HYBRID VERIFIED DECODING: LEARNING TO ALLOCATE VERIFICATION IN SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) generation remains expensive because autoregressive decoding requires a model call for each new token. Speculative decoding reduces this cost by drafting multiple tokens and verifying them with the target model in one step, but its speedup depends on how many draft tokens are accepted. Parameter-free draft sources, such as suffix caches, can propose long continuations at low cost in structured and agentic workloads, yet their payoff varies across decoding steps. When a promising cache match yields only a short accepted prefix, most of the verification work is wasted. We propose Hybrid Verified Decoding, which uses a payoff predictor to estimate the accepted length of a cache draft before verification and then decides whether to verify the cache draft or fall back to a model-based drafter. Through experiments spanning three LLMs and sixteen datasets, we show that Hybrid Verified Decoding is especially effective on agentic workflows, where it outperforms EAGLE3 in every setting with speedups of 1.4× to 5.5×. Our analysis demonstrates that high-payoff cache drafts are rare and that the payoff predictor identifies most of them. We further show that the predictor transfers to unseen datasets and other target models without retraining.

1 INTRODUCTION

LLM generation is limited by autoregressive decoding: each new token requires a forward pass over the current prefix and key-value (KV) cache. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) reduces this sequential cost through a draft-then-verify interface: a lower-cost source proposes candidate tokens, and the target LLM itself verifies which prefix can be committed. Its speedup depends on how many proposed tokens are accepted at each verification step.

Recent speculative decoding methods instantiate the draft source in different ways. Model-based drafters, such as the EAGLE family (Li et al., 2024b;a; 2025), improve draft reliability through learned proposal models. Lookup, retrieval, and cache-based methods (Saxena, 2023; He et al., 2024; Fu et al., 2024; Oliaro et al., 2025) reuse continuations from the prompt, generation history, or external corpora at low proposal cost. Such reuse is common in structured and agentic workloads, where recurring context can support long cache drafts. The payoff of a cache draft, however, depends on the current generation step. A local match can produce a long draft, but a similar prefix may require different arguments, a different edit goal, or a different piece of evidence. When the target model accepts only a short prefix, the decoder pays the cost of verification but makes little speculative progress.

We study this per-step decision in speculative decoding and propose Hybrid Verified Decoding, which uses a payoff predictor to estimate the accepted length of the current cache draft before verification. If the predicted payoff is high, the cache draft is submitted for verification. Otherwise, the system uses a model-based drafter. The target model remains responsible for committed tokens, while the learned component allocates verification work across draft sources and generation steps. We train the payoff predictor from trace replay and evaluate the resulting decoder across three LLMs and sixteen datasets, where Hybrid Verified Decoding outperforms EAGLE3 in every agentic setting with speedups of 1.4× to 5.5×. Our contributions are:

- We formulate state-dependent draft selection in speculative decoding as payoff-guided verification allocation, with accepted length as the payoff variable.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

- We propose Hybrid Verified Decoding, which predicts the accepted length of the current cache draft and chooses between cache-based drafting and a model-based drafter before verification.
- We evaluate our method across three LLMs, sixteen datasets, and four baseline approaches spanning model-based, cache-based, rule-based hybrid, and greedy decoding.
- We show that high-payoff cache drafts are rare and that the payoff predictor identifies most of them. The predictor transfers across datasets and target models, and the gain carries over to a vLLM serving prototype.

2 Related Work

Parameter-Based Speculative Decoding. Parameter-based methods improve speculative decoding by adding learned proposal modules. The EAGLE family (Li et al., 2024b;a; 2025) uses model representations to construct stronger drafts. Medusa (Cai et al., 2024) and PARD (An et al., 2025) expand candidate generation through multiple heads or parallel draft modules. These methods invest additional parameters and training to improve draft reliability or candidate diversity. Hybrid Verified Decoding uses a model-based drafter as its fallback path and focuses on the verification decision made before that fallback is used.

Parameter-Free Speculative Decoding. Parameter-free methods propose draft tokens without training a separate proposal model. Prompt Lookup Decoding (Saxena, 2023), REST (He et al., 2024), Lookahead Decoding (Fu et al., 2024), and SuffixDecoding (Oliaro et al., 2025) reuse continuations from the prompt, generation history, or external datastores. SAM-Decoding (Hu et al., 2025) uses suffix automata for the same purpose, and Token Recycling (Luo et al., 2025) reuses previously rejected draft tokens. This line of work lowers proposal cost and fits workloads with repeated context, structured formats, or recurring intermediate states. A cache or retrieval match still leaves an open runtime question: how many of the proposed tokens will the target model accept at the current generation step? Our method builds on parameter-free drafting by estimating this accepted length before verification and using it to decide whether the current cache draft should be verified.

Adaptive Speculation. Adaptive speculation methods adjust the behavior of speculative decoding at runtime. AdaEAGLE (Zhang et al., 2024) adapts draft structures, Nightjar (Li et al., 2026) and AdaSpec (Huang et al., 2026) adjust serving-time speculation, ECHO (Hu et al., 2026) and TALON (Liu et al., 2026a) control token trees or candidate budgets, and MARS (Song et al., 2026) modifies verification behavior. These methods show that speculation benefits from runtime control over draft length, tree shape, budget, or verification criteria. They leave an important gap for parameter-free drafts: after a cache match is found, the system still needs to decide whether the proposed continuation is worth verification. Hybrid Verified Decoding addresses this gap by predicting the accepted length of the proposed cache draft before verification and using that payoff estimate to choose between cache verification and a model-based drafter at each generation step.

3 Method

The quality of a cache draft depends on the current decoding state, as an incorrect reuse may be rejected early by the target model. A cache match often diverges after a few tokens, at a changed argument or a different edit, and only about 5% to 9% of cache drafts are accepted for six or more tokens (Section 5). These high-payoff states, however, share signals that are visible before verification. Draft selection therefore reduces to per-step payoff prediction: estimating one quantity, the accepted length of the cache draft, before verification. Hybrid Verified Decoding verifies the cache draft when this estimate is high and otherwise falls back to the model-based drafter. Because the decision rests on this single estimate, it requires no change to the target model, its verification rule, or either drafter. Figure 1 illustrates this decision on two decoding states.

3.1 Draft Verification and Accepted Length

Consider a decoding step t with current prefix $x_{1:t}$, consisting of the input prompt and the tokens generated so far. A draft source D proposes a length-k continuation conditioned on this prefix:

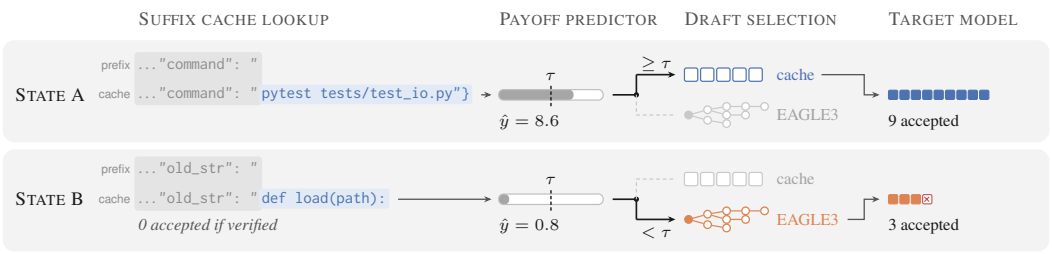


Figure 1: Hybrid Verified Decoding on two illustrative decoding states. The payoff predictor estimates the accepted length $\hat{y}$ of the cache draft before verification: if $\hat{y} \geq \tau$, the cache draft is verified (State A); otherwise, EAGLE3 provides the draft (State B).

$$d_t = (\hat{x}_{t+1}, \ldots, \hat{x}_{t+k}) = D(x_{1:t}).$$

The target model then verifies d_t and accepts a prefix of the proposal. We denote the number of accepted draft tokens by

$$a_t(d_t) \in \{0, \ldots, k\}.$$

Target verification also returns the token block committed after this step:

$$(a_t(d_t), z_t) = \text{Verify}(x_{1:t}, d_t; M).$$

The first $a_t(d_t)$ tokens of z_t match the draft prefix $(\hat{x}_{t+1}, \ldots, \hat{x}_{t+a_t(d_t)})$. If verification rejects the draft before completion, z_t also contains the target-model token at the rejection point, so decoding continues under verification.

