tends to hallucinate deep branches that are destined for rejection. A fixed depth D forces the allocation of computational resources to these invalid tokens, incurring latency overhead with zero marginal utility.

5 Method

We introduce **TALON**, a budget-driven framework that optimizes draft tree construction by dynamically allocating resources based on real-time model

confidence.

5.1 From Static Grids to Dynamic Budgets

To overcome the inefficiencies of the *static tree structures* analyzed in Section 4–specifically the misalignment between fixed topology and varying token difficulty–**TALON** shifts the constraint from *shape* to *capacity*. We define a global **Token Budget** (N), representing the maximum number of nodes allowed in the draft tree $\mathcal{T}$. The objective is to iteratively "invest" this budget to grow a draft tree that dynamically adapts–*deep-and-narrow* for deterministic contexts and *shallow-and-wide* for uncertain ones–thereby maximizing effective speculation length under a fixed computational cost.

5.2 Hybrid Tree Expansion Strategy

TALON constructs the tree layer-by-layer using a hybrid strategy: a robust initialization at the root (Layer 0) followed by confidence-gated expansion for subsequent layers (Layer $d \geq 1$).

5.2.1 Confidence-Gated Expansion

For depths $d \geq 1$, we employ a global filtering mechanism to gate candidates based on their relative confidence. Let $\mathcal{P}_d$ be the set of parent nodes. We define the **Candidate Pool** $\mathcal{S}_d$ as the union of all child extensions:

$$\mathcal{S}_d = \bigcup_{v \in \mathcal{P}_d} \{(v, w) \mid w \in \mathcal{V}\}. \tag{3}$$

Each candidate $u = (v, w) \in \mathcal{S}_d$ is assigned a cumulative path probability $p(u) = p(v) \cdot P_{\mathcal{M}_d}(w \mid x_{\leq v})$.

Motivated by (Minh et al., 2025), we first identify the **anchor confidence** $m_d = \max_{u \in \mathcal{S}_d} p(u)$. We then retain candidates whose confidence falls within a dynamic margin μ of the anchor:

$$\mathcal{P}_{d+1} = \{u \in \mathcal{S}_d \mid p(u) \geq \mu \cdot m_d\}, \tag{4}$$

where $\mu \in (0, 1]$ is a hyperparameter. To strictly respect the budget, if $|\mathcal{P}_{d+1}|$ exceeds the remaining budget $N - |\mathcal{T}|$, we retain only the top candidates with the highest path scores.

Intuition. This mechanism naturally aligns topology with entropy. In deterministic contexts (e.g., "The capital of France is *Paris*"), the anchor $m_d \approx 1.0$ imposes a strict threshold, automatically pruning branches to form a *deep-and-narrow* chain. In high-entropy contexts (e.g., "Quantum computing can..."), a lower m_d relaxes the threshold, admitting diverse candidates to form a *shallow-and-wide* layer that prioritizes coverage.

5.2.2 Robust Tree Initialization

While confidence gating effectively allocates budget at depth, applying it at the root ($d = 0$) compromises robustness due to draft model ***over-confidence***. Small draft models often exhibit imperfect calibration, assigning near-certain probability (e.g., > 0.99) to incorrect tokens in simple contexts. Relative gating would prematurely collapse the tree into a single wrong branch–an *over-confidence trap*. Unlike deeper errors, a root failure causes catastrophic rejection of the entire draft sequence. To mitigate calibration errors, we employ a **Standard Top-K Initialization** for the first layer. Regardless of the confidence distribution, we explicitly expand the top-K tokens:

$$\mathcal{P}_1 = \{(x_t, v) \mid v \in \text{Top-}K\ (P_{\mathcal{M}_d}(\cdot|x_t))\}. \tag{5}$$

This ensures the speculative tree covers diverse plausible directions at the most critical juncture before switching to budget-efficient gating.

5.3 Evaluating TALON with Draft Efficiency

To rigorously quantify the cost-effectiveness of **TALON**, we analyze the wall-time speedup by decomposing it into quality and cost factors.

Let N_p denote the total number of forward passes executed by the target model (verification steps), N_q be the total forward passes executed by the draft model to generate a sequence and L be the output length. We introduce a new metric, **Draft Efficiency** (δ), defined as the ratio of these two quantities:

$$\delta = \frac{N_q}{N_p}. \tag{6}$$

Intuitively, δ represents the **speculation cost**–the average number of draft steps the system "invests" to secure a single verification opportunity. By deriving the wall-time speedup R as a function of the Mean Accepted Tokens ($\tau = L/N_p$) and the relative model cost (c), we obtain the following speedup formulation (see derivation in Appendix D):

$$R = \frac{\tau}{1 + c \cdot \delta}. \tag{7}$$

With Equation 7, we can theoretically explain **why TALON achieves superior performance**. Approaches like EAGLE enforce a fixed depth D, effectively locking the draft cost to a constant

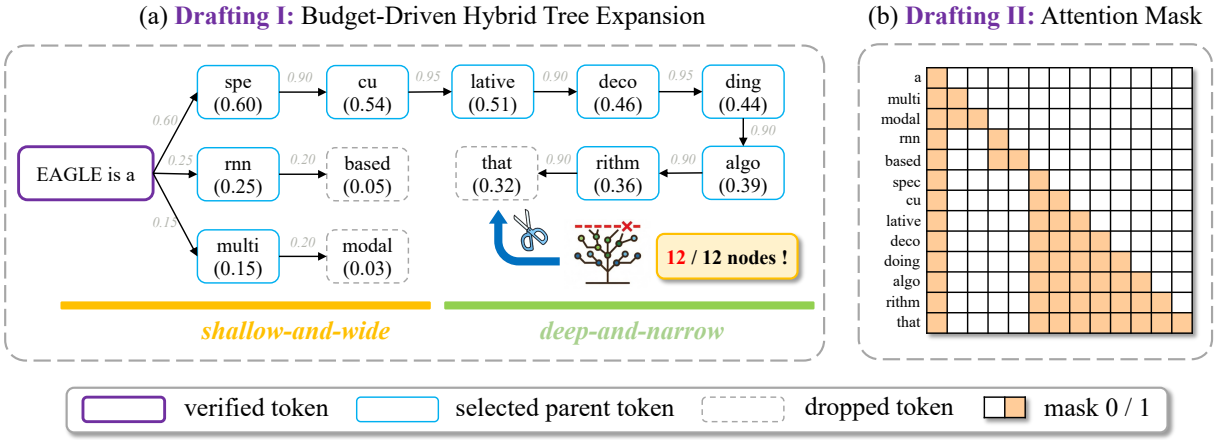


Figure 5: (a) **TALON** 's Budget-Driven Tree. **TALON** dynamically allocates the token budget based on confidence. It uses Top-K at the root for robustness and confidence gating at deeper layers, resulting in adaptive topologies—deep chains for high-confidence tokens (e.g., the folded "algo-rithm" sequence) and wide branches for uncertain ones. The expansion stops when the global budget is met (indicated by the icon). (b) Tree Attention Mask. The structural mask used by the target model to verify the adaptive tree in parallel. The verification process follows the standard token-tree verification protocol (see Appendix B for details).

$\delta = D + 1$ regardless of the context complexity. This rigidity leads to dual inefficiencies.

In **high-entropy contexts** (hard cases), the acceptance reward τ naturally drops. EAGLE continues to pay the high fixed cost δ, causing the speedup R to degrade significantly. **TALON** addresses this by halting expansion early; the resulting reduction in "investment" ($\delta \downarrow$) compensates for the lower reward, thereby mitigating the performance drop. Conversely, in **low-entropy contexts** (easy cases), static methods artificially cap the reward τ at depth D. **TALON** allows the draft tree to extend far beyond this limit, significantly increasing τ. While growing deeper also increases the draft cost δ, this investment yields a positive net gain. Since the relative cost ratio is typically small ($c \ll 1$), $(1 + c\delta)$ grows much slower than τ, ensuring that the overall wall-time speedup R continues to improve as the tree deepens.

6 Experiments

6.1 Experimental Setup

Tasks and Datasets. We evaluate **TALON** on a comprehensive suite of LLM backbones, including Llama-3.1-8B-Instruct (Team, 2024), Qwen3 8B and 32B (QwenTeam et al., 2025), DeepSeek-R1-Distill-LLaMA-8B (DSL) (DeepSeek-AI et al., 2025), and Vicuna-13B (Zheng et al., 2023). Following standard protocols, we conduct evaluations on six diverse benchmarks: MT-Bench (Zheng et al., 2023), Alpaca (Ding et al., 2023), GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), CNN/DM (Nallapati et al., 2016), and QA (Kwiatkowski et al., 2019), which are widely used benchmarks for instruction, math reasoning, code generation, chat, QA and summarization.

Implementation Details. We evaluate **TALON** against the state-of-the-art tree-based method EAGLE-3 (Li et al., 2025b). All experiments are conducted on a single NVIDIA H200 (140GB) GPU with batch size 1 by default. The global token budget N is set to `60` for both **TALON** and EAGLE-3. The threshold μ in **TALON** is set to `0.03` for all experiments. Detailed configurations (hyperparameters, hardware, baselines, models) are provided in Appendix E.1. We use two metrics: mean accepted tokens (MAT) and wall-time speedup (Spd.).

6.2 Main Results

We present the main comparison with EAGLE-3 in Table 1. We further evaluate **TALON** under stochastic sampling ($T = 1$) in Table 3, and provide detailed ablation studies on tree initialization and budget settings in Appendix E.4.

We observe the following: (a) **Universal Speedup.** **TALON** consistently outperforms EAGLE-3 across all 8 models and 6 datasets. Notably, it achieves up to $5.16\times$ speedup on HumanEval with Vicuna-13B and $2.30\times$ speedup on CNN/DM with Qwen3-8B, significantly surpassing the baseline's $4.77\times$ and $1.95\times$. (b) **Reasoning Adaptability.** The performance gap is particularly pronounced in reasoning-intensive tasks. On

Table 1: **Main Results on Six Benchmarks (Temperature=0).** Comparison between EAGLE-3 and our proposed **TALON** across various models. We report Mean Acceptance Tokens (MAT) and Wall-time Speedup relative to standard decoding. **Bold** numbers denote the best speedup performance.

