AGENT TURNS ARE NOT CHAT TURNS: MATCHING DRAFT SUPPLY WITH VERIFICATION CAPACITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative Decoding (SD) accelerates LLM serving by verifying several draft tokens in one forward pass. Current SD approaches work well for conversation workloads, where contexts are short and content is largely new. Yet agent workloads are different: a coding agent solves tasks through tens of turns of generation, tool use and observation, building contexts that reach tens of thousands of tokens. Much of what the agent writes—re-edited files, re-issued commands, summarized context—closely follows text already in the prompt: 70% of generated tokens match the agent's own earlier output, and the reuse grows with the session. The prompt itself thus provides a growing candidate supply that needs no auxiliary model, but does need a retrieval-based drafter that can adapt it. Scaling retrieval-based drafting to concurrent serving raises two problems: building a per-request index blocks the scheduler thread, and the draft tree must be sized to the verification capacity the hardware affords. We present THRIFT, a training-free retrieval-based drafter that indexes the prompt with a suffix automaton and the current generation with a trie, builds each request's corpus asynchronously, and sizes the draft tree to the batch's verification capacity. We also propose an evaluation protocol that stratifies turns by their relation to the agent's earlier output and reports decode-stage next to end-to-end speedup. On BeyondSWE with Qwen3-14B at batch size 8, THRIFT achieves the best end-to-end speedup up to $3.9\times$ over autoregressive decoding; its accept length grows from 4.7 to 8.1 tokens per forward across a session while training-based baselines stay flat.

1 INTRODUCTION

Speculative decoding accelerates LLM inference by verifying several draft tokens in one forward (Leviathan et al., 2023). Current approaches work well for conversation workloads, where contexts are short and content is largely new. Agent workloads are different: a coding agent solves tasks through repeated cycles of generation, tool use and observation (Yao et al., 2023; Yang et al., 2024), building contexts that reach tens of thousands of tokens over dozens of turns. Much of what the agent writes—re-edited files, re-issued commands, summarized context—closely follows text already in the prompt; 70% of generated tokens match the agent's own earlier output (Section 2), and the reuse grows with the session. Retrieval-based drafting is a natural fit: the prompt itself provides a growing candidate supply that needs no draft model and no draft KV cache. Figure 1c shows how the cost of verifying draft tokens changes with batch size. The drafter must use the growing candidate supply within that hardware budget.

Retrieval drafting from the prompt is by now established (Saxena, 2023; He et al., 2024; Hu et al., 2025; Oliaro et al., 2025), but the reusable text has two distinct characters that one structure cannot serve well, and scaling to concurrent serving raises two further problems. First, the reusable text sits in two corpora with different character. The prompt holds tens of thousands of tokens from earlier turns; matches there are typically long spans that occur once. The current generation holds at most a few hundred tokens; matches there are short patterns that recur many times. The two corpora differ in scale, build cost, mutability and match frequency, so one indexing structure cannot serve both well (Section 3.1). Second, corpus management: a production n-gram drafter keeps one corpus per worker, so concurrent sessions overwrite one another's history and the per-session candidate supply

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.

degrades under concurrency (Section 3.1). A corpus per request is the natural fix, but each agent turn is a long prompt followed by a short burst of decoding, and building the index synchronously on the scheduler thread blocks every request in the batch. Third, the draft tree must be sized to the verification capacity the hardware affords. At high batch sizes the forward inflation of a wide tree exceeds what the candidate supply can repay; the width must track this trade-off, not just the drafter's accept rate.