The accepted length $a_t(d_t)$ is the payoff of the step: a large value lets one verification pass commit several draft tokens, while a value near zero pays for the pass with little speculative progress. Draft selection therefore amounts to estimating this quantity before verification. The rest of this section describes the cache draft that Hybrid Verified Decoding considers at every step and how its accepted length is predicted before verification.

3.2 CACHE-BASED DRAFTING

Parameter-free drafting constructs proposals by matching recurring token patterns in the input and generation history. We use the suffix cache of SuffixDecoding (Oliaro et al., 2025) as this source because it captures repeated token sequences both within the current prompt and across the outputs of previous requests. The cache $\mathcal{C}$ consists of two suffix trees: a global tree $\mathcal{C}_g$ over previous generations, initialized from warm-up traces, and a request tree $\mathcal{C}_r$ over the prompt of the current request. At each step, the cache finds the longest suffix of $x_{1:t}$ that also occurs in $\mathcal{C}_g$ or $\mathcal{C}_r$ (the shaded match in Figure 1). The cache draft d_t^c is the continuation that most often followed this match. It ends at a maximum draft length, or earlier once no continuation is frequent enough. The lookup also returns the match length and a frequency score s_t. After verification, the committed block z_t is appended to $\mathcal{C}$, so later steps and later requests can match it. We write this lookup as

$$d_t^c = D_{\mathcal{C}}(x_{1:t}; \mathcal{C}).$$

Section 3.3 estimates how many of its tokens the target model will accept at the current step. In Hybrid Verified Decoding, this cache can be replaced by any parameter-free drafter that provides candidate drafts and match statistics.

3.3 LEARNING WHEN TO VERIFY A CACHE DRAFT

We denote the accepted length of the cache draft by

$$y_t = a_t(d_t^c).$$

It is unknown before verification. A long accepted length arises when the current step repeats an earlier one, and the lookup that produced the draft indicates how strong this repetition is: the length

of the matched suffix, how consistently the continuation followed it, and whether recent cache drafts were accepted. Our goal is to estimate y_t from this information before verification. The accepted length can be read off recorded generations without further model calls, and it depends only on the target model and the decoding state.

We encode this information, together with the draft itself and the position in the generation, in a feature vector

$$\phi_t = \Phi(x_{1:t}, d_t^c, \mathcal{C}, h_t),$$

where h_t summarizes the outcomes of recent cache drafts. As detailed in Appendix B, the features fall into four groups: the cache match (frequency score, matched length, draft length, and lookup time), the decoding progress (prompt length, generated length, and relative position), recent cache history (mean accepted length and first-token failure rate), and surface-level draft statistics (fractions of whitespace, punctuation, line breaks, brackets, and digits, and an end-of-sequence flag). All of them are available before verification without a model call. The payoff predictor maps these features to an accepted-length estimate,

$$\hat{y}_t = g_\theta(\phi_t).$$

We instantiate g_θ as a multilayer perceptron. A threshold τ turns the estimate into a verification decision: cache drafts with $\hat{y}_t \geq \tau$ are sent to verification, while lower-scoring states use a model-based drafter. τ is the accepted length above which a cache draft is worth verifying instead of running a model-based drafting cycle. Where this point lies depends on how expensive that cycle is on the hardware and serving engine in use, so a new system requires only a new τ. The predictor itself depends only on the target model and the decoding state, and Section 5 shows that it transfers across datasets and target models without retraining.

Training labels come from recorded generations of the target model and require no further model calls. We replay each trace in order and treat every token position as a decoding state. For a position t, the visible prefix is $x_{1:t}$. The suffix cache built from this prefix returns a cache draft d_t^c, and the features ϕ_t are computed from this prefix alone. Replay therefore gives the predictor the same inputs it sees during decoding. The following tokens $x_{t+1:}$ provide the target continuation for that state. The accepted-length label is the longest prefix of the cache draft that agrees with this continuation:

$$y_t = \max\left\{\ell \in \{0, \ldots, |d_t^c|\} : d_{t,1:\ell}^c = x_{t+1:t+\ell}\right\}.$$

We train g_θ with accepted-length regression:

$$\min_\theta \sum_t \left(g_\theta(\phi_t) - y_t\right)^2.$$

3.4 Hybrid Verified Decoding

For each decoding step, Hybrid Verified Decoding obtains a cache draft d_t^c and computes its predicted accepted length $\hat{y}_t = g_\theta(\phi_t)$. The draft sent to verification is selected by

$$d_t = \begin{cases} d_t^c, & \text{if } \hat{y}_t \geq \tau, \\ D_m(x_{1:t}), & \text{otherwise,} \end{cases}$$

where D_m denotes the model-based drafter, which runs only when the cache draft is not selected. Algorithm 1 in Appendix B summarizes the resulting decoding loop. The target model verifies the selected draft under the rule of Section 3.1. The committed block z_t is appended to the output, the suffix cache is updated with z_t, and decoding proceeds.

4 Experiments

4.1 Experiment Setup

We evaluate our method on three instruction-tuned target models: Qwen3-8B, Qwen3-4B (Yang et al., 2025), and Llama3.1-8B (Grattafiori et al., 2024). Each target model is paired with its corresponding EAGLE3 checkpoint as the model-based fallback drafter. All end-to-end timing runs are measured on H100 GPUs with deterministic decoding.

Table 1: Dataset-level end-to-end speedup of Hybrid Verified Decoding over four baselines across three target models. G, E3, SD, and FH denote Greedy Decoding, EAGLE3, SuffixDecoding, and Frequency-Based Hybrid Decoding. Cell shading follows speedup magnitude. The final row reports the unweighted dataset average.