Model	Method	Alpaca		GSM8K		HumanEval		MT-Bench		QA		CNN/DM	
		MAT	Spd.	MAT	Spd.	MAT	Spd.	MAT	Spd.	MAT	Spd.	MAT	Spd.
Vicuna-13B	EAGLE-3	6.61	3.78×	6.74	3.87×	8.31	4.77×	7.12	4.05×	5.24	3.03×	6.93	3.43×
	SD	2.19	1.30×	2.20	1.19×	2.86	1.47×	2.51	1.29×	1.97	1.20×	2.63	1.41×
	MEDUSA	2.44	1.90×	2.63	2.05×	2.78	2.18×	2.58	2.01×	2.10	1.62×	2.09	1.56×
	HYDRA	3.51	2.40×	3.66	2.53×	3.87	2.67×	3.64	2.46×	2.88	1.95×	2.82	1.86×
	OPT-Tree	6.56	3.71×	6.77	3.65×	8.21	4.35×	6.95	3.75×	5.22	3.10×	6.91	3.22×
	TALON	6.36	**3.94**×	6.80	**3.99**×	9.48	**5.16**×	7.29	**4.27**×	5.03	**3.26**×	7.20	**3.58**×
DSL-8B	EAGLE-3	5.64	3.23×	7.40	4.24×	6.70	3.85×	5.82	3.33×	5.02	2.86×	5.03	2.89×
	TALON	5.18	**3.46**×	7.46	**4.43**×	6.28	**3.96**×	5.45	**3.56**×	4.66	**3.12**×	4.75	**3.14**×
Llama3-8B	EAGLE-3	6.93	3.92×	6.42	3.63×	7.08	4.05×	6.38	3.67×	5.36	3.01×	5.46	3.06×
	TALON	6.51	**4.04**×	6.24	**3.81**×	7.28	**4.20**×	6.22	**3.85**×	5.09	**3.27**×	5.28	**3.30**×
Qwen3-8B	EAGLE-3	3.45	2.08×	3.92	2.37×	3.89	2.35×	3.64	2.20×	3.46	2.09×	3.27	1.95×
	TALON	3.39	**2.44**×	3.85	**2.67**×	3.78	**2.69**×	3.60	**2.57**×	3.41	**2.46**×	3.20	**2.30**×
Qwen3-32B	EAGLE-3	2.75	1.83×	3.36	2.22×	2.98	1.96×	2.98	1.90×	2.66	1.78×	2.57	1.57×
	TALON	2.72	**2.05**×	3.30	**2.38**×	2.96	**2.15**×	2.94	**2.10**×	2.64	**1.99**×	2.54	**1.72**×

GSM8K (Math) and HumanEval (Code), **TALON** achieves substantial gains (e.g., 2.67× vs 2.37× on Qwen3-8B GSM8K). This verifies that our budget-driven adaptive expansion effectively captures correct paths in low-entropy reasoning steps where static trees often under-explore. (c) **Robustness.** As shown in Table 3, **TALON** maintains strong performance with $T = 1$. While static trees suffer from flattened distributions, **TALON**'s confidence-gated mechanism successfully adapts the tree topology, achieving 2.51× speedup on Qwen3-8B (HumanEval) compared to EAGLE-3's 2.20×.

6.3 Evaluating TALON with Draft Efficiency

We evaluate the draft efficiency (defined in Section 5.3) of **TALON** and EAGLE and visualize the relationship between the overhead (δ) and the benefit (τ) in Figure 6. Static methods (orange line) pay a fixed high computational cost without considering context difficulty, resulting in sub-optimal speedup. In contrast, **TALON** (blue line) closely approaches the zero-waste Oracle line ($\tau = \delta$). By dynamically shrinking the token trees in uncertain regions and expanding them in deterministic ones, **TALON** effectively decouples draft cost from tree depth. This confirms that our framework maximizes speedup by ensuring computational resources are only invested where they yield high acceptance utility. We also provide more visualization results of different models in Appendix E.2.

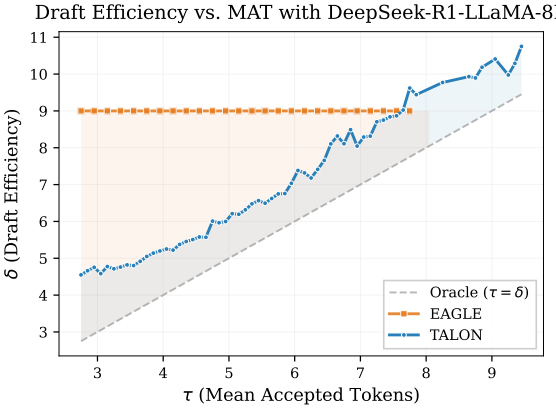


Figure 6: Draft Efficiency (δ) vs. Mean Accepted Tokens (τ). The gray dashed line represents the Oracle baseline ($\tau = \delta$). The orange shaded region highlights the computation waste of static methods (EAGLE-3). **TALON** (blue line and shaded region) closely tracks the Oracle, minimizing waste by dynamically aligning draft cost with generation difficulty.

6.4 Ablation Study

6.4.1 Ablating Robust Tree Initialization

TALON employs a hybrid strategy that begins with robust tree initialization for the first layer to ensure robustness, followed by confidence-gated expan-

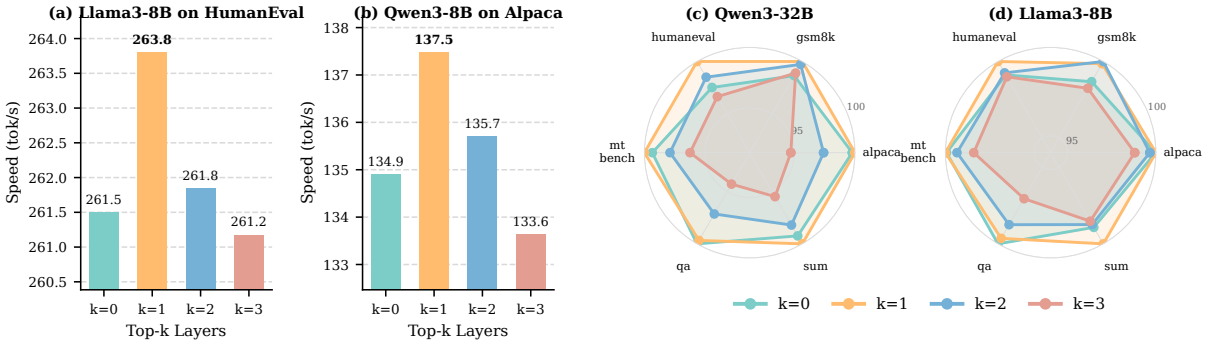


Figure 7: Ablation study on the number of initial Top-K layers (k). $k = 0$ represents the pure adaptive strategy without robust initialization, while $k \geq 1$ indicates using static Top-K expansion for the first k layers. (a)-(b) Wall-time speedup on HumanEval and Alpaca shows that $k = 1$ (**TALON**'s default) achieves the highest throughput. (c)-(d) Radar charts across six benchmarks further confirm that $k = 1$ (orange line) yields the most robust performance, outperforming both the uninitialized ($k = 0$) and over-extended ($k \geq 2$) configurations.

sion layers. However, a natural question arises, *"Why is robust tree initialization necessary only for the first layer"?* To answer this question, we conduct an ablation study that using robust tree initialization for the first k layers. $k = 0$ corresponds to a pure confidence-gated strategy without robust tree initialization.

Necessity of Robust Initialization ($k = 1\ vs.\ k = 0$). As shown in Fig. 7 (a) and (b), removing the robust tree initialization leads to a noticeable drop in generation speed (e.g., declining from 263.8 to 261.5 tok/s on Llama3-8B).This degradation validates our hypothesis that draft models often suffer from root layer *over-confidence*; without a forced Top-K expansion at the first layer, the model risks falling into incorrect branches due to over-confidence, leading to early draft rejections.

Overhead of additional Robust Initialization ($k \geq 2$). Conversely, extending the robust initialization phase to deeper layers ($k = 2, 3$) yields suboptimal returns. It delays the transition to the more efficient confidence-gating expansion, thereby increasing the computational overhead (δ) without a proportional gain in acceptance.The radar charts in Fig. 7 (c) and (d) clearly illustrate that $k = 1$ (orange line) consistently outperforms other configurations, confirming that a single layer of robust initialization followed immediately by adaptive gating achieves the optimal speedup.

6.4.2 Ablating Confidence-gated Expansion

To further investigate the mechanism behind **TALON**, we conduct two additional ablation studies on the influence of different threshold μ and different budget N in Appendix E.4. The results show that the threshold μ in **TALON** serves as a trade-off hyperparameter **between exploration and exploitation**. A larger μ leads to more deep-and-narrow draft trees, especially effective in reasoning tasks such as coding and math, while a smaller μ is more appropriate for some creative tasks. Moreover, the results demonstrate that **TALON** is more flexible to different computation budget. For some resource-intensive scenarios, static methods waste more computation with fixed K and D. However, **TALON** seamlessly adapts to different computation budget by simply adjusting N.

6.5 Case Study

We also provide an real-world case of **TALON** generated draft trees in Appendix E.5. Additionally, we provide an in-depth runtime breakdown of the tree construction phase in Appendix E.6, verifying that the algorithmic overhead of **TALON** remains negligible even with large vocabularies.

7 Conclusion

In this work, we presented **TALON**, a training-free framework that shifts speculative decoding from rigid geometric constraints to a flexible, budget-driven paradigm. By employing a hybrid expansion strategy that combines robust initialization with confidence gating, **TALON** dynamically shapes the draft tree—evolving into *deep-and-narrow* chains for deterministic contexts or *shallow-and-wide* branches for uncertain ones. Extensive evaluations across 5 LLMs and 6 benchmarks demonstrate that **TALON** consistently outperforms SOTA methods like EAGLE-3, achieving up to 5.16× speedup.

8 Limitations

While **TALON** demonstrates significant speedups and adaptability across various benchmarks, we identify several limitations that present avenues for future research:

Scalability to Large Batch Sizes. Our current evaluation focuses on latency-critical scenarios with a batch size of 1, which is the primary use case for real-time interaction. In high-throughput scenarios with large batch sizes, the compute-bound nature of the GPU may saturate, and the overhead of maintaining diverse dynamic tree structures for each request in a batch could become non-trivial. The memory management for varying tree topologies across a batch also presents implementation challenges. Extending budget-driven adaptive speculation to large-batch serving systems remains an open engineering challenge.

Hyperparameter Generalization. While we found a fixed threshold ($\mu = 0.03$) and budget ($N = 60$) to be robust across most tested models and datasets, optimal performance in highly specialized domains might require task-specific tuning. Developing an auto-tuning mechanism that adjusts μ and N on-the-fly based on acceptance history would be a valuable extension.