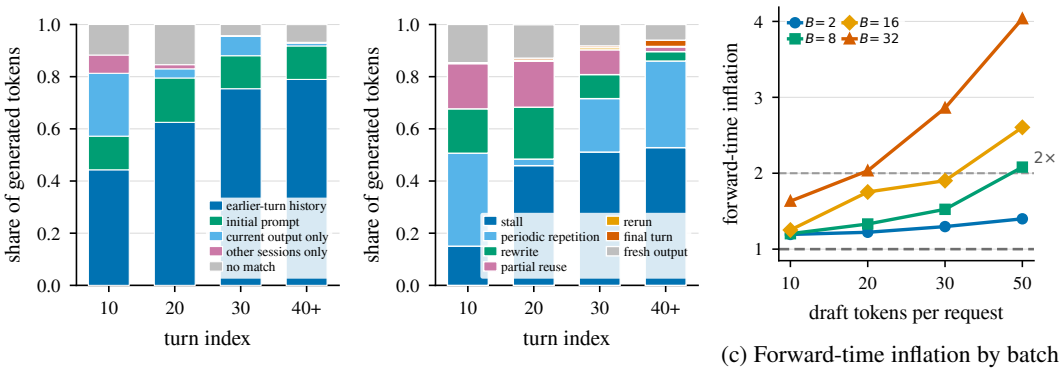
To address those issues, we present **THRIFT**, a training-free speculative decoder for coding-agent output. It uses a prompt suffix automaton to retrieve long copied spans and a generation Trie to retrieve short within-turn recurrences, then merges both sources into one scored draft tree. A profiled verification-cost bound and acceptance feedback adjust the tree width as the batch changes. Corpora are built while the prompt is prefilled and released off the scheduler thread, so a corpus per request does not block the decode path. Accept length grows from 4.72 to 8.11 tokens per forward across a session while the learned drafters' accept lengths change little. We also propose an evaluation protocol for agent speculative decoding that stratifies turns by their relation to earlier output, separates decode-stage from end-to-end speedup, and replays with production concurrency. On BeyondSWE (Chen et al., 2026a) with Qwen3-14B (Yang et al., 2025) at batch size 8, THRIFT achieves the best end-to-end speedup in 11 of 12 settings in the 16K–32K buckets, reaching $3.86\times$ over autoregressive decoding; the decode-stage margin over the stock n-gram drafter, $3.06\times$ against $2.94\times$, is clear while the end-to-end margin is within run-to-run noise. SuffixDecoding has already evaluated speculative decoding with concurrent coding agents (Oliaro et al., 2025). Our evaluation separates productive prompt reuse from stalls and periodic repetition, measures the two draft sources by turn class, and tests verification-cost-aware sizing under concurrent serving. Our contributions are:

- **Two complementary structures for two kinds of drafts.** A suffix automaton indexes the prompt for long single-occurrence spans and a trie indexes the current generation for short recurring patterns; one scored expansion merges them, and an ablation by turn class confirms each structure owns a different regime (Sections 3.1 and 4.4).
- **Scored expansion with reached-node acceptance estimates.** One breadth-first expansion pools candidates from both structures in score order, mixing corpus frequency with a Bayesian acceptance estimate that counts only verifier-reached edges in the denominator, avoiding the systematic underestimate from counting rejections at unaccepted parents (Sections 3.2 and C.2).
- **Verification-capacity-aware draft sizing.** A cost-aware floor constrains draft width using profiled forward inflation for the current batch, while an AIMD rule responds to acceptance as the text alternates between copied spans and fresh output (Sections 3.2 and 4.5).
- **An evaluation protocol for agent speculative decoding.** Our protocol stratifies turns by their relation to earlier output, excludes stalls and periodic repetition from the primary results, reports both decode-stage and end-to-end speedup under concurrent replay, and attributes accepted tokens to their source (Section 4.1).

2 MOTIVATION

Training-based drafters cannot exploit prompt-specific repetition: their accept length is flat at about 3 tokens per forward regardless of context, and one draft model's drops from 3.6 to 2.4 as the prompt grows from 8K to 32K (Li et al., 2025; Chen et al., 2026b), because a draft network's capacity is fixed at training time and cannot specialize to the session at hand.

Coding-agent prompts offer a different supply. We analyze 40 BeyondSWE instances recorded from Qwen3-14B with thinking disabled at temperature 0. In Figure 1a, the four-grams ending at 70.3% of generated tokens match earlier-turn history in the prompt; another 13.3% match the initial task, system or tool text. Reuse includes both recurring short patterns and long spans with a single earlier occurrence, motivating retrieval from both the prompt and current generation. Figure 1b shows that turns classified as fresh output account for only 8.9% of generated tokens. Reuse remains common throughout a session, although some comes from stalls and periodic repetition, which we exclude from the main evaluation (Section 4.1).