		Qwen3-8B				Qwen3-4B				Llama3.1-8B			
Group	Dataset	G	E3	SD	FH	G	E3	SD	FH	G	E3	SD	FH
Agentic	Delegate-52	3.25	1.44	2.14	1.06	3.40	1.90	1.64	1.18	3.46	3.38	0.81	1.24
	InstructEdit/FineEdit	3.57	1.51	2.30	1.32	4.21	2.67	1.72	1.22	7.56	5.49	1.15	1.20
Code	RepoBench	2.54	1.09	2.15	1.15	3.57	1.56	1.44	0.81	2.12	2.99	1.09	1.26
	SWE-bench Verified	3.70	1.09	3.21	1.28	2.50	1.08	1.52	0.88	3.31	2.11	2.60	1.26
	Magicoder	3.31	1.08	3.79	1.34	1.73	1.02	1.82	0.87	5.86	2.51	4.28	1.27
	MBPP	4.49	1.17	4.78	1.54	2.73	0.90	2.11	1.01	3.68	0.99	4.65	1.05
Structured	BFCL	2.31	0.98	3.13	1.27	2.19	0.98	2.47	1.13	1.02	0.90	1.32	1.01
	Spider	3.12	0.97	2.98	1.33	1.56	1.01	1.79	0.88	4.46	1.85	3.61	1.24
Multi-hop QA	HotpotQA	2.57	0.95	3.77	1.37	2.03	0.81	1.97	1.02	2.23	0.93	2.70	0.96
	MuSiQue	3.55	0.95	2.88	1.17	2.12	1.18	1.35	1.05	3.00	3.22	1.15	1.97
	2WikiMultiHopQA	2.73	0.96	3.28	1.40	2.01	1.19	1.65	0.98	2.52	1.04	2.49	0.87
Long-context	InfiniteBench	5.03	2.42	3.04	1.21	5.53	2.51	2.40	1.16	2.23	2.84	1.47	2.94
	CNN/DailyMail	2.33	0.94	3.85	1.34	1.79	0.99	2.75	1.02	2.43	0.94	3.83	1.04
	GovReport	2.03	0.95	3.36	1.18	1.52	1.01	2.08	1.02	0.95	1.31	0.98	1.15
Open-ended	Alpaca	2.73	0.97	3.84	1.40	2.06	0.85	2.40	0.83	3.14	0.96	5.68	1.05
	MT-Bench	3.34	0.98	4.68	1.41	1.59	0.99	2.06	0.87	3.24	1.18	4.76	1.10
Avg. across datasets		3.16	1.15	3.32	1.30	2.53	1.29	1.95	0.99	3.20	2.04	2.66	1.29

Our evaluation covers sixteen datasets across six workload families: agentic and workflow editing (Delegate-52, InstructEdit/FineEdit), repository-level and standalone coding (RepoBench, SWE-bench Verified, Magicoder, MBPP), structured generation and tool use (BFCL, Spider), multi-hop QA (HotpotQA, MuSiQue, 2WikiMultiHopQA), long-context document tasks (InfiniteBench, CNN/DailyMail, GovReport), and open-ended instruction/chat (Alpaca, MT-Bench). These workloads differ in input length, structural constraints, context reuse, and output format.

All methods are evaluated inside the generation loop of the official EAGLE repository[1] and use the chat template associated with each target model. The Qwen3 models run with thinking enabled. We train one payoff predictor per target model on replay data pooled from all sixteen datasets and use $\tau = 6$ throughout (Appendix A). We report run-level throughput as generated tokens per second, and define speedup as the throughput ratio relative to the corresponding baseline.

We compare Hybrid Verified Decoding with four baselines. Greedy decoding uses the target model alone. EAGLE3 (Li et al., 2025) drafts with the EAGLE3 checkpoint of the target model. SuffixDecoding (Oliaro et al., 2025) proposes drafts from the suffix cache of Section 3.2 and verifies each of them. Frequency-Based Hybrid Decoding uses the same cache-first, EAGLE3-fallback structure as our method and verifies the cache draft when its frequency score s_t exceeds a fixed threshold. Appendix A lists the configuration of every method.

4.2 Main Results

Table 1 reports end-to-end speedup for Hybrid Verified Decoding across sixteen datasets and three target models. Across the 48 model-dataset pairs, our method is faster than Greedy Decoding in 47 cases and faster than SuffixDecoding in 46 cases, with average speedups of 2.97× and 2.64×, respectively. SuffixDecoding, which uses the cache alone, runs 1.1× to 1.9× faster than Greedy Decoding in aggregate on the three models. Hybrid Verified Decoding reaches 2.6× to 3.3× with the

[1] https://github.com/SafeAILab/EAGLE

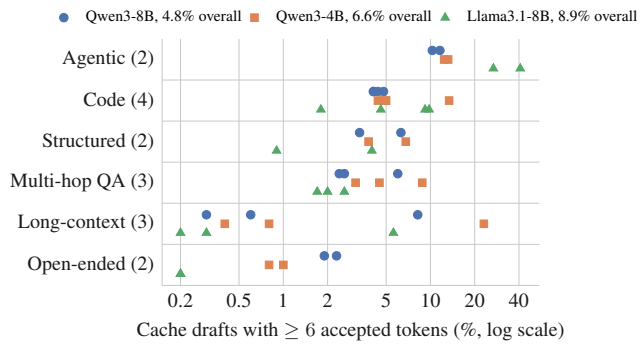


Figure 2: Share of cache drafts that the target model accepts for at least six tokens, one point per dataset, grouped by workload family (log scale).

same cache (see Tables 14 to 16 in Appendix D for the aggregate throughput of every method). The gain therefore comes from combining the cache with the EAGLE3 fallback and the payoff predictor.

Against EAGLE3, the gain is largest on the agentic and workflow-editing workloads. On Delegate-52 and InstructEdit/FineEdit, Hybrid Verified Decoding outperforms EAGLE3 for all three target models. The speedup ranges from 1.44× to 5.49×. These workloads repeatedly expose tool-call structure, editing context, repository state, and intermediate outputs, so the cache returns long drafts at many steps. In multi-hop QA, long-document, and open-ended workloads, cache drafts rarely pay off. The predictor falls back to EAGLE3 at most steps, and the two methods run at similar speed: on Qwen3-8B, Hybrid Verified Decoding stays within 6% of EAGLE3 on every such dataset, and across all three models the lowest ratio is 0.81. Hybrid Verified Decoding thus captures the gain where cache drafts pay off and stays close to EAGLE3 where they do not. Against Frequency-Based Hybrid Decoding, Hybrid Verified Decoding is faster in 39 of the 48 settings, on all sixteen datasets for Qwen3-8B, and on both agentic datasets for every model. The two methods share the cache and the fallback, so this difference is what the learned selection contributes.

The model-level differences follow the cost trade-off identified in recent analyses of speculative decoding (Liu et al., 2026b; Yan et al., 2025): speedup depends jointly on accepted length, drafting cost, verification cost, and system overhead, while acceptance behavior varies across requests, token positions, and datasets. On Qwen3-4B, Hybrid Verified Decoding still improves consistently over Greedy Decoding and SuffixDecoding, but it is on par with Frequency-Based Hybrid Decoding, at 0.99× on average over the sixteen datasets. A verification pass is cheap on this smaller model, so the cache drafts that the frequency rule verifies without payoff cost little, and the two selectors end up close. Section 5 examines where the payoff of cache drafts concentrates, how well the predictor identifies it, and where the speedup comes from.

5 Analysis

5.1 Distribution of Cache Draft Payoff

Figure 2 shows, for every dataset, the share of cache drafts that the target model accepts for at least six tokens, the threshold τ used throughout. Only 4.8%, 6.6%, and 8.9% of cache drafts reach it on Qwen3-8B, Qwen3-4B, and Llama3.1-8B. The share is highest on the two agentic datasets for every model, between 10% and 41%, and on InfiniteBench and RepoBench for Qwen3-4B. On the open-ended chat datasets it stays below 2.5%. These few drafts carry most of the payoff: they are accepted for 18 to 23 tokens on average and account for 56% to 75% of all tokens that cache drafts commit. Verifying every cache draft wastes most verification passes, as SuffixDecoding does in Table 1, and verifying none forfeits the agentic gains. The value of Hybrid Verified Decoding lies in telling the two apart before verification. Appendix F relates this share to the structure of the prompts. Section 5.2 examines how well the payoff predictor does this.