9 Ethic Statements

Inheritance of Model Behaviors. **TALON** is an inference acceleration framework designed to speed up existing Large Language Models without modifying their weights. As a speculative decoding method, it aims to losslessly recover the distribution of the target model. Consequently, **TALON** inherits the ethical properties, biases, and potential safety risks of the underlying target LLM and draft model. It does not introduce new capabilities for generating harmful content, nor does it mitigate existing biases in the base models. Users should continue to apply standard safety guardrails and alignment techniques to the target models deployed with **TALON**.

References

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. 2024. Hydra: Sequentially-dependent draft heads for medusa decoding. *Preprint*, arXiv:2402.05109.

Anthropic Team. 2025. System card: Claude opus 4.5. https://assets.anthropic.com/m/64823ba7485345a7/Claude-Opus-4-5-System-Card.pdf. System card, Anthropic.

Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Edgar Schönfeld, Ali Thabet, and Jonas Kohler. 2025. Judge decoding: Faster speculative sampling requires going beyond model alignment. *Preprint*, arXiv:2501.19309.

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 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

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

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, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. 2025. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *Preprint*, arXiv:2402.12374.

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

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *Preprint*, arXiv:2305.14233.

Cunxiao Du, Jing Jiang, Xu Yuanchen, Jiawei Wu, Sicheng Yu, Yongqi Li, Shenggui Li, Kai Xu, Liqiang Nie, Zhaopeng Tu, and Yang You. 2024. Glide with a cape: a low-hassle method to accelerate speculative decoding. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2024. Break the sequential dependency of llm inference using lookahead decoding. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Xiangxiang Gao, Weisheng Xie, Yiwei Xiang, and Feng Ji. 2025. Falcon: faster and parallel inference of large language models through enhanced semi-autoregressive drafting and custom-designed decoding tree. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. 2024. Better & faster large language models via multi-token prediction. *Preprint*, arXiv:2404.19737.

Google DeepMind Team. 2025. Gemini 3 pro model card. https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf. Model card, Google DeepMind.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. 2024. REST: Retrieval-based speculative decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1582–1595, Mexico City, Mexico. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Yuxuan Hu, Ke Wang, Xiaokang Zhang, Fanjin Zhang, Cuiping Li, Hong Chen, and Jing Zhang. 2024. Sam decoding: Speculative decoding via suffix automaton. *Preprint*, arXiv:2411.10666.

Kaixuan Huang, Xudong Guo, and Mengdi Wang. 2025. Specdec++: Boosting speculative decoding via adaptive candidate lengths. *Preprint*, arXiv:2405.19715.

Feiye Huo, Jianchao Tan, Kefeng Zhang, Xunliang Cai, and Shengli Sun. 2025. C2t: A classifier-based tree construction method in speculative decoding. *Preprint*, arXiv:2502.13652.

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*, 7:453–466.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR.

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*, pages 7421–7432, Miami, Florida, USA. Association for Computational Linguistics.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025a. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *Preprint*, arXiv:2503.01840.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025b. Eagle: Speculative sampling requires rethinking feature uncertainty. *Preprint*, arXiv:2401.15077.

Jiahao Liu, Qifan Wang, Jingang Wang, and Xunliang Cai. 2024. Speculative decoding via early-exiting for faster llm inference with thompson sampling control mechanism. *Preprint*, arXiv:2406.03853.

Tianyu Liu, Yun Li, Qitan Lv, Kai Liu, Jianchen Zhu, Winston Hu, and Xiao Sun. 2025a. PEARL: Parallel speculative decoding with adaptive draft length. In *The Thirteenth International Conference on Learning Representations*.

Tianyu Liu, Qitan Lv, Hao Li, Xing Gao, and Xiao Sun. 2025b. Logitspec: Accelerating retrieval-based speculative decoding via next next token speculation. *Preprint*, arXiv:2507.01449.

Xianzhen Luo, Yixuan Wang, Qingfu Zhu, Zhiming Zhang, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. 2024. Turning trash into treasure: Accelerating inference of large language models with token recycling. *Preprint*, arXiv:2408.08696.

Jonathan Mamou, Oren Pereg, Daniel Korat, Moshe Berchansky, Nadav Timor, Moshe Wasserblat, and Roy Schwartz. 2024. Dynamic speculation lookahead accelerates speculative decoding of large language models. *Preprint*, arXiv:2405.04304.

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 '24, page 932–949, New York, NY, USA. Association for Computing Machinery.

Nguyen Nhat Minh, Andrew Baker, Clement Neo, Allen G Roush, Andreas Kirsch, and Ravid Shwartz-Ziv. 2025. Turning up the heat: Min-p sampling for creative and coherent LLM outputs. In *The Thirteenth International Conference on Learning Representations*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, 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*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. 2020. Cuda, release: 10.2.89.

OpenAI Team. 2025. Gpt-5 system card. `https://cdn.openai.com/gpt-5-system-card.pdf`. System card, OpenAI.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Preprint*, arXiv:1912.01703.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2022. Efficiently scaling transformer inference. *Preprint*, arXiv:2211.05102.

QwenTeam, 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, and 42 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Apoorv Saxena. 2023. Prompt lookup decoding.

Ziteng Sun, Uri Mendlovic, Yaniv Leviathan, Asaf Aharoni, Jae Hun Ro, Ahmad Beirami, and Ananda Theertha Suresh. 2025. Block verification accelerates speculative decoding. In *The Thirteenth International Conference on Learning Representations*.

GLM Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, and 152 others. 2025. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *Preprint*, arXiv:2508.06471.

Llama Team. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jikai Wang, Yi Su, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, and Min Zhang. 2024. Opt-tree: Speculative decoding with adaptive draft tree structure. *Preprint*, arXiv:2406.17276.

Yepeng Weng, Dianwen Mei, Huishi Qiu, Xujie Chen, Li Liu, Jiang Tian, and Zhongchao Shi. 2025. Coral: Learning consistent representations across multi-step training with lighter speculative drafter. *Preprint*, arXiv:2502.16880.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Bin Xiao, Chunan Shi, Xiaonan Nie, Fan Yang, Xiangwei Deng, Lei Su, Weipeng Chen, and Bin Cui. 2024. Clover: Regressive lightweight speculative decoding with sequential knowledge. *Preprint*, arXiv:2405.00263.

Yunfan Xiong, Ruoyu Zhang, Yanzeng Li, Tianhao Wu, and Lei Zou. 2024. Dyspec: Faster speculative decoding with dynamic token tree structure. *Preprint*, arXiv:2410.11744.

Jinwei Yao, Kaiqi Chen, Kexun Zhang, Jiaxuan You, Binhang Yuan, Zeke Wang, and Tao Lin. 2025. Deft: Decoding with flash tree-attention for efficient tree-structured llm inference. *Preprint*, arXiv:2404.00242.

Ziqian Zeng, Jiahong Yu, Qianshi Pang, Zihao Wang, Huiping Zhuang, Hongen Shao, and Xiaofeng Zou. 2024. Chimera: A lossless decoding method for accelerating large language models inference by fusing all tokens. *Preprint*, arXiv:2402.15758.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. 2024a. Draft & verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11263–11282, Bangkok, Thailand. Association for Computational Linguistics.

Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. 2025. Learning harmonized representations for speculative sampling. In *The Thirteenth International Conference on Learning Representations*.

Situo Zhang, Hankun Wang, Da Ma, Zichen Zhu, Lu Chen, Kunyao Lan, and Kai Yu. 2024b. Adaeagle: Optimizing speculative decoding via explicit modeling of adaptive draft structures. *Preprint*, arXiv:2412.18910.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Algorithm 1 Static Tree Construction (EAGLE)

Require: Draft Model $\mathcal{M}_d$, Prefix $x_{\leq t}$, Depth D, Width K, Budget N

$\mathcal{T} \leftarrow \{x_t\}$, $\mathcal{P}_0 \leftarrow \{x_t\}$, $p(x_t) \leftarrow 1.0$
Fixed depth and width for-loop
for $d = 0$ to D **do**
  $\mathcal{S}_d \leftarrow \emptyset$
  ▷ Expand to $K \times K$ child nodes
  for $v \in \mathcal{P}_d$ **do**
    $C_v \leftarrow \text{Top-}K(P_{\mathcal{M}_d}(\cdot \mid x_{\leq v}))$
    $p(u) \leftarrow p(v) \cdot P_{\mathcal{M}_d}(u \mid x_{\leq v}), \forall u \in C_v$
    $\mathcal{S}_d \leftarrow \mathcal{S}_d \cup C_v$
  end for
  ▷ Shrink to K next-layer parent nodes
  $\mathcal{P}_{d+1} \leftarrow \text{Top-}K(\mathcal{S}_d \text{ by } p(u))$
  $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{P}_{d+1}$
end for
▷ Final pruning to budget N
$\mathcal{T} \leftarrow \text{Top-}N(\mathcal{T} \text{ by } p(u))$
return $\mathcal{T}$

Algorithm 2 Adaptive Tree Construction (**TALON**)

Require: Draft Model $\mathcal{M}_d$, Prefix $x_{\leq t}$, Budget N, Width K, Threshold μ

$\mathcal{T} \leftarrow \{x_t\}$, $\mathcal{P}_0 \leftarrow \{x_t\}$, $p(x_t) \leftarrow 1.0$
$d \leftarrow 0$
▷ Budget-driven adaptive tree expansion
while $|\mathcal{T}| < N$ **do**
  $\mathcal{S}_d \leftarrow \emptyset$
  ▷ Gather next-layer candidate set $\mathcal{S}_d$
  for $v \in \mathcal{P}_d$ **do**
    $C_v \leftarrow \{(v, w) \mid w \in \mathcal{V}\}$
    $p(u) \leftarrow p(v) \cdot P_{\mathcal{M}_d}(w \mid x_{\leq v}), \forall u = (v, w) \in C_v$
    $\mathcal{S}_d \leftarrow \mathcal{S}_d \cup C_v$
  end for
  ▷ Hybrid Expansion Strategy
  if $d = 0$ **then**
    ▷ Robust Init
    $\mathcal{P}_{d+1} \leftarrow \text{Top-}K(\mathcal{S}_d \text{ by } p(u))$
  else
    ▷ Confidence Gating
    $m_d \leftarrow \max_{u \in \mathcal{S}_d} p(u)$
    $\mathcal{P}_{d+1} \leftarrow \{u \in \mathcal{S}_d \mid p(u) \geq \mu \cdot m_d\}$
  end if
  ▷ Budget Check
  if $|\mathcal{T}| + |\mathcal{P}_{d+1}| > N$ **then**
    $\mathcal{P}_{d+1} \leftarrow \text{Top-}(N - |\mathcal{T}|)(\mathcal{P}_{d+1} \text{ by } p(u))$
  end if
  $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{P}_{d+1}$
  $d \leftarrow d + 1$
end while
return $\mathcal{T}$

A Formalized Algorithms

In this section, we provide the formalized algorithms for both the static baseline and our proposed framework to illustrate the structural differences in their expansion strategies.