(a) Four-grams sources. (b) Turn categories by tokens. (c) Forward-time inflation by batch size.

Figure 1: Reuse and verification cost on BeyondSWE, Qwen3-14B, temperature 0. Turns are numbered from 0 within each session. Panel (c) plots verification-forward inflation against draft width for batch sizes 2, 8, 16 and 32, averaged over KV lengths below 20K tokens.

Whether these matches translate into wall-clock savings depends on the hardware cost of verifying draft tokens. In speculative decoding the decode-stage speedup is

$$S_{\text{decode}}(K) = \frac{\bar{\alpha}(K)\, T_{\text{AR}}}{T_{\text{prop}}(K) + T_{\text{ver}}(K)}, \tag{1}$$

where K is the draft width, $\bar{\alpha}(K)$ the mean tokens committed per verification forward, T_{AR} the one-token autoregressive forward time, $T_{\text{prop}}(K)$ the proposal time, and $T_{\text{ver}}(K)$ the verification time. For retrieval-based drafting, T_{prop} is a sub-millisecond lookup (Section 4.5), so the speedup is governed by the ratio $I(K) = T_{\text{ver}}(K)/T_{\text{AR}}$—the forward-time inflation. A net speedup requires $\bar{\alpha} > I(K)$. Figure 1c measures this inflation on Qwen3-14B (TP=2, two H100-80G) at batch sizes from 2 to 32: even at B=32, verifying 20 draft tokens costs about 2× a one-token forward, so the break-even accept length is just above 2 once proposal time is included. The budget varies with the batch's hardware state, so a fixed draft width either wastes available room or exceeds it; the width must adapt to the verification capacity the batch affords (Section 3.2), and the corpus must be built without blocking concurrent decoding (Section 3.3).

3 Method

THRIFT serves many coding-agent sessions concurrently; requests share the target model's decode batch and never share speculative state. Figure 2 illustrates one decoding step through three concerns: where candidates come from (Section 3.1), how they are sized to the hardware (Section 3.2), and what the per-request corpus costs (Section 3.3).

3.1 Candidate Sources: Prompt and Generation as Two Corpora

N-gram drafting fills a prefix-closed tree of K draft tokens by exact retrieval from a corpus; the target model verifies the tree and commits $\alpha \geq 1$ tokens per step. Section 2 identified two kinds of reusable text in two corpora: short recurring n-grams in the current generation (small, growing each step, frequency-rich) and long single-occurrence spans in the prompt (large, immutable, sparse). A single structure over both would either miss single-occurrence spans or dilute frequency signals across tens of thousands of tokens.

In the top row of Figure 2, the last 11 generated tokens (grey) also occur in the prompt, where a longer continuation (blue) follows them—this is the span the agent is about to copy. The orange underline marks the two most recent tokens, the short context the Trie will walk. The *prompt suffix automaton* (Blumer et al., 1985; Hu et al., 2025) indexes the prompt: it encodes every substring of the n prompt tokens in fewer than $2n$ states. During decoding the drafter tracks the state s of the longest suffix of the generation that also occurs in the prompt; each committed token either follows a transition, which lengthens the match, or follows suffix links to the longest shorter suffix that can