Table 2: Selection quality of the payoff predictor on the offline replay set, in percent. Selected is the share of all cache drafts. The other metrics take drafts with accepted length at least 6 as the positive class.

Model	Selected	Precision	Recall	F1	Payoff recall
Qwen3-8B	3.8	78.2	61.4	68.8	83.3
Qwen3-4B	5.4	82.9	67.8	74.6	87.4
Llama3.1-8B	8.1	87.6	79.8	83.6	93.2

Table 3: Generalization under leave-one-dataset-out evaluation across sixteen workloads (Qwen3-8B).

Predictor training	F1	Mean throughput (tokens/s)	Throughput retained
All sixteen datasets	68.80	128.0	100.0%
Leave-one-dataset-out	68.25	124.8	97.5%

Table 4: Zero-shot transfer of the Qwen3-8B predictor to other target models ($\tau = 6$).

Target model	Target-specific (tokens/s)	Transferred (tokens/s)	Transferred / target-specific
Qwen3-4B	101.09	115.56	114.3%
Llama3.1-8B	126.25	118.33	93.7%

5.2 Identifying High-Payoff Drafts Before Verification

For every cache draft proposed on the test prompts, we compute its accepted length y_t from the generation trace of the target model (Section 3.3). Drafts with $y_t \geq 6$ are high-payoff. Table 2 reports precision, recall, and F1 with high-payoff drafts as the positive class, and payoff recall, the share of the accepted tokens of high-payoff drafts that fall in the selected set.

Only 4.8%, 6.6%, and 8.9% of the cache drafts are high-payoff, and the predictor selects 3.8%, 5.4%, and 8.1%. Its precision is 78% to 88% and its recall 61% to 80%. The high-payoff drafts it recovers are accepted for 25 to 27 tokens on average, the ones it misses for about 8, so payoff recall reaches 83% to 93%. Most of the payoff of cache drafts can be identified before verification.

5.3 Predictor Generalization Across Datasets and Models

We test whether the payoff predictor depends on task-specific supervision using a leave-one-dataset-out protocol across the sixteen workloads. As reported in Table 3, held-out evaluation across the sixteen folds yields an aggregate F1 of 68.25 (versus 68.80 under all-data training) and preserves 97.5% of mean decoding throughput (124.8 vs. 128.0 tokens/s). This consistency indicates that the features behind accepted-length prediction capture general cache reuse dynamics without overfitting to domain-specific prompt patterns.

Beyond task shifts, we examine zero-shot transfer across model scales and families by directly deploying the Qwen3-8B predictor on Qwen3-4B and Llama3.1-8B without retraining or recalibrating the decision threshold ($\tau = 6$). On Qwen3-4B and Llama3.1-8B, the transferred predictor achieves 115.6 and 118.3 tokens/s, respectively, or 114.3% and 93.7% of the throughput obtained with dedicated, model-specific predictors (Table 4). Because the features rely purely on prefix-matching statistics and cache state without accessing model hidden states, the learned selection boundary transfers effectively across model sizes and families.

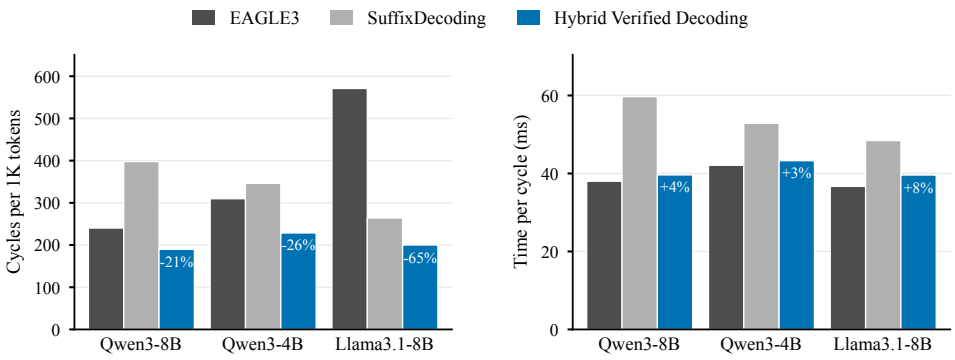


Figure 3: Execution breakdown on the three target models. Left: decoding cycles per 1K output tokens. Right: wall-clock time per decoding cycle. Labels on the Hybrid Verified Decoding bars give the change relative to EAGLE3. Lower values are better in both panels.

Table 5: Single-request serving throughput and speedup on Qwen3-8B inside vLLM (batch size 1, H100).

Dataset	EAGLE3 (tok/s)	SuffixDecoding (tok/s)	Hybrid Verified Decoding (tok/s)	vs. EAGLE3	vs. SuffixDecoding
MT-Bench	182.5	348.8	420.9	2.31×	1.21×
CNN/DailyMail	192.7	593.7	686.3	3.56×	1.16×

5.4 Speedup Breakdown: Cycles and Latency per Cycle

Decoding speedup comes down to two factors: verification cycles and latency per cycle. Figure 3 separates both factors on the three target models. Detailed measurements across all target models and hardware platforms appear in Appendix D and Appendix E.

Across all three target models, Hybrid Verified Decoding cuts verification cycles by 21% to 65% compared to EAGLE3. These savings come from the 9% to 19% of cycles that verify cache drafts. When accepted, these drafts commit long token sequences and replace multiple sequential EAGLE3 steps. Per-cycle latency stays within 3% to 8% of EAGLE3 across all models. Most cycles still run EAGLE3, the payoff predictor adds only 0.066 ms ($<$0.2% of a cycle), and the remaining difference comes from the cache lookup and the EAGLE3 catch-up pass, which processes newly committed tokens in one forward pass before drafting again.

SuffixDecoding verifies every cache draft without selection. On the Qwen3 models, cache drafts average under two accepted tokens. SuffixDecoding therefore needs more cycles than EAGLE3 and inflates per-cycle latency by 26% to 57%. As a result, it ends up slower than EAGLE3 on both models. Hybrid Verified Decoding avoids this penalty by skipping low-payoff drafts and verifying only the long ones.

5.5 Serving Prototype in vLLM

We evaluate Hybrid Verified Decoding inside the vLLM serving engine (Kwon et al., 2023) to test whether payoff-guided draft selection remains effective in a serving runtime. The prototype serves one request at a time (batch size 1) on an H100 GPU. It runs Qwen3-8B on MT-Bench and CNN/DailyMail with the payoff predictor and decision threshold ($\tau = 6$) from the offline experiments. As reported in Table 5, Hybrid Verified Decoding reaches 420.9 tokens per second on MT-Bench and 686.3 tokens per second on CNN/DailyMail. It is 2.31× and 3.56× faster than EAGLE3, and 1.21× and 1.16× faster than SuffixDecoding. Payoff-guided draft selection therefore retains its speedup outside the offline generation loop. Appendix H details the prototype implementation and serving parameters.