Algorithm 1 outlines the static tree construction strategy employed by state-of-the-art methods such as EAGLE. This approach relies on rigid geometric constraints defined by a fixed depth D and width K. The construction proceeds layer-by-layer up to depth D. At each step d, the draft model expands $K \times K$ child nodes from the parent set $\mathcal{P}_d$ and subsequently applies a Top-K operation to shrink the candidates back to size K for the next layer. Finally, the tree is pruned globally to meet the token budget N. This "expand-then-shrink" mechanism enforces a static topology regardless of generation difficulty, often generating redundant nodes that are discarded during intermediate steps. Note that "static" here means that each layer of the draft tree has a fixed number of nodes regardless of the context difficulty. The "dynamic" claimed in (Li et al., 2024) means that the draft model dynamically select top-K nodes as next-layer parents, but it cannot adjust K to adapt for context. Figure 8 shows 4 different EAGLE-style draft trees with static tree topology.

Algorithm 2 presents the details of **TALON**, our proposed budget-driven adaptive framework. Unlike static methods, **TALON** is constrained only by a global node budget N and constructs the tree iteratively until this capacity is filled. The core of Algorithm 2 is the Hybrid Expansion Strategy, which dynamically selects the expansion logic based on the current depth. For the root layer ($d = 0$), **TALON** employs a Robust Tree Initialization via a standard Top-K expansion to mitigate potential mis-calibration in the draft model's initial prediction. For all subsequent layers ($d \geq 1$), the algorithm switches to a Confidence-Gated Expansion mechanism. It calculates an anchor confidence m_d within the candidate pool and filters nodes using a relative threshold μ (i.e., $p(u) \geq \mu \cdot m_d$). This "prune-while-expanding" approach allows the tree topology to adapt between "deep-and-narrow" for deterministic contexts and "shallow-and-wide" for uncertain ones.

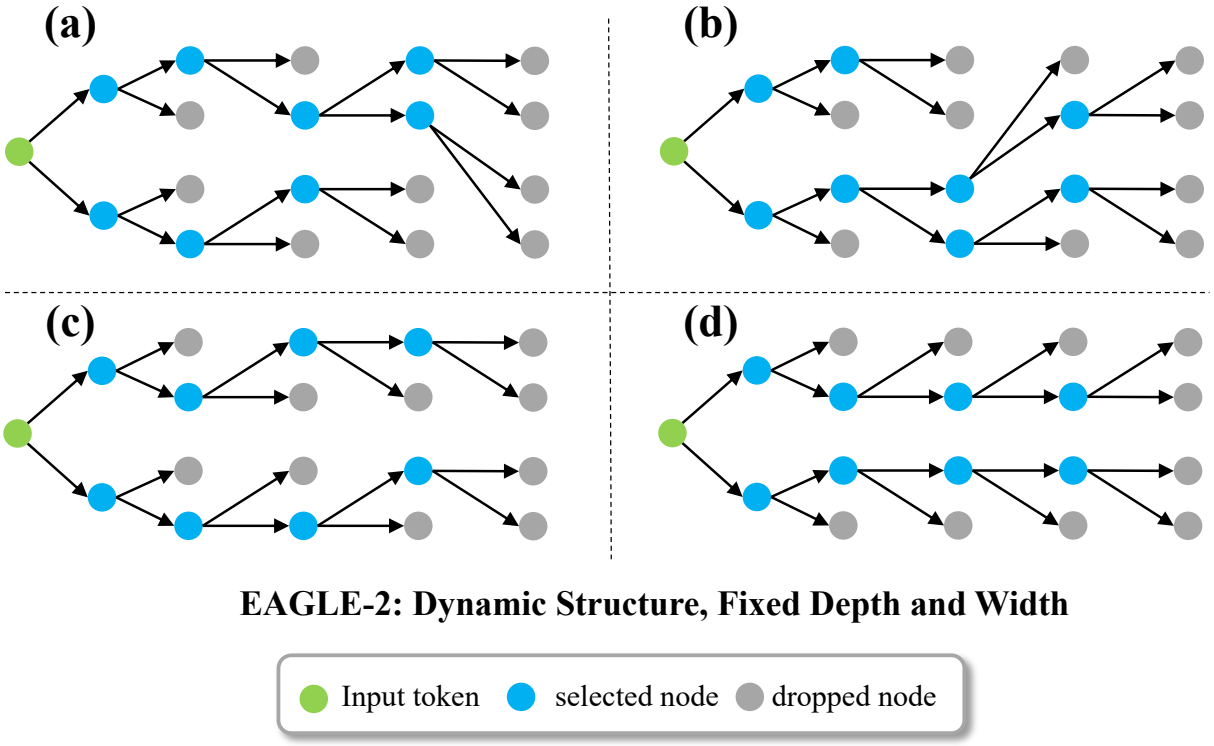


Figure 8: Examples of EAGLE-style draft token trees. At each layer, EAGLE uses a top-K operation to select K nodes as next layer parents. In this way, EAGLE can dynamically adjust the edges between last-layer parents and current-layer children. However, they cannot adjust K to adapt for contexts, which leads to significant resource wastes.

B Token-Tree Verification

The verification phase of our framework adheres to the standard paradigm of tree-based speculative decoding, widely adopted in the field (Li et al., 2025b; Miao et al., 2024). By utilizing the Tree Attention mechanism, the target LLM verifies the entire draft tree in a single forward pass. A structured attention mask ensures that each token within the tree attends only to its predecessors along the root-to-leaf path, effectively simulating independent parallel verifications for all candidate branches while maintaining causal consistency.

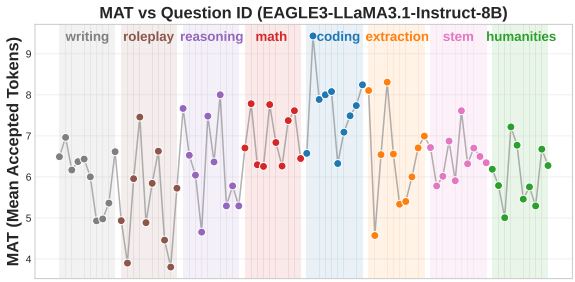
Following the methodology established in Li et al. (2025b), the verification proceeds in three key steps: (1) calculating the posterior probability of each token in the tree given the target model's output; (2) selecting the optimal valid prefix that satisfies the acceptance criteria; and (3) resampling a correction token from the residual distribution at the point of divergence. This rigorous process guarantees that the final output distribution is mathematically identical to that of the target model. We refer readers to the original EAGLE paper (Li et al., 2025b) for detailed algorithms and proofs.

C Additional Motivated Experiments

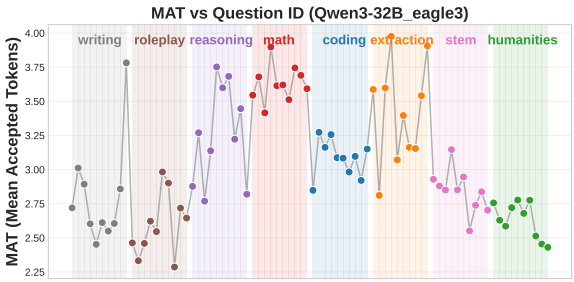
To demonstrate the generality of the limitations observed in Section 3, specifically the "Acceptance Funnel" phenomenon and the "Static Depth Dilemma," we conduct additional empirical analyses on a broader range of LLMs. These experiments confirm that the inefficiencies of static tree structures are not model-specific artifacts but inherent challenges in speculative decoding.

C.1 Heat Map Visualization of LlaMA-3.1-Instruct-8B

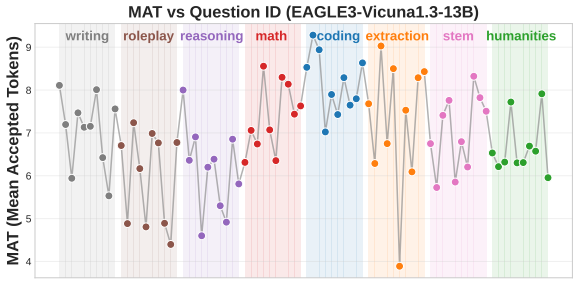
In Section 4, we utilized Qwen3-8B to illustrate the funnel-like distribution of accepted tokens. To verify whether this pattern holds across different architectures, we perform the same visualization on Llama-3.1-8B-Instruct. As shown in Figure 10, the results exhibit a striking similarity to our previous findings. The acceptance probability in the initial layer is relatively dispersed, necessitating a robust search width to capture correct continuations. However, as the tree deepens, the acceptance mass concentrates sharply on the high-confidence regions. This consistent "Acceptance Funnel" across models further justifies **TALON**'s hybrid expansion strat-



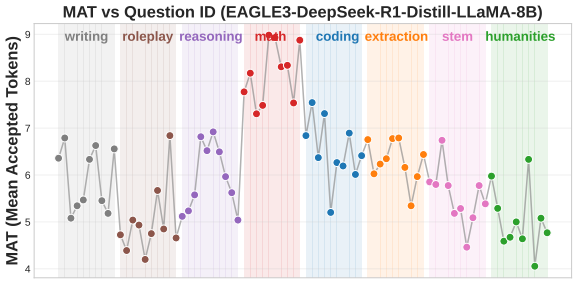
(a) Llama-3.1-Instruct-8B



(b) Qwen3-32B



(c) Vicuna-13B



(d) DeepSeek-R1-Distill-LLaMA-8B

Figure 9: Real-World Mean Accepted Tokens (MAT) distribution across different queries with EAGLE baseline. We visualize the MAT fluctuations on (a) Llama-3.1-8B, (b) Qwen3-32B, (c) Vicuna-13B, and (d) DeepSeek-R1-Distill-8B. Across varying parameter scales and architectures, we consistently observe significant volatility in generation difficulty: low-entropy tasks (e.g., Math/Coding) often allow for high acceptance, while high-entropy tasks (e.g., Roleplay) necessitate shallow trees. This universal variance highlights the limitation of static fixed-depth policies and underscores the necessity of **TALON**'s adaptive strategy.

egy: enforcing robustness at the root while employing confidence-gated pruning at deeper layers to eliminate the redundancy of static width.