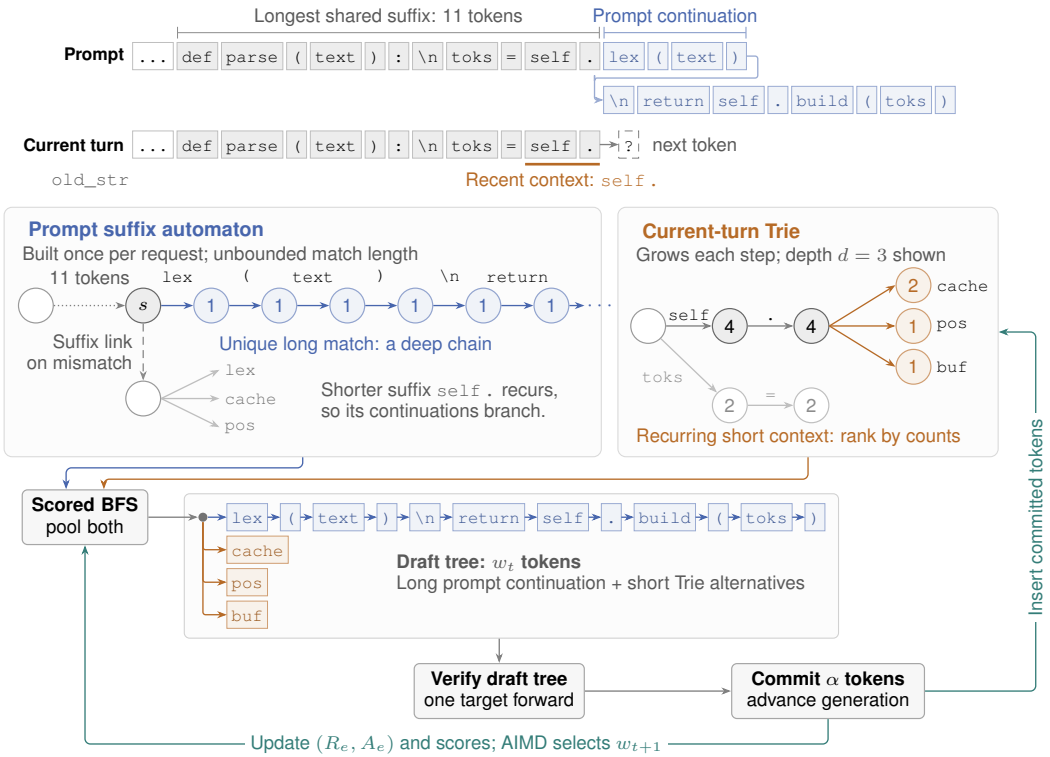


Figure 2: One decoding step of a `str_replace_editor` call. The agent copies code from an earlier tool result into `old_str`; tokens are drawn as words, counts are illustrative.

continue. In the left panel, state s is reached after 11 tokens; the unique match yields a deep chain of single-outgoing-transition states (the six nodes labeled 1), while a suffix link drops to a shorter, recurring suffix whose continuations branch. Transitions from s yield candidates; we write $P(c \mid v)$ for the conditional frequency of a candidate token c given the generation context v, estimated here from occurrence counts. Because the agent typically copies a prompt span verbatim, the match is long but sparse: it usually occurs once in the prompt, so the candidates form an unbranched chain. The *current-turn Trie* (Chen et al., 2025) indexes the generation: it inserts every token the model commits, together with its $d-1$ predecessors, into a depth-d tree and records a traversal count at each node. Walking the most recent tokens from the root reaches a node whose children supply $P(c \mid v)$; in the right panel, the context `self .` reaches a node with three children ranked by traversal count—a shallow fan, not a deep chain. Its context window is at most $d-1$ tokens, short enough to recur many times within a turn, so the counts separate likely continuations from unlikely ones. The Trie cannot replay long copied spans, but it captures short recurring patterns in the generated text that no earlier turn contains; its node pool is capped and recycled LRU. At each step the scored BFS (bottom left) queries both structures at the same context, pools their candidates into one frequency $P(c \mid v)$, and fills the draft tree in score order (Section 3.2): the resulting tree is a long spine from the automaton with short side branches from the Trie. Neither structure subsumes the other; Section 4.4 measures each alone and combined.

Both structures are scoped to the request. Earlier turns reach the automaton through the prompt; a corpus shared across concurrent requests, as in the stock drafter, mixes sessions and loses 41% of the per-session supply.

3.2 Hardware-Aware Draft Construction

Scored expansion with reached-node statistics. One breadth-first expansion builds the draft tree from both structures (the "Scored BFS" box in Figure 2). Each child c of a drafted node v is scored

$$\text{score}(c \mid v) = \tfrac{1}{2} P(c \mid v) + \tfrac{1}{2} \hat{q}_e, \qquad \hat{q}_e = \frac{A_e + 1}{R_e + 2}, \tag{2}$$

where $P(c \mid v)$ is the conditional frequency pooled over both sources, e is the edge from v to c (keyed by its three preceding tokens and the candidate), R_e counts the times the verifier reached e, and A_e the times c was then accepted. After each verification, committed tokens feed back to the Trie (the right arrow in Figure 2), and the reached/accepted counts (R_e, A_e) update at edges the verifier actually reached (the bottom arrow). Only reached exposures enter the denominator: counting rejections from an unaccepted parent systematically underestimates acceptance at depth two and beyond (Appendix C.2). The prior lets unseen edges be drafted; at request scope only 23% of nodes carry history, so the ranking rests mainly on the frequency term. Children are expanded in score order until the tree holds w_t tokens.