Table 6: Sensitivity to prompt template omission on Llama3.1-8B MT-Bench. Main results in Table 1 use official chat templates.

Prompt construction	Method	Mean accepted tokens ↑	Cycles / 1K tokens ↓	Throughput ↑
Template omitted	EAGLE3	1.20	455	55.6
Official template	EAGLE3	4.21	192	134.4
Template omitted	Hybrid Verified Decoding	7.32	120	194.2

5.6 Robustness to Prompt Formatting Shifts

All main evaluations follow the official chat template for each target model (Section 4.1). To evaluate draft selection when formatting degrades the model-based drafter, we conduct an isolated diagnostic on Llama3.1-8B MT-Bench. Table 6 compares decoding performance with and without the official Llama chat template.

When the template is omitted, EAGLE3 degrades sharply: accepted tokens drop from 4.21 to 1.20 per cycle, cycles per 1K tokens increase from 192 to 455, and throughput drops to 55.6 tokens per second. In contrast, Hybrid Verified Decoding yields 7.32 accepted tokens per cycle and reaches 194.2 tokens per second under the same template-omitted setting. Suffix matching operates on exact token sequences and does not depend on learned prompt formatting. Cache drafts therefore remain effective even when the model-based drafter degrades.

6 Conclusion

In this paper, we introduce Hybrid Verified Decoding, a payoff-guided method for state-dependent draft selection in speculative decoding. It predicts the accepted length of a cache draft before verification and uses this estimate to choose between cache verification and a model-based drafter. Across target models and task families, our experiments show improved end-to-end throughput and concentrated cache verification on high-payoff decoding states. These results point to a broader direction for speculative decoding, where runtime systems decide which drafts merit verification while continuing to build stronger drafters.

AI Use Statement

We used generative AI tools to assist with coding and experiments, to give feedback on experimental design, and to help interpret results. We also used them to polish the writing and to search the literature. The authors reviewed all AI-assisted work, verified all references, and take full responsibility for this paper.

References

Zihao An, Huajun Bai, Ziqiong Liu, Dong Li, and Emad Barsoum. Pard: Accelerating llm inference with low-cost parallel draft model adaptation, 2025. URL https://arxiv.org/abs/2504.18583.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5209–5235. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/cai24b.html.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling, 2023. URL https://arxiv.org/abs/2302.01318.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of LLM inference using lookahead decoding. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 14060–14079. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/fu24a.html.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily

Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. REST: Retrieval-based speculative decoding. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1582–1595, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.88. URL https://aclanthology.org/2024.naacl-long.88/.

Xinyi Hu, Yuhao Shen, Baolin Zhang, Hengxin Zhang, Jun Dai, Shuang Ge, Lei Chen, Yue Li, and Mingcheng Wan. Echo: Elastic speculative decoding with sparse gating for high-concurrency scenarios, 2026. URL https://arxiv.org/abs/2604.09603.

Yuxuan Hu, Ke Wang, Xiaokang Zhang, Fanjin Zhang, Cuiping Li, Hong Chen, and Jing Zhang. SAM decoding: Speculative decoding via suffix automaton. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12187–12204, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.595. URL https://aclanthology.org/2025.acl-long.595/.

Kaiyu Huang, Hao Wu, Zhubo Shi, Han Zou, Minchen Yu, and Qingjiang Shi. Adaspec: Adaptive speculative decoding for fast, slo-aware large language model serving. In *Proceedings of the*

2025 ACM Symposium on Cloud Computing, SoCC '25, pp. 361–374, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400722769. doi: 10.1145/3772052.3772239. URL https://doi.org/10.1145/3772052.3772239.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Rui Li, Zhaoning Zhang, Libo Zhang, Huaimin Wang, Xiang Fu, and Zhiquan Lai. Nightjar: Dynamic adaptive speculative decoding for large language models serving, 2026. URL https://arxiv.org/abs/2512.22420.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster inference of language models with dynamic draft trees. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7421–7432, Miami, Florida, USA, November 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.422. URL https://aclanthology.org/2024.emnlp-main.422/.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Forty-first International Conference on Machine Learning*, 2024b. URL https://openreview.net/forum?id=1NdN7eXyb4.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test, 2025. URL https://arxiv.org/abs/2503.01840.

Tianyu Liu, Qitan Lv, Yuhao Shen, Xiao Sun, and Xiaoyan Sun. Talon: Confidence-aware speculative decoding with adaptive token trees, 2026a. URL https://arxiv.org/abs/2601.07353.

Xiaoxuan Liu, Jiaxiang Yu, Jongseok Park, Ion Stoica, and Alvin Cheung. Speculative decoding: Performance or illusion?, 2026b. URL https://arxiv.org/abs/2601.11580.

Xianzhen Luo, Yixuan Wang, Qingfu Zhu, Zhiming Zhang, Xuanyu Zhang, Qing Yang, and Dongliang Xu. Turning trash into treasure: Accelerating inference of large language models with token recycling. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6816–6831, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.338. URL https://aclanthology.org/2025.acl-long.338/.

Gabriele Oliaro, Zhihao Jia, Daniel F Campos, and Aurick Qiao. Suffixdecoding: Extreme speculative decoding for emerging AI applications. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=uwL0vbeEVn.

Apoorv Saxena. Prompt lookup decoding, November 2023. URL https://github.com/apoorvumang/prompt-lookup-decoding/.

Jingwei Song, Xinyu Wang, Hanbin Wang, Xiaoxuan Lei, Bill Shi, Shixin Han, Eric Yang, Xiao-Wen Chang, and Lynn Ai. Mars: Unleashing the power of speculative decoding via margin-aware verification, 2026. URL https://arxiv.org/abs/2601.15498.

Woomin Song, Saket Dingliwal, Sai Muralidhar Jayanthi, Bhavana Ganesh, Jinwoo Shin, Aram Galstyan, and Sravan Babu Bodapati. Accelerated test-time scaling with model-free speculative sampling. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language*

Processing, pp. 30623–30636, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1558. URL https://aclanthology.org/2025.emnlp-main.1558/.

Minghao Yan, Saurabh Agarwal, and Shivaram Venkataraman. Decoding speculative decoding. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6460–6473, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.328. URL https://aclanthology.org/2025.naacl-long.328/.

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. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Situo Zhang, Hankun Wang, Da Ma, Zichen Zhu, Lu Chen, Kunyao Lan, and Kai Yu. Adaeagle: Optimizing speculative decoding via explicit modeling of adaptive draft structures, 2024. URL https://arxiv.org/abs/2412.18910.

A Experimental Details

All end-to-end evaluations use the same generation loop, prompt tokenization, target model, decoding budget, and hardware class for a given model-dataset setting.

Models. Table 7 lists the Hugging Face[2] checkpoints used for target models and EAGLE3 drafters.

Table 7: Target model and EAGLE3 checkpoints.

Model	Target model checkpoint	EAGLE3 checkpoint
Qwen3-8B	Qwen/Qwen3-8B	Tengyunw/qwen3_8b_eagle3
Qwen3-4B	Qwen/Qwen3-4B	AngelSlim/Qwen3-4B_eagle3
Llama3.1-8B	meta-llama/Llama-3.1-8B-Instruct	yuhuili/EAGLE3-LLaMA3.1-Instruct-8B

Generation Configuration. Table 8 lists the shared generation configuration. For the stochastic sampling evaluation in Appendix D, we set temperature to 0.6, top-p to 1.0, and the random seed to 0, with top-k disabled.