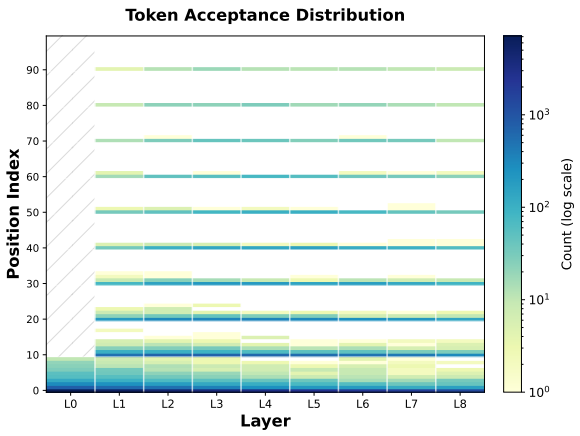


Figure 10: Visualization of token acceptance frequency with Llama-3-8B using a static draft tree ($K = 10, D = 8$). Consistent with Figure 3, the heatmap demonstrates the "Acceptance Funnel" phenomenon: acceptance is dispersed in the first layer but rapidly concentrates on top-ranked candidates in deeper layers, rendering wide static expansion inefficient.

C.2 Dynamic Accepted Length of Various Models

We further extend the analysis of the "Static Depth Dilemma" to evaluate how generation difficulty fluctuates across different models and tasks. Figure 9 presents the Mean Accepted Tokens (MAT) distribution for Llama-3.1-8B-Instruct, Qwen3-32B-Instruct, Vicuna-13B, and DeepSeek-R1-Distill-LLaMA-8B, respectively.

Across all evaluated models and diverse task categories, we observe significant volatility in the effective speculation length. In deterministic, logic-constrained tasks such as **Math and Coding**, the MAT frequently reaches high peaks. In these low-entropy scenarios, the draft model is often confident and accurate, yet a pre-defined static depth limit artificially caps the potential speculation length, preventing the system from fully exploiting the easy context for maximum speedup. Conversely, in open-ended or high-entropy tasks like **Roleplay or Writing**, the acceptance length often drops significantly due to higher uncertainty. Here, a rigid static tree forces the draft model to hallucinate deep branches that are destined for rejection, resulting in wasted computation. This uni-

versal variance underscores that a "one-size-fits-all" static tree structure is fundamentally suboptimal, highlighting the necessity of **TALON**'s budget-driven adaptive mechanism that dynamically adjusts tree depth based on real-time confidence.

D Derivation of Draft Efficiency

In this section, we provide the detailed derivation of the wall-time speedup formula presented in Section 5.3. We define the wall-clock time for a single forward pass of the draft model and the target model as T_q and T_p, respectively. Let L denote the total number of tokens in the final generated output sequence. Throughout the generation process, the draft model executes a total of N_q forward passes, while the target model executes N_p verification steps.

In standard Auto-Regressive (AR) decoding, the target model generates tokens sequentially, resulting in a total inference latency:

$$T_{AR} = L \cdot T_p, \tag{8}$$

In contrast, Speculative Decoding (SD) decouples the process into drafting and verification phases. The total latency is the sum of time spent on both phases, expressed as:

$$T_{SD} = N_p \cdot T_p + N_q \cdot T_q. \tag{9}$$

The wall-time speedup R is strictly defined as the ratio of the baseline latency to the speculative latency:

$$R = \frac{T_{AR}}{T_{SD}} = \frac{L \cdot T_p}{N_p \cdot T_p + N_q \cdot T_q}. \tag{10}$$

To relate this speedup to algorithmic efficiency, we simplify the fraction by dividing both the numerator and the denominator by $N_p \cdot T_p$. We introduce three key metrics: (1) the relative model cost c:

$$c = \frac{T_q}{T_p}, \tag{11}$$

(2) the Mean Accepted Tokens τ:

$$\tau = \frac{L}{N_p}, \tag{12}$$

and (3) the Draft Efficiency δ

$$\delta = \frac{N_q}{N_p}, \tag{13}$$

which quantifies the computation invested in drafting per verification step. Substituting these variables into Equation 10 yields:

$$R = \frac{\frac{L}{N_p}}{1 + \frac{N_q}{N_p} \cdot \frac{T_q}{T_p}} = \frac{\tau}{1 + c \cdot \delta}. \tag{14}$$

This relationship highlights the advantage of **TALON**. Static methods enforce a fixed depth D, locking draft efficiency to a constant $\delta = D + 1$. Consequently, when the acceptance length τ drops in difficult contexts, the speedup R degrades significantly. In contrast, **TALON** dynamically adjusts the tree size, reducing δ in uncertain scenarios. By ensuring the draft cost δ scales down alongside τ, **TALON** preserves a robust speedup R across varying generation difficulties.

E Details of Experiments

E.1 Implementation Details

E.1.1 Comparison Baselines

To strictly evaluate the effectiveness of **TALON**, we compare it against a comprehensive set of competitive baselines, categorizing them into chain-based, tree-based and other MLP-based speculative decoding approaches. We first include standard **Speculative Decoding (SD)** (Leviathan et al., 2023; Chen et al., 2023) as the fundamental chain-based baseline to measure the raw speedup gain over auto-regressive decoding. For MLP-based methods, which aim to reuse target hidden states with MLP to predict multiple draft tokens, we compare against MEDUSA (Cai et al., 2024) and HYDRA (Ankner et al., 2024). These methods employ lightweight MLP decoding heads to predict subsequent tokens in parallel, representing a distinct direction in efficiently drafting.

In the realm of tree-based speculative decoding, we select EAGLE-3 (Li et al., 2025a) as the primary state-of-the-art baseline. EAGLE-3 utilizes feature-level auto-regression with a multi-layer fusion mechanism and typically constructs a static draft tree with fixed width and depth. Comparing against EAGLE-3 allows us to directly demonstrate the advantages of our adaptive topology over rigid geometric constraints.

Most importantly, we include **OPT-Tree** (Wang et al., 2024) as a key baseline, as it represents the related work most closely aligned with **TALON**. Similar to our approach, OPT-Tree aims to optimize the draft tree topology. However, a critical

Target Model	Method	Draft Model Checkpoint (Hugging Face)
Vicuna-1.3-13B	Medusa	`FasterDecoding/medusa-vicuna-13b-v1.3`
	Hydra	`ankner/hydra-vicuna-13b-v1.3`
	SD (Standard)	`double7/vicuna-68m`
	OPT-Tree / EAGLE-3 / **TALON**	`yuhuili/EAGLE3-Vicuna1.3-13B`
DSL-8B	EAGLE-3 / **TALON**	`yuhuili/EAGLE3-DeepSeek-R1-Distill-LLaMA-8B`
Llama-3.1-8B-Instruct	EAGLE-3 / **TALON**	`yuhuili/EAGLE3-LLaMA3.1-Instruct-8B`
Qwen3-8B	EAGLE-3 / **TALON**	`AngelSlim/Qwen3-8B_eagle3`
Qwen3-32B	EAGLE-3 / **TALON**	`AngelSlim/Qwen3-32B_eagle3`

Table 2: List of draft model checkpoints used in our experiments. **TALON** shares the identical draft weights with EAGLE-3 across all tested benchmarks to ensure a fair evaluation of the tree topology efficiency.

distinction lies in the construction paradigm: OPT-Tree typically relies on search-based heuristics or a "generate-then-prune" strategy, which can introduce non-trivial computational overhead during inference. In contrast, **TALON** adopts a training-free, budget-driven expansion strategy that dynamically shapes the tree on the fly based on real-time confidence, thereby achieving adaptivity without the latency costs associated with complex search algorithms.

E.1.2 Hardware and Software Configurations

Hardware Environment. All experiments are conducted on a server equipped with a single **NVIDIA H200 (141GB) GPU** (NVIDIA et al., 2020). We perform all evaluations with a batch size of 1 to follow the standard settings and simulate real-world latency-critical inference scenarios.

Software Environment. Our implementation is based on `PyTorch` (Paszke et al., 2019) version `2.6.0+cu124` and `Transformers` (Wolf et al., 2020) version `4.57.1`. The code is compiled with CUDA `12.8`. For SD (Vanilla Speculative Decoding), we use Transformers' official assisted decoding with their default settings. For MEDUSA and HYDRA, we use their official codebases and adhere to their recommended environment settings to ensure fair comparison. For OPT-Tree, as its official codebase only supports EAGLE-2, we re-implement OPT-Tree to use official EAGLE-3 draft model checkpoint.