Width from verification budget. The decode-stage speedup (Equation 1) is governed by the forward-time inflation $I(K) = T_{\text{ver}}(K)/T_{\text{AR}}$, which depends on the batch's current memory and compute state rather than on the drafter. A fixed draft width either wastes available verification room or exceeds it as the batch changes, so the width must track the budget that the hardware affords. THRIFT pre-captures one CUDA graph at each level of a width ladder $\mathcal{L} = \{8, 16, 24, 32, 40, 50\}$; the draft width w_t is always drawn from $\mathcal{L}$. The same feedback path that updates (R_e, A_e) also selects w_{t+1} via an AIMD rule: when the mean acceptance $\bar{\alpha}$ exceeds 1, the width moves up by half a ladder gap; when $\bar{\alpha} \leq 1$, it drops one level. A cost-aware floor prevents levels whose profiled forward inflation exceeds $1.10\times$: the floor rises as the batch fills, keeping w_t inside the region where verification stays nearly free. Because agent text alternates between long copied spans (high acceptance) and fresh output (low acceptance) within a single turn, the reactive AIMD rule tracks the local regime without requiring a global accept-rate estimate. Section 4.5 reports the ladder's effect on accept length.

3.3 Request Lifecycle

The suffix automaton is built once per request (left panel annotation); the Trie grows each step (right panel annotation). Seeding the automaton from a 16K-token prompt takes about 6 ms on one core, and the scheduler thread is serialized with every decode step. Built synchronously, this cost 26.4 ms of admission overhead per request against 3.5 ms for the stock drafter—more than the higher accept length returned. The lifecycle therefore builds both structures during prefill on a native thread and queues release once a request leaves the batch. During decoding, the step only fetches candidates from the already-built structures and inserts committed tokens into the Trie (the right feedback arrow), both sub-millisecond operations. The Trie's node pool is capped at 16 384 nodes and recycled LRU, so memory stays bounded across long turns. Admission overhead falls to 4.4 ms per request, and this is where the wall-clock gain of Section 4.2 comes from.

4 Experiments

4.1 Setup

We tested our method as well as widely used SD baselines on coding-agent trajectories from BeyondSWE, which consists of four catalogs (Cross-Repo, DomainFix, DepMigrate, Doc2Repo) with Model Qwen3-14B/32B, a model series widely used for light weight model agent and suitable for all baselines, especially train-based ones. All experimental runs use SGLang 0.5.15.post1 (Zheng et al., 2024) on H100 GPUs, with Qwen3-14B at tensor parallel 2 (primary) and Qwen3-32B at tensor parallel 4, both with the native 40,960-token context and primary batch size 8. The agent trajectories are divided into buckets of 8K ($\leq$8K), 16K, 24K and 32K ($>$24K). We compare autoregressive decoding, two SGLang retrieval drafters (N-Gram over a worker-global Trie and a suffix automaton (Hu et al., 2025) over an external corpus), and two learned drafters, EAGLE-3 (Li et al., 2025) and DFlash (Chen et al., 2026b). The DFlash draft is retrained for one epoch on OpenPerfectBlend (Labonne, 2024) with SpecForge (Li et al., 2026b).

For a more truth-worthy presentation, all tables report a *cleaned view* that drops stalls and periodic self-repetition. Accept length (Acc) is tokens committed per target forward, bonus token included, so auto-regressive decoding sits at 1.00; S↑ is end-to-end speedup. Baseline parameters and the full evaluation protocol are described in detail in Appendix B.