Table 8: Shared generation configuration.

Setting	Value
Batch size	1
Maximum new tokens	8192
Maximum sequence length	16384
Temperature	0.0
Data type	float16
Split seed	42

[2] https://huggingface.co/

Drafting Configuration. Table 9 lists the EAGLE3 and suffix-cache configuration. The threshold $\tau = 6$ was selected once on Qwen3-8B with the sweep in Appendix C and kept for every dataset and model.

Table 9: Drafting configuration.

Component	Setting	Value
EAGLE3	Total draft tokens	32
EAGLE3	Tree depth	8
EAGLE3	Top-k	4
Suffix cache	Maximum tree depth	64
Suffix cache	Maximum draft tokens	64
Suffix cache	Maximum speculative factor	1.0
Suffix cache	Maximum speculative offset	0.0
Suffix cache	Minimum token probability	0.1

Baselines. Greedy decoding uses the target model alone, without a drafter or a suffix cache. EAGLE3 uses the EAGLE3 checkpoint of the target model (Table 7) with the tree configuration in Table 9. SuffixDecoding proposes drafts from the suffix cache with the configuration in Table 9 and submits every draft for verification. Frequency-Based Hybrid Decoding follows the cache-first, EAGLE3-fallback structure of Hybrid Verified Decoding and verifies the cache draft when its frequency score from SuffixDecoding is at least 8. Hybrid Verified Decoding replaces this rule with the payoff predictor. The cache draft is selected when the predicted accepted length is at least 6. Predictor features, architecture, and training details are given in Appendix B.

Evaluation Metrics. Main end-to-end results are measured on H100 GPUs. Including trace preparation, payoff-predictor training, and the reported timing, diagnostic, and hardware runs, the experiments used approximately 50 GPU-hours. For each run, we record generated token count N, wall-clock decoding time T, and decoding-cycle count C. Throughput is computed as

$$\mathrm{Throughput} = \frac{N}{T}.$$

For a method m and baseline b evaluated on the same model, dataset, and prompt set, speedup is computed as

$$\mathrm{Speedup}(m, b) = \frac{\mathrm{Throughput}_m}{\mathrm{Throughput}_b}.$$

For execution analysis, we also report decoding cycles per 1K generated tokens,

$$\mathrm{CyclesPer1K} = 1000 \cdot \frac{C}{N},$$

and average time per decoding cycle,

$$\mathrm{TimePerCycle} = 1000 \cdot \frac{T}{C},$$

where $\mathrm{TimePerCycle}$ is measured in milliseconds. The aggregate metrics reported in the paper are computed from the recorded run outputs.

B Payoff Label Construction and Predictor Training

Payoff Label Construction. We construct training examples by replaying cache lookup on recorded target-model generations produced with greedy decoding. For each trace, we treat each token position t as a decoding state with prefix $x_{1:t}$. The suffix cache is queried with this prefix and returns a cache draft d_t^c. The label is the number of draft tokens that match the following tokens in the same trace:

$$y_t = \max\left\{\ell \in \{0, \ldots, |d_t^c|\} : d_{t,1:\ell}^c = x_{t+1:t+\ell}\right\}.$$

The recorded continuation therefore provides the accepted-length label for the cache draft, without additional target-model inference for each replayed draft.

Table 10: Dataset sources used to construct the evaluation prompt package.

Workload family	Dataset	Source
Agentic / workflow editing	Delegate-52	`microsoft/delegate52`
	InstructEdit/FineEdit	`YimingZeng/FineEdit_bench`
Code / repository	RepoBench	`tianyang/repobench_python_v1.1`
	SWE-bench Verified	`SWE-bench/SWE-bench_Verified`
	Magicoder	`ise-uiuc/Magicoder-Evol-Instruct-110K`
	MBPP	`google-research-datasets/mbpp`
Structured generation	BFCL	`llamastack/bfcl_v3`
	Spider	`NBAmine/xlangai-spider-with-context`
Multi-hop QA	HotpotQA	`hotpotqa/hotpot_qa`
	MuSiQue	`dgslibisey/MuSiQue`
	2WikiMultiHopQA	`Alab-NII/2wikimultihop`
Long-context	InfiniteBench	`xinrongzhang2022/InfiniteBench`
	CNN/DailyMail	`abisee/cnn_dailymail`
	GovReport	`ccdv/govreport-summarization`
Open-ended instruction / chat	Alpaca	`tatsu-lab/alpaca`
	MT-Bench	Official EAGLE GitHub repository (MT-Bench prompts)

Predictor Features. The predictor uses only signals available before verification. These features describe the cache match that produced the draft, the current decoding state, recent cache-draft behavior, and simple token patterns in the proposed draft. Table 11 groups these features into cache-match, decoding-progress, recent-history, and draft-structure signals. The feature set is designed to capture whether the current draft is likely to remain valid in the current decoding state, without using the target model's verification result for that draft.

Table 11: Payoff predictor feature groups.

Feature group	Description
Cache match	Frequency score, matched length, draft length, lookup time
Decoding progress	Prompt length, generated length, relative decode position
Recent history	Mean accepted length and first-token failure rate of recent cache drafts
Draft structure	Fractions of whitespace, punctuation, line breaks, brackets, and digits in the draft, and whether it contains an end-of-sequence token

Predictor Training. The predictor maps the pre-verification feature vector ϕ_t to a predicted accepted length:

$$\hat{y}_t = g_\theta(\phi_t).$$

We train it with squared-error regression:

$$\min_\theta \sum_t \left(g_\theta(\phi_t) - y_t\right)^2.$$

The predictor is a small multilayer perceptron. Table 12 lists the training configuration.

Runtime Use. At each decoding step, the same pre-verification features are computed for the current cache draft. Hybrid Verified Decoding selects the cache draft when the predicted accepted length is at least 6. Otherwise, it uses the model-based drafter. The target model and verification rule are unchanged. Algorithm 1 summarizes the complete decoding loop.

Table 12: Payoff predictor training configuration.

Setting	Value
Hidden layers	2
Hidden size	256
Activation	ReLU
Learning rate	0.001
Batch size	65536
Epochs	20
Seed	42

Algorithm 1 Hybrid Verified Decoding

Require: target model M, model-based drafter D_m, suffix cache $\mathcal{C}$, payoff predictor g_θ, threshold τ

Initialize the current prefix x with the input prompt
Initialize $\mathcal{C}$ from cache-warmup traces and the input prompt
Initialize the recent-history summary h
while generation is not finished **do**
    $d^c \leftarrow D_{\mathcal{C}}(x; \mathcal{C})$
    $\phi \leftarrow \Phi(x, d^c, \mathcal{C}, h)$, $\hat{y} \leftarrow g_\theta(\phi)$
    if $\hat{y} \geq \tau$ **then**
        $d \leftarrow d^c$
    else
        $d \leftarrow D_m(x)$
    end if
    $a, z \leftarrow \text{Verify}(x, d; M)$
    Append the committed block z to x
    Update $\mathcal{C}$ and h with z
end while

C Payoff Predictor Diagnostics

Feature Ablation. To isolate the contribution of each feature group, we train the predictor on four feature subsets and evaluate them on held-out Qwen3-8B cache draft candidates under $\tau = 6$ (Table 13). The cache match features and lookup time alone reach an F1 score of 67.5, against 68.8 for the full set, and account for most of the predictive signal. The remaining groups provide small but complementary gains: draft structure raises recall by 1.9 points at a cost of 0.6 points of precision, and decode state and recent history raise precision by 1.3 points. The full set attains the highest F1 score at a cost of 0.066 ms per state.