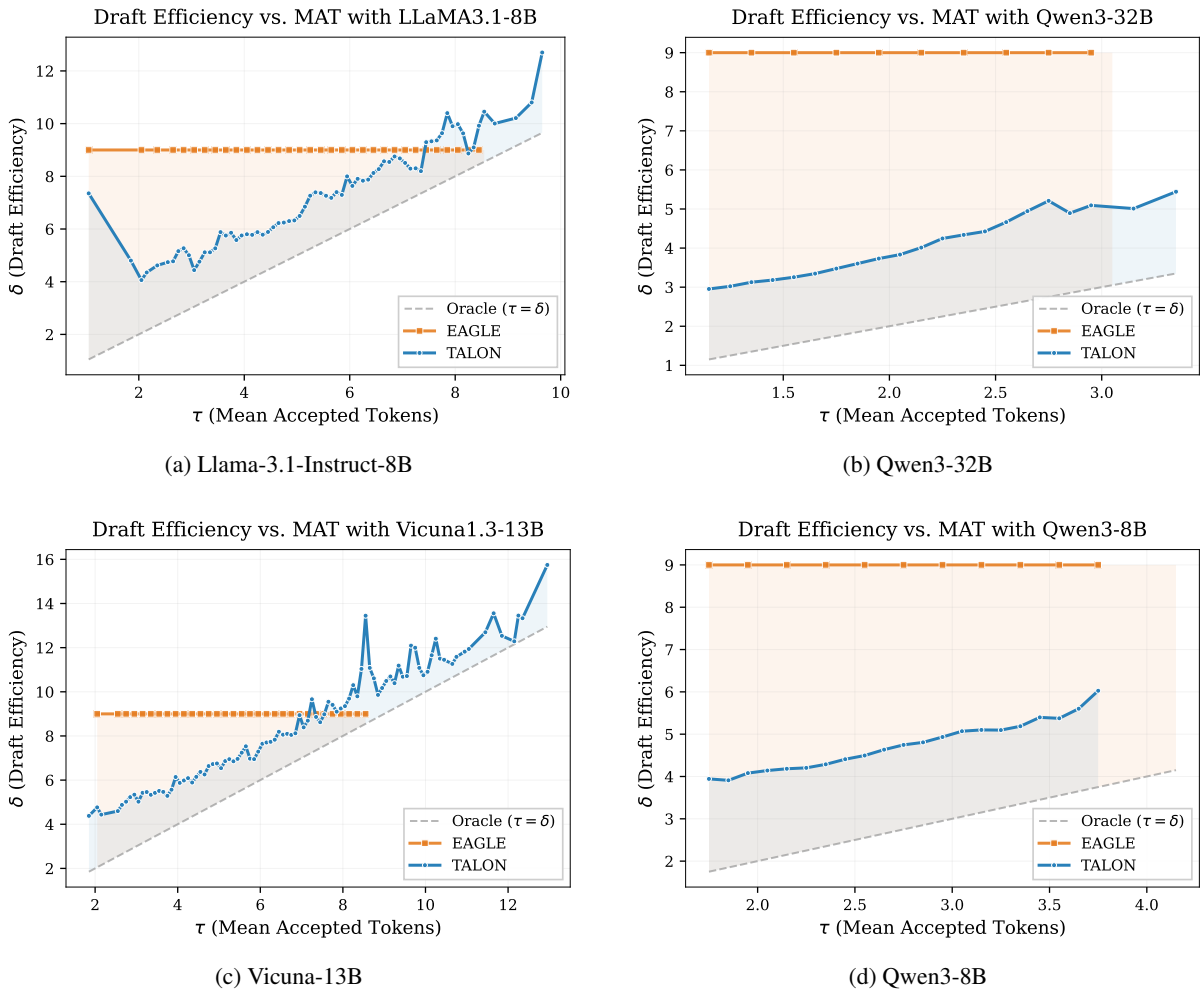
Generation Configuration. Unless otherwise specified, we employ greedy decoding (Temperature $T = 0$) for the main speedup benchmarks reported in Table 1. For the robustness experiments under stochastic sampling (Table 3), we set the temperature to $T = 1.0$ (No additional top-k or top-p operation). The maximum generation length is set to `1024` tokens for standard benchmarks. Regarding **TALON**'s hyper-parameters, we set the global token budget $N = 60$ and the confidence threshold $\mu = 0.03$ by default across all models, $K = 10$ for robust tree initialization. For EAGLE-3, we set the tree width $K = 10$ and $D = 8$, which is reported as optimal values in their manuscripts.

E.1.3 Draft Model Selection

To ensure full reproducibility and a strictly fair comparison, we list the exact Hugging Face model checkpoints used for all draft models in our experiments. Crucially, for TALON, we utilize the *exact same* draft model checkpoints as the state-of-the-art baseline EAGLE-3 (and OPT-Tree where applicable). This ensures that any observed performance gains are attributed solely to our adaptive tree expansion algorithm rather than superior draft weights. The detailed configurations are provided in Table 2.

E.2 More Visualization Results of Draft Efficiency

To comprehensively validate the theoretical efficiency analysis presented in Section 5.3, we provide extended visualizations of the relationship between Draft Efficiency (δ) and Mean Accepted Tokens (τ) across varying model architectures. Figure 11 illustrates this correlation for Llama-3.1-8B-Instruct, Qwen3-32B, Vicuna-13B, and Qwen3-8B. In these plots, the x-axis represents the acceptance length (how many tokens are actually accepted within each decoding step), while the y-axis represents the computational overhead draft efficiency (the ratio of draft steps to verification steps). The gray dashed line serves as the Oracle baseline ($\tau = \delta$), representing an ideal zero-waste scenario



(a) Llama-3.1-Instruct-8B

(b) Qwen3-32B

(c) Vicuna-13B

(d) Qwen3-8B

Figure 11: **Detailed visualization of Draft Efficiency (δ) versus Mean Accepted Tokens (τ) across four different LLMs.** The x-axis represents the mean accepted length, while the y-axis represents the computational cost (draft steps per verification). The gray dashed line denotes the optimal Oracle baseline ($\tau = \delta$) where no computation is wasted. While the static EAGLE-3 baseline (orange) maintains a fixed high cost regardless of difficulty, TALON (blue) dynamically adjusts its draft budget, closely tracking the Oracle curve and significantly reducing computational waste in high-entropy (low τ) scenarios and increasing acceptance reward in low-entropy (high τ) scenarios.

where every drafted token is accepted.

As clearly demonstrated across all four subfigures, the static baseline (EAGLE-3) exhibits a rigid, horizontal trajectory. This pattern confirms that static tree-based methods incur a constant computational overhead determined solely by their fixed geometric hyperparameters (width K and depth D), regardless of the actual generation difficulty. Consequently, in high-entropy scenarios where the acceptance length τ is low, static methods suffer from a substantial "efficiency gap"—highlighted by the extensive orange shaded regions—indicating that the system is squandering computational resources on generating draft branches that are destined to be rejected.

In sharp contrast, TALON demonstrates a highly adaptive behavior, with its efficiency curve strictly tracking the Oracle baseline across the entire spectrum of generation difficulties. The upward-sloping blue trajectory indicates that TALON dynamically modulates its resource investment: it autonomously reduces the draft budget in uncertain contexts to minimize waste, while scaling up the tree size in deterministic contexts to maximize the acceptance length. This tight alignment between the draft cost (δ) and the acceptance reward (τ) empirically verifies that our confidence-gated expansion strategy effectively decouples the draft structure from rigid constraints, ensuring that computational resources are allocated only when they yield a positive marginal utility. (Notably, the curve of Qwen3-8B and Qwen3-32B is relatively far from oracle line, suggesting that there is still a room to increase TALON's end-to-end speedup. We will show in fol-

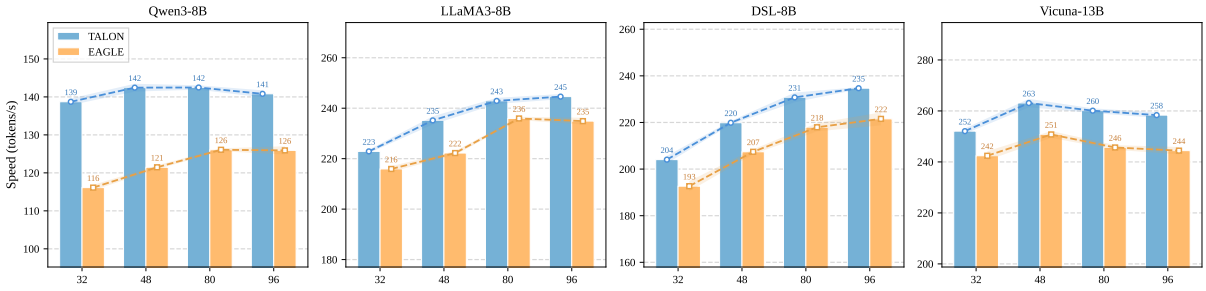


Figure 12: **Ablation study on the global token budget (N).** We compare the wall-time generation speed of **TALON** against the EAGLE-3 baseline across varying budget constraints ranging from $N = 32$ to $N = 96$. The results demonstrate that **TALON** consistently outperforms the static baseline across all budget levels. The performance gap is particularly pronounced in resource-constrained settings (e.g., $N = 32$), confirming that **TALON**'s confidence-gated expansion strategy is significantly more efficient at prioritizing high-value draft tokens than the rigid "expand-then-shrink" mechanism of static methods.

lowing sections that decreasing **TALON**'s threshold from default `0.03` to smaller threshold `0.01` yields better performance.)

E.3 More Evaluation Results under Temperature Settings

While the primary evaluations in the main text focus on greedy decoding (Temperature $T = 0$), real-world LLM applications—particularly creative writing and open-ended chat—frequently rely on stochastic sampling to induce diversity in the generated content. To rigorously assess the robustness of our framework in these non-deterministic scenarios, we conduct a comprehensive evaluation using standard sampling with Temperature $T = 1$. The detailed comparative results between **TALON** and the state-of-the-art baseline EAGLE-3 are reported in Table 3.

As evidenced by the quantitative results, **TALON** consistently outperforms EAGLE-3 across all evaluated models and benchmarks, demonstrating superior adaptability to stochastic environments. A fundamental challenge in high-temperature settings is that the target model's probability distribution becomes flatter, reducing the dominance of the top-1 token and increasing the likelihood of selecting lower-ranked candidates. Static approaches like EAGLE-3, which enforce a fixed width and depth, often struggle in this regime because they rigidly expand the top-K candidates regardless of the entropy. This leads to a scenario where the draft tree either misses the sampled token due to insufficient coverage in uncertain branches or wastes computation on high-confidence paths that are not selected.

In contrast, **TALON**'s confidence-gated expansion strategy naturally excels under these conditions. By determining the tree topology based on relative probability thresholds rather than a fixed node count, TALON dynamically widens the search space in high-entropy layers to capture the dispersed probability mass, while keeping the tree narrow in clearer contexts. This flexibility is reflected in the substantial speedup gains; for instance, on the Vicuna-13B model for HumanEval, **TALON** achieves a wall-time speedup of 3.97×, significantly surpassing the 3.53× speedup of EAGLE-3. Similarly, on Qwen3-8B, **TALON** maintains a distinct advantage in both reasoning-intensive tasks like GSM8K and open-ended tasks like Alpaca. These findings confirm that **TALON**'s budget-driven mechanism is not limited to deterministic acceleration but is a robust solution for diverse generation settings, effectively mitigating the efficiency degradation often observed in stochastic speculative decoding.

E.4 More Ablation Studies of TALON

In this section, we conduct in-depth ablation studies to evaluate the sensitivity and robustness of **TALON** with respect to its two key hyper-parameters: the global token budget (N) and the confidence threshold (μ).

Impact of Global Token Budget (N). We first investigate how the inference performance scales with the available computational budget. Figure 12 compares the wall-time speedup of **TALON** against the EAGLE-3 baseline across varying node budgets ranging from 32 to 96. For a fair comparison, we apply the same global budget constraint to EAGLE-3 using its standard pruning mechanism. The results demonstrate that **TALON** consistently outperforms the static baseline across all budget levels.