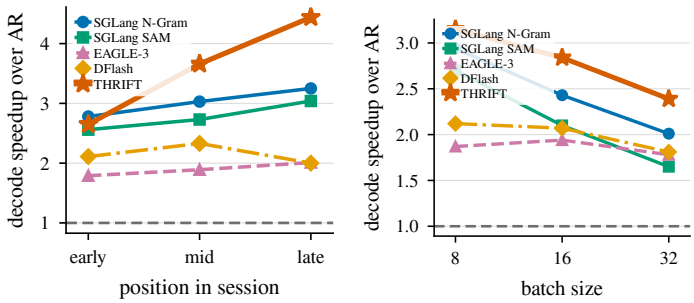
Table 1: End-to-end serving on four BeyondSWE catalogs, Qwen3-14B at temperature 0, batch size 8, cleaned view, mean over four GPU pairs. Acc is tokens committed per target forward; S↑ is request-level speedup over autoregressive decoding on the same pair. Best speedup per catalog and bucket in bold.

		8K			16K			24K			32K		
	Method	Acc	Tok/s	S↑	Acc	Tok/s	S↑	Acc	Tok/s	S↑	Acc	Tok/s	S↑
CrossRepo	Autoregressive	1.00	58.9	1.00	1.00	64.1	1.00	1.00	53.1	1.00	1.00	53.5	1.00
	SGLang N-Gram	5.37	87.1	**1.48**	6.12	118.2	1.84	3.71	85.9	1.57	6.45	94.5	1.82
	SGLang SAM	4.72	84.0	1.43	5.33	112.3	1.75	3.63	80.7	1.48	5.51	87.7	1.67
	EAGLE-3	2.79	76.6	1.30	3.08	99.0	1.55	2.84	70.7	1.29	2.77	77.7	1.47
	DFlash	3.58	79.4	1.35	3.43	98.8	1.54	2.43	66.2	1.21	2.47	70.2	1.36
	THRIFT	4.22	82.6	1.40	8.05	122.2	**1.91**	4.22	92.5	**1.69**	6.71	103.8	**2.00**
DomainFix	Autoregressive	1.00	53.6	1.00	1.00	52.6	1.00	1.00	65.4	1.00	1.00	71.7	1.00
	SGLang N-Gram	3.72	65.5	1.19	4.59	72.5	1.39	5.72	120.9	**1.78**	4.38	130.5	1.84
	SGLang SAM	3.86	67.0	**1.22**	4.47	72.1	1.38	5.09	113.5	1.68	4.32	127.4	1.79
	EAGLE-3	2.81	65.1	1.19	2.66	65.1	1.24	2.74	99.0	1.46	2.82	113.5	1.59
	DFlash	2.64	59.8	1.09	2.88	65.7	1.26	2.82	95.0	1.43	2.35	100.4	1.41
	THRIFT	2.43	61.0	1.11	4.91	72.9	**1.40**	5.74	117.7	1.71	5.83	136.4	**1.92**
DepMigrate	Autoregressive	1.00	59.8	1.00	1.00	69.6	1.00	1.00	45.0	1.00	1.00	61.1	1.00
	SGLang N-Gram	5.35	89.5	**1.50**	5.92	127.4	1.83	7.50	65.3	1.79	6.13	119.0	1.86
	SGLang SAM	4.78	86.9	1.45	5.30	121.0	1.74	6.68	64.4	1.76	5.10	108.3	1.71
	EAGLE-3	2.85	79.5	1.33	3.02	106.6	1.53	3.60	58.2	1.58	3.15	98.5	1.55
	DFlash	3.66	82.6	1.38	3.42	108.5	1.56	2.81	54.0	1.43	2.47	85.7	1.36
	THRIFT	4.45	85.4	1.43	8.03	129.8	**1.86**	22.50	67.1	**2.17**	7.93	130.9	**2.07**
Doc2Repo	Autoregressive	1.00	59.3	1.00	1.00	82.2	1.00	1.00	74.9	1.00	1.00	93.2	1.00
	SGLang N-Gram	4.10	79.6	**1.28**	3.65	148.9	1.78	5.34	160.2	2.13	10.73	296.9	3.18
	SGLang SAM	3.71	80.7	1.27	3.72	143.8	1.73	4.83	146.7	1.96	7.71	259.0	2.77
	EAGLE-3	2.58	74.5	1.20	2.56	124.3	1.50	2.67	121.7	1.63	2.97	172.3	1.85
	DFlash	3.30	79.3	1.21	3.23	134.2	1.62	3.23	125.0	1.70	2.85	160.4	1.73
	THRIFT	3.39	84.0	1.26	4.44	156.2	**1.87**	8.46	173.6	**2.33**	17.23	362.5	**3.86**