Table 13: Feature ablation of the payoff predictor on held-out Qwen3-8B cache draft candidates ($\tau = 6$). The first row uses the cache match features and lookup time. Each of the next two rows adds one feature group to the first row. The last row uses all features. The groups follow Table 11.

Feature set	Precision (%) ↑	Recall (%) ↑	F1 (%) ↑
Cache match and lookup time	77.9	59.5	67.5
+ Draft structure	77.3	61.4	68.4
+ Decode state and recent history	79.2	59.7	68.1
All features	78.2	61.4	68.8

Decision Threshold Sensitivity. The decision threshold τ determines which cache drafts proceed to verification. A low threshold admits short drafts that quickly fail, and a high threshold routes long drafts to EAGLE3. Figure 4 measures this trade-off on Qwen3-8B across ten thresholds from 1 to 16 through offline replay of decoding traces, where each step incurs the empirical mean cycle latency of

its selected draft source. The estimated speedup over EAGLE3 peaks at 1.328× at $\tau = 6$ and stays within 0.8% of this peak across $\tau \in [5, 10]$. Only 1.8% of cache drafts receive predicted lengths in this range, and any cutoff in this interval yields identical decisions on the other 98.2%. System throughput is therefore insensitive to the exact threshold value. The value $\tau = 6$ lies near the center of this plateau. We selected it once on Qwen3-8B and kept it for every dataset and model.

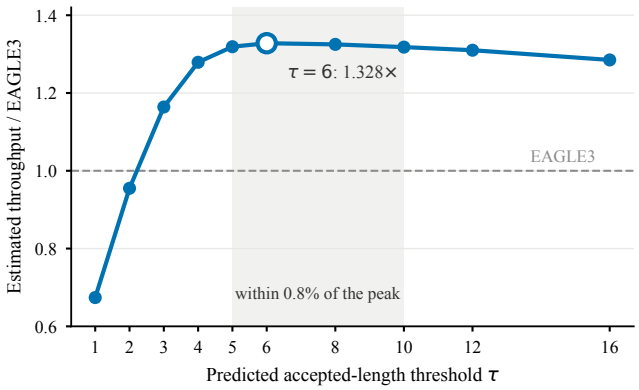


Figure 4: Threshold sensitivity of the payoff predictor on Qwen3-8B. Estimated speedup over EAGLE3 for ten thresholds from 1 to 16. The estimate stays within 0.8% of its peak from $\tau = 5$ to $\tau = 10$.

D Additional End-to-End Results

Tables 14–16 report aggregate measurements over the sixteen datasets. In addition to greedy decoding results across sixteen datasets, we evaluate decoding performance on Qwen3-8B in two further generation settings: decoding with thinking disabled and exact speculative sampling.

Table 14: Aggregate results for Qwen3-8B. Columns report throughput, cycles per 1K output tokens, mean accepted length, and draft-source fractions. Dashes mark fields outside the greedy path.

Method	Throughput tokens/s	Cycles /1K	Accepted tokens	Cache %	EAGLE3 %
Greedy	39.9	998.7	–	–	–
EAGLE3	109.5	240.2	3.16	0.0	100.0
Suffix	42.1	397.8	1.51	97.6	0.0
Frequency Hybrid	105.0	188.4	4.31	8.0	92.0
Hybrid Verified	133.0	189.7	4.27	9.7	90.3

Table 15: Aggregate results for Qwen3-4B, using the column definitions in Table 14.

Method	Throughput tokens/s	Cycles /1K	Accepted tokens	Cache %	EAGLE3 %
Greedy	38.3	1000.0	–	–	–
EAGLE3	76.7	309.7	2.23	0.0	100.0
Suffix	54.6	346.1	1.89	97.6	0.0
Frequency Hybrid	104.5	223.9	3.47	8.2	91.8
Hybrid Verified	101.1	228.6	3.37	8.8	91.2

Table 16: Aggregate results for Llama3.1-8B, using the column definitions in Table 14.

Method	Throughput tokens/s	Cycles /1K	Accepted tokens	Cache %	EAGLE3 %
Greedy	40.7	1000.0	–	–	–
EAGLE3	47.7	571.0	0.75	0.0	100.0
Suffix	78.2	264.0	2.79	93.9	0.0
Frequency Hybrid	98.3	234.9	3.26	9.4	90.6
Hybrid Verified	126.2	200.1	4.00	18.6	81.4

Thinking Disabled. To evaluate draft selection when the thinking mode of Qwen3 is off, we run Qwen3-8B with the official chat template under `enable_thinking=false` on InstructEdit/FineEdit, CNN/DailyMail, and MT-Bench (20 test prompts each). All runs directly reuse the payoff predictor and decision threshold ($\tau = 6$) from the main experiments. Because disabling thinking substantially shortens generations (for example, CNN/DailyMail total output drops from 11.5K to 3.7K tokens), absolute throughput is not directly comparable to Table 1. As reported in Table 17, Hybrid Verified Decoding is 2.66× faster than EAGLE3 on InstructEdit/FineEdit and remains within 3% of EAGLE3 on CNN/DailyMail and MT-Bench. Across the three datasets, it achieves an aggregate 1.75× speedup over EAGLE3.

Table 17: Throughput with thinking disabled (`enable_thinking=false`) on Qwen3-8B, 20 prompts per dataset. The total row divides all generated tokens by all generation time.

Dataset	EAGLE3 (tok/s)	Hybrid Verified Decoding (tok/s)	vs. EAGLE3
InstructEdit/FineEdit	79.61	211.96	2.66×
CNN/DailyMail	69.97	68.85	0.98×
MT-Bench	82.56	80.64	0.98×
Total (all three)	79.33	138.44	1.75×

Exact Speculative Sampling. We evaluate draft selection under stochastic sampling with temperature 0.6, top-p 1.0, and random seed 0. Because cache drafts lack model proposal probabilities, we apply exact speculative sampling following STAND (Song et al., 2025): cache tokens are accepted according to target-model probabilities, and rejected tokens trigger residual sampling to recover the target output distribution exactly. EAGLE3 uses its native rejection sampling. Hybrid Verified Decoding directly applies the predictor trained on greedy traces and the original decision threshold ($\tau = 6$). Across all three datasets, Hybrid Verified Decoding yields the highest throughput (Table 18). It is on average 2.81× faster than target-only decoding and 1.22× faster than EAGLE3. SuffixDecoding degrades on CNN/DailyMail and MT-Bench due to high cycle counts from low acceptance. Hybrid Verified Decoding skips these low-payoff cache drafts and matches the throughput of EAGLE3 on these two datasets.

Table 18: Exact speculative sampling (temperature 0.6, top-p 1.0) on Qwen3-8B, 20 prompts per dataset. Dashes mark fields outside the target-only path.