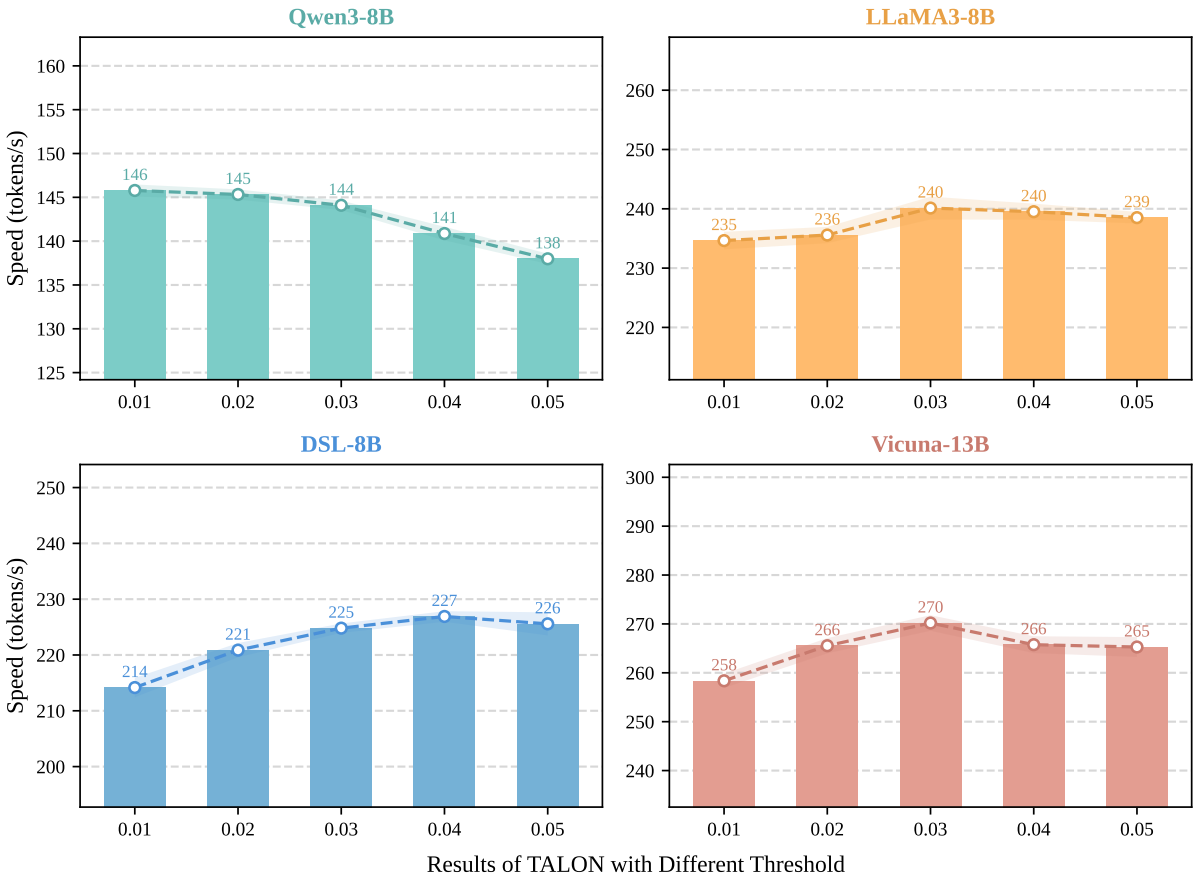


Figure 13: **Sensitivity analysis of the confidence threshold μ.** This parameter controls the trade-off between exploration width and generation depth. The results indicate that the optimal μ is positively correlated with the model's draft-target alignment (MAT). For models with lower alignment like Qwen3-8B (a), a lower threshold ($\mu = 0.01$) yields the best performance by encouraging a "shallow-and-wide" search to ensure coverage. In contrast, for highly aligned models like DSL-8B (c), a higher threshold ($\mu = 0.04$) is preferred to form "deep-and-narrow" chains that maximize the speculation length.

Table 3: **Main Results on Six Benchmarks (Temperature=1).** Comparison between EAGLE-3 and our proposed **TALON** across various models. We report Mean Acceptance Tokens (MAT) and Wall-time Speedup relative to standard decoding. **Bold** numbers denote the best speedup performance.

Model	Method	Alpaca		GSM8K		HumanEval		MT-Bench		QA		CNN/DM	
		MAT	Spd.	MAT	Spd.	MAT	Spd.	MAT	Spd.	MAT	Spd.	MAT	Spd.
Vicuna-13B	EAGLE-3	5.61	3.15×	5.95	3.29×	6.77	3.78×	5.79	3.21×	4.72	2.64×	6.01	2.96×
	SD	1.84	1.11×	1.84	1.08×	2.27	1.29×	2.04	1.13×	1.71	1.05×	2.21	1.23×
	MEDUSA	2.83	2.19×	2.82	2.20×	2.97	2.35×	2.84	2.22×	2.49	1.92×	2.26	1.69×
	HYDRA	4.23	2.86×	3.98	2.73×	4.18	2.91×	4.06	2.77×	3.31	2.25×	3.15	2.07×
	OPT-Tree	5.56	3.21×	5.97	3.25×	6.47	3.53×	5.69	3.16×	4.86	2.77×	5.84	2.79×
	TALON	5.43	**3.28**×	5.82	**3.35**×	7.25	**3.97**×	5.88	**3.38**×	4.63	**2.82**×	5.95	**2.96**×
DSL-8B	EAGLE-3	4.48	2.49×	6.43	3.58×	5.47	3.04×	4.56	2.55×	4.04	2.25×	4.32	2.39×
	TALON	4.29	**2.75**×	6.44	**3.70**×	5.30	**3.25**×	4.44	**2.81**×	3.91	**2.54**×	4.17	**2.63**×
Llama3-8B	EAGLE-3	5.19	2.85×	4.59	2.51×	6.24	3.46×	3.96	2.20×	3.12	1.72×	4.44	2.42×
	TALON	5.21	**3.15**×	4.74	**2.84**×	6.40	**3.56**×	4.13	**2.58**×	3.20	**2.08**×	4.33	**2.62**×
Qwen3-8B	EAGLE-3	3.32	1.95×	3.81	2.24×	3.77	2.20×	3.46	2.02×	3.17	1.87×	3.17	1.85×
	TALON	3.28	**2.25**×	3.74	**2.50**×	3.70	**2.51**×	3.41	**2.32**×	3.15	**2.17**×	3.11	**2.11**×
Qwen3-32B	EAGLE-3	2.55	1.67×	3.22	2.10×	2.95	1.89×	2.74	1.75×	2.44	1.61×	2.46	1.51×
	TALON	2.53	**1.84**×	3.17	**2.27**×	2.91	**2.07**×	2.71	**1.92**×	2.41	**1.77**×	2.43	**1.64**×

Notably, in resource-constrained scenarios (e.g., $N = 32$), TALON exhibits a significant advantage. This indicates that our confidence-gated expansion is highly efficient at prioritizing the most promising candidate tokens, ensuring that a limited budget is invested in high-probability paths rather than being diluted by a fixed-width expansion. As the budget increases to 96, **TALON** continues to scale effectively, utilizing the additional capacity to extend generation depth in deterministic regions, whereas static methods often hit a performance plateau due to their rigid structural constraints.

Sensitivity to Confidence Threshold (μ). Next, we analyze the influence of the confidence threshold μ on generation speed, as visualized in Figure 13. The threshold μ acts as a gatekeeper that balances the trade-off between exploration width and generation depth. Interestingly, we observe that the optimal μ is positively correlated with the degree of alignment between the draft and target models, which can be approximated by the Mean Accepted Tokens (MAT).

For models exhibiting high alignment and high MAT, such as DeepSeek-R1-Distill-LLaMA-8B (DSL-8B), the optimal threshold tends to be relatively high (peaking around $\mu = 0.04$). In these well-aligned scenarios, the draft model's confidence is a reliable proxy for correctness; thus, a stricter threshold effectively filters out unlikely branches, shaping the tree into a "deep-and-narrow" structure that maximizes draft length. Conversely, when the draft-target alignment is weaker (resulting in lower MAT), a lower threshold becomes more advantageous. This phenomenon explains the behavior of Qwen3-8B in Figure 13, where the performance peaks at a lower threshold of $\mu = 0.01$. Here, the draft model is less certain, necessitating a more lenient threshold to encourage "shallow-and-wide" exploration, thereby ensuring sufficient coverage of the target distribution to prevent early rejection.

E.5 More Case Studies of TALON

To qualitatively analyze the behavior of **TALON**, we visualize the decoding traces of a code generation task in Figure 14. Code generation typically represents a low-entropy regime, where the next tokens (e.g., syntax keywords, standard indentations) can be predicted with high confidence.

As shown in the left panel, EAGLE employs a relatively static expansion strategy. Even when the model is confident, EAGLE is limited by its fixed tree structure, resulting in frequent interruptions (pink tokens) and forcing the model to resample from the target model. This limits the Mean Accepted Tokens (MAT) per step.

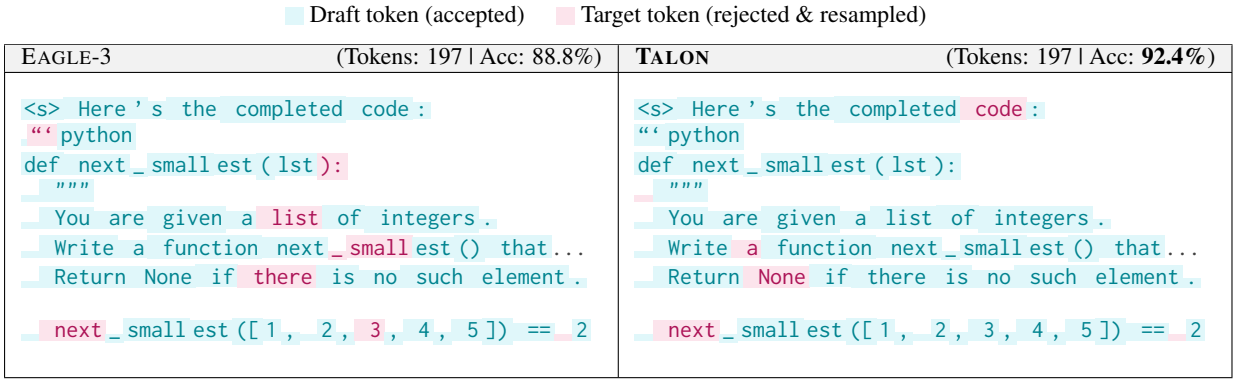


Figure 14: Comparison of token generation traces. **Left:** EAGLE-3 shows frequent interruptions (pink) due to verification failures. **Right:** **TALON** maintains longer accepted chains (cyan) by utilizing confidence-aware adaptive trees.