Dataset	Method	Throughput (tok/s)	Cycles / 1K tokens	Mean accepted tokens	Cache cycles (%)
InstructEdit/FineEdit	Target-only	33.98	1000.0	–	–
	EAGLE3	79.42	278.8	2.59	0.0
	SuffixDecoding	63.85	267.9	2.73	98.9
	Hybrid Verified Decoding	129.99	156.1	5.40	16.5
CNN/DailyMail	Target-only	43.30	1000.0	–	–
	EAGLE3	92.54	280.7	2.56	0.0
	SuffixDecoding	30.01	682.7	0.47	96.0
	Hybrid Verified Decoding	93.24	281.7	2.55	0.1
MT-Bench	Target-only	45.44	1000.0	–	–
	EAGLE3	108.68	249.5	3.01	0.0
	SuffixDecoding	32.14	684.4	0.46	89.2
	Hybrid Verified Decoding	111.34	240.0	3.17	1.9

E Cross-GPU Case Study

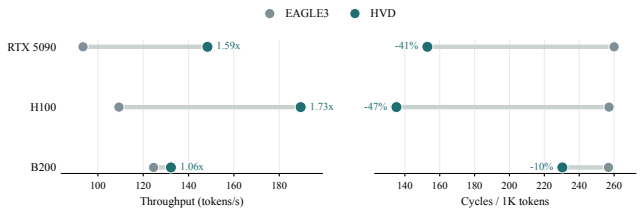
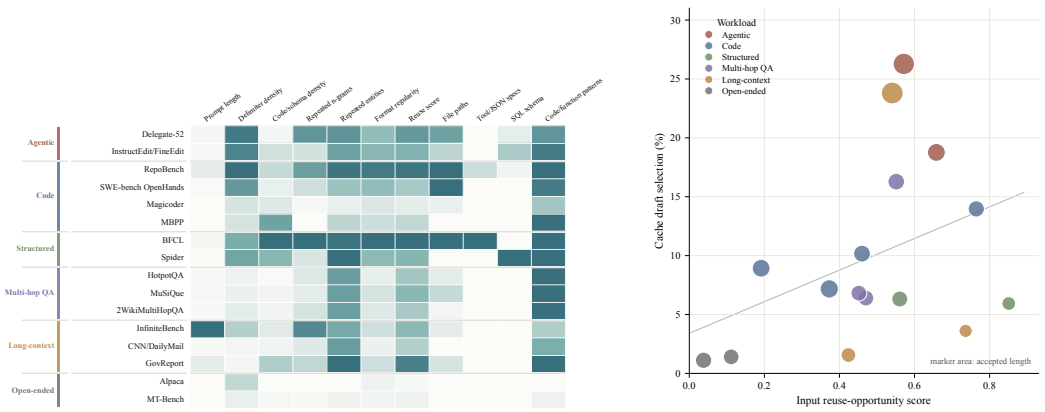


Figure 5: Cross-GPU case study on Qwen3-8B InfiniteBench. The experiment uses the same four prompts, prompt token limit, and decoding budget on RTX 5090, H100, and B200. Left: end-to-end throughput. Right: decoding cycles per 1K generated tokens.

We use a small cross-GPU case study to compare EAGLE3 and Hybrid Verified Decoding under the same setting: Qwen3-8B on InfiniteBench with the same four prompts and decoding budget. Figure 5 reports both end-to-end throughput and cycles per 1K generated tokens. Relative to EAGLE3, Hybrid Verified Decoding reaches 1.59× and 1.73× speedup on RTX 5090 and H100, where cycles per 1K tokens drop from 260.0 to 152.9 and from 257.1 to 135.2, respectively. On B200, the speedup is 1.06×, with cycles per 1K tokens decreasing from 256.8 to 230.3. These results suggest that Hybrid Verified Decoding is particularly effective on consumer and standard server GPUs, where reducing verification cycles can translate into substantial throughput gains.

F Prompt Structure and Cache Opportunity

Figure 6(a) reports normalized input-structure features by workload family. Agentic/workflow editing and code/repository workloads expose dense structural signals, including repeated entities, file paths, delimiters, and code/function patterns. Structured generation workloads show schema-like and tool-oriented formats, while long-context workloads provide extended context and recurring evidence. Open-ended chat prompts are sparser along these dimensions. Figure 6(b) shows that workloads with stronger prompt-side reuse structure trigger cache drafts more often: agentic/workflow editing and long-context tasks occupy the higher-selection region, and open-ended chat tasks remain near the lower-selection region. The same families have the highest share of high-payoff cache drafts in Figure 2.



(a) Prompt-side structure.

(b) Cache-draft selection.

Figure 6: Prompt-side cache opportunity and runtime cache-draft behavior. Panel (a) reports normalized input structure features by workload family. Darker cells indicate larger feature values after column-wise normalization. Panel (b) reports cache-draft selection rate against prompt-side reuse opportunity, with marker area scaled by mean accepted length.

G Dataset and Prompt Details

We evaluate on sixteen datasets grouped into six workload families. For every dataset, we fix 20 prompts for reported timing and analysis, giving 320 measured prompts per target model. Separate cache-warmup prompts from the same dataset are used to initialize the suffix cache and construct replay-based payoff labels. The test prompts are never used for cache warmup or predictor training.

For Hugging Face datasets, Table 10 reports the dataset identifier. For other sources, it reports the project or benchmark source used to construct the prompt package.

H vLLM Serving Prototype

We implement Hybrid Verified Decoding in the vLLM speculative decoding path. For each request, the serving path runs the selected draft source, verifies the proposed tokens with the target model, and updates the decoding state used by the payoff predictor. The predictor scores cache drafts before verification and selects cache verification when the predicted accepted length reaches the threshold. Otherwise, the request uses the model-based drafter.

The serving case study uses Qwen3-8B with its EAGLE3 checkpoint and the same payoff predictor used in the offline experiments. We run batch size 1 serving on H100 and evaluate MT-Bench and CNN/DailyMail under deterministic decoding. Table 19 summarizes the serving configuration.

Table 19: Configuration for the vLLM serving case study.

Setting	Value
Serving engine	vLLM
Target model	Qwen3-8B
Model-based drafter	EAGLE3
Cache drafter	Suffix cache
Predictor decision	Predicted accepted length ≥ 6
Batch size	1
Hardware	H100
Datasets	MT-Bench, CNN/DailyMail

I Artifact Licenses and Intended Use

We use publicly available model checkpoints, datasets, benchmarks, and code artifacts for research evaluation of decoding efficiency under their respective licenses and access terms. The Qwen checkpoints are released under Apache-2.0, and Llama3.1-8B-Instruct uses the Llama 3.1 Community License. The EAGLE repository and vLLM are released under Apache-2.0. Dataset licenses and source terms vary across benchmarks, including Apache-2.0, MIT, CC-BY, CC-BY-SA, CC-BY-NC, and CDLA-Permissive terms. Our code will be released under the MIT License.

J Analysis Metric Definitions

In Section 5.2 and Table 2, a proposed cache draft is an oracle high-payoff draft when its replayed accepted length is at least 6. A predictor-selected draft has predicted accepted length at least 6. Precision is the fraction of predictor-selected drafts that are oracle high-payoff drafts, and recall is the fraction of oracle high-payoff drafts selected by the predictor. Cache selection rate is the fraction of decoding cycles in which Hybrid Verified Decoding submits a cache draft for verification. Mean accepted length is averaged over verified draft cycles for the corresponding method or draft source.