In contrast, **TALON** (right panel) dynamically leverages its confidence-gated expansion mechanism. Upon detecting the low-entropy nature of the current context, **TALON** adaptively allocates the token budget to extend the tree *depth* rather than width. This allows the draft model to speculate deep-and-narrow draft tokens. Consequently, **TALON** significantly reduces the frequency of verification calls and achieves a higher MAT, demonstrating the efficiency of adaptive token trees in deterministic generation tasks.

E.6 Tree Construction Overhead

To rigorously quantify the algorithmic efficiency of our proposed method, we conduct a micro-benchmark focusing specifically on the **tree construction overhead**. We simulate the next-token probability distributions using a Zipfian distribution $P(r) \propto 1/r^{\alpha}$ (Holtzman et al., 2020), covering diverse generation scenarios ranging from high-entropy creative writing tasks ($\alpha = 0.7$) and standard natural language ($\alpha = 1.35$) to highly deterministic code generation ($\alpha = 5.0$). The warmup step is `20`, and we run each tree expansion strategy `100` times and report their mean latency. In this controlled environment, we compare the single-layer expansion latency of EAGLE against **TALON**. EAGLE relies on a rigid dual Top-K mechanism that performs sorting operations twice per layer—once for child selection and again for parent ranking—which scales poorly with large vocabulary sizes. In contrast, **TALON** utilizes a single confidence-gated sampling operation, implemented via efficient element-wise masking and non-zero index retrieval, which does not involving ranking operation.

As illustrated in Figure 15, **TALON** consistently reduces the tree construction latency across varying vocabulary sizes ($32K$ to $152K$), achieving speedups ranging from $1.18\times$ to $1.44\times$. The advantage is particularly pronounced in deterministic settings ($\alpha = 5.0$), where our adaptive mechanism naturally sparsifies the candidate set, thereby minimizing memory access overhead. However, it is important to acknowledge that **this sampling and tree construction latency constitutes a negligible fraction ($< 5\%$) of the total end-to-end inference time**, which remains dominated by the model's forward passes. Consequently, **TALON**'s primary wall-time speedup derives from its superior Draft Efficiency (δ) rather than this micro-optimization in sampling.

F More Discussions to Related Work

This appendix provides an extended discussion on the evolution of speculative decoding, tracing the field's progression from chain-based verification to tree-structured verification. We highlight how varying approaches trade off draft quality, verification cost, and implementation complexity.

Chain-based speculative decoding. The classical formulation of speculative decoding, often referred to as speculative sampling, is a *draft-and-verify* procedure (Leviathan et al., 2023; Chen et al., 2023; Zhang et al., 2024a). Given a prefix, a fast drafter autoregressively proposes a length-K continuation. The target model then verifies these K positions in a single forward pass, accepting the longest matching prefix and resampling at the first mismatch. While attractive for its lossless nature, the speedup of this framework hinges strictly on the acceptance length and the computational cost

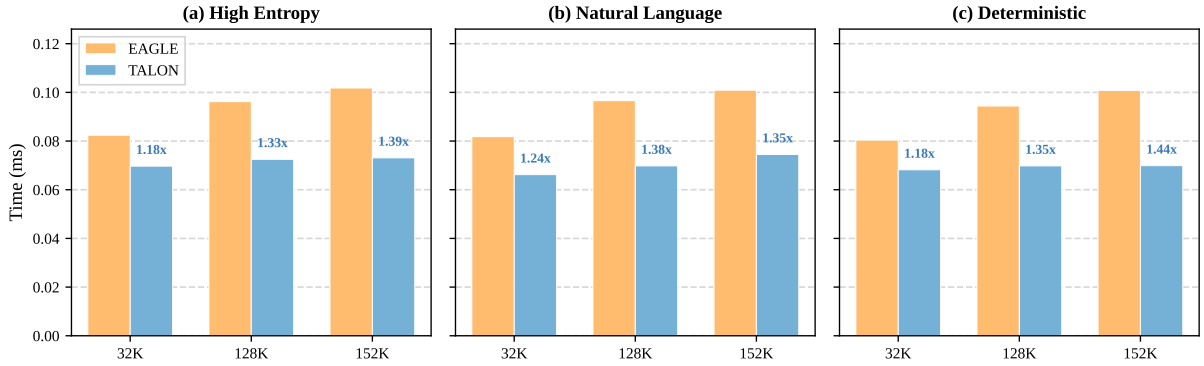


Figure 15: **Runtime breakdown of the tree expansion overhead.** We benchmark the latency of a single-layer expansion step across varying vocabulary sizes ($32K$, $128K$, $152K$) using Zipfian distributions (Holtzman et al., 2020) to simulate High Entropy ($\alpha = 0.7$), Natural Language ($\alpha = 1.35$), and Deterministic ($\alpha = 5.0$) contexts. **TALON** consistently outperforms the static dual Top-K approach of EAGLE, achieving up to **1.44**× speedup in the construction phase by avoiding expensive sorting operations. **Note that this step accounts for less than 5% of the total inference time.**

ratio between the drafter and the target.

Improving the drafter. A major line of work focuses on producing higher-quality drafts with minimal overhead to boost acceptance rates. Medusa (Cai et al., 2024) introduces multiple lightweight decoding heads atop the target model to predict several next tokens in parallel by reusing target hidden states. Because independent heads may ignore intra-draft dependencies, follow-up approaches like Hydra (Ankner et al., 2024) and Clover (Xiao et al., 2024) introduce sequential dependencies among heads to better approximate autoregressive drafting. Chimera (Zeng et al., 2024) proposes a lightweight draft architecture combining short-range and full-context signals to enhance quality while maintaining speed. Complementarily, GliDe with CaPE (Du et al., 2024) reuses the target KV cache to lower drafting overhead and employs confidence-guided proposal expansion to provide stronger candidates.

Controlling draft length and reducing wasted work. Even with a strong drafter, a fixed speculation length K can be suboptimal: large K wastes computation when acceptance is low, while small K caps potential speedup. SpecDec++ (Huang et al., 2025) formulates the candidate length selection as an MDP, adaptively stopping drafting using an acceptance-prediction signal. DISCO (Mamou et al., 2024) similarly predicts when to stop speculation by dynamically selecting the speculation length. PEARL (Liu et al., 2025a) addresses the system-level bottleneck of mutual waiting between drafting and verification by overlapping phases and enabling segmented, adaptive draft lengths. Additionally, Block Verification (Sun et al., 2025) has been proposed to accelerate the verification phase itself.

Addressing training–inference misalignment. When a specialized drafter is trained, mismatches between training distributions and inference-time contexts can hurt acceptance. HASS (Zhang et al., 2025) proposes harmonized objectives and context alignment to reduce these inconsistencies. CORAL (Weng et al., 2025) further improves consistency via cross-step representation alignment and reduces the effective cost of the LM head. Orthogonally, Judge Decoding (Bachmann et al., 2025) observes that many rejected tokens are plausible and proposes relaxing verification via a compact judging module.

Tree-based speculative decoding. Tree-based Speculative Decoding (SD) generalizes the draft chain into a *draft tree*, enabling the verifier to choose among multiple candidate branches. This significantly reduces the penalty of early mismatches. The key enabler is *tree attention*, which allows the target model to verify multiple paths in parallel (Miao et al., 2024; Cai et al., 2024).

Token-tree verification and structured drafting. SpecInfer (Miao et al., 2024) is a representative early system that organizes drafter outputs into a token tree verified via tree-based attention. EAGLE (Li et al., 2025b) advances this by performing autoregression at the feature level. EAGLE-2 (Li et al., 2024) introduces a context-aware dynamic draft tree guided by drafter confidence, while EAGLE-3 (Li et al., 2025a) further improves drafting via multi-layer fusion and training-time tech-

niques.

Training-free tree construction via retrieval. To avoid training specialized drafters, some methods combine SD with retrieval. REST (He et al., 2024) retrieves draft tokens from a datastore, while Prompt Lookup Decoding (Saxena, 2023) leverages n-gram matching within the current context. Token Recycling (Luo et al., 2024) stores previously observed candidate-token transitions in a compact adjacency matrix to retrieve a draft tree. SAM-Decoding (Hu et al., 2024) utilizes a suffix automaton to efficiently find the longest suffix match for drafting. LogitSpec (Liu et al., 2025b) proposes to use the last logit as a guidance to retrieve more matched and accurate draft tokens. While plug-and-play, these methods are bounded by the availability of high-quality matches in the context or datastore. Alternative draft-model-free approaches like Lookahead Decoding (Fu et al., 2024) generate parallel n-grams using Jacobi iteration, and EESD (Liu et al., 2024) uses early exiting from intermediate layers to generate drafts.

Tree-attention efficiency. Since tree-based SD relies on non-trivial KV updates, system efficiency is critical. DeFT (Yao et al., 2025) proposes an IO-aware flash tree-attention algorithm tailored for tree-structured inference to improve verification throughput.

Summary and positioning of TALON. While tree-based SD has become a standard paradigm, existing approaches face distinct limitations in adaptability and efficiency. Learning-based methods, such as C2T (Huo et al., 2025) and AdaEAGLE (Zhang et al., 2024b), require training specialized modules and remain constrained by rigid parameter spaces (e.g., predicting a fixed K), failing to achieve fully elastic topology changes. Optimization-based methods also exhibit drawbacks: Sequoia (Chen et al., 2025) relies on *offline* algorithms to find a globally optimal tree, resulting in a *static* template that cannot adapt to instance-wise difficulty. Similarly, search-based strategies like OPT-Tree (Wang et al., 2024) often follow a "generate-then-prune" paradigm or employ complex search heuristics at inference time, which introduces non-trivial computational overhead. **TALON** distinguishes itself through a *training-free, budget-driven* framework that operates via *on-the-fly* construction. Analogous to "pre-pruning" in decision trees, **TALON** adopts a "prune-while-expanding" strategy: it iteratively allocates the node budget based on real-time confidence, naturally halting expansion in uncertain branches without generating wasteful nodes first. This allows the draft tree to fluidly morph between "deep-and-narrow" and "shallow-and-wide" shapes, maximizing speculation utility with minimal construction cost.

G LLM Usage

We used a large language model (LLM)–based writing assistant solely for grammar and wording improvements on draft text. The LLM did not generate research ideas, claims, proofs, algorithms, code, figures, or analyses, and it did not have access to any non-public data. All edits suggested by the LLM were manually reviewed and either accepted or rewritten by the authors, who take full responsibility for the final content. The LLM is not an author of this paper.