4.2 Main Results

We evaluate THRIFT on four BeyondSWE catalogs using Qwen3-14B at temperature 0 and batch size 8. Table 1 groups the results by catalog and prompt length, with four buckets labeled 8K, 16K, 24K, and 32K. Each bucket reports accept length (Acc), end-to-end generation throughput (Tok/s), and request-level speedup over autoregressive decoding (S↑). Acc counts the tokens committed per target forward, including the bonus token. Results exclude stalls and periodic self-repetition, following the protocol in Section 4.1.

THRIFT achieves the highest end-to-end speedup in 11 of the 12 settings at 16K and above, reaching 3.86× on Doc2Repo at 32K, compared with 3.18× for SGLang N-Gram (Table 1). Retrieval baselines lead in all four 8K buckets, with DomainFix at 24K the only remaining exception. Figure 3a complements this context-length comparison: THRIFT's decode-stage speedup rises from 2.65× in the first session third to 4.44× in the last, overtaking N-Gram by mid-session. N-Gram and



(a) By session third. (b) By batch size.

Figure 3: Decode-stage speedup, Qwen3-14B, T=0.

SAM improve more modestly, EAGLE-3 shows a small increase, and DFlash declines after mid-session. Together, these results show that THRIFT's advantage is strongest in longer contexts and later stages of a session.

THRIFT's growing advantage reflects both the reusable text it captures and the cost of verifying its drafts. The prompt suffix automaton retrieves long spans from accumulated session history, while the generation Trie captures short patterns emerging within the current turn. Together, they expose more reusable continuations, allowing successful drafts to commit more tokens per target forward and amortize verification cost. The cost-aware draft budget further limits verification overhead when longer drafts offer little additional benefit. By contrast, the learned drafters evaluated here use fixed parameters during inference. Without additional training or an explicit mechanism for incorporating session-specific text, they may struggle to translate newly accumulated history into comparable drafting gains.

4.3 Temperature, Model Scale and Batch Size

Table 2: Sampling temperature and model scale, batch size 8, cleaned view. "All" is every request; "32K" is the requests whose prompt exceeds 16K tokens. Each model and temperature runs on its own trajectories. Qwen3-32B uses tensor parallel 4: at tensor parallel 2 its KV pool retained only 42% of prompt tokens against 97% for Qwen3-14B. DFlash has no 32B draft.

	Qwen3-14B, $T = 0$				Qwen3-14B, $T = 0.3$				Qwen3-32B, $T = 0$			
	All		32K		All		32K		All		32K	
Method	Acc	S↑	Acc	S↑	Acc	S↑	Acc	S↑	Acc	S↑	Acc	S↑
Autoregressive	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
SGLang N-Gram	4.80	1.75	5.85	2.04	4.71	1.73	7.77	2.26	3.28	1.46	3.52	1.56
SGLang SAM	4.52	1.68	5.19	1.90	4.60	1.69	6.60	2.11	3.30	1.43	3.39	1.50
EAGLE-3	2.76	1.47	2.84	1.59	2.72	1.47	3.01	1.65	2.10	1.33	2.14	1.37
DFlash	3.08	1.48	2.72	1.50	3.26	1.53	3.32	1.66	2.84	1.47	2.92	1.54
THRIFT	5.61	**1.78**	8.18	**2.11**	5.63	**1.77**	8.07	**2.31**	3.54	**1.53**	4.17	**1.68**

To evaluate THRIFT's robustness across experimental settings, we vary the sampling temperature, model scale, and batch size. These experiments test whether its speedup gains persist beyond the default configuration. Table 2 compares Qwen3-14B at temperatures 0 and 0.3, together with Qwen3-32B at temperature 0, all at batch size 8. Each setting reports accept length and end-to-end speedup (Marked as S↑) for all requests and for the long-context subset labeled "32K" (prompts exceeding 16K tokens). Each model and temperature runs on its own trajectories. Figure 3b separately varies batch size from 8 to 32 and reports decode-stage speedup.

Raising the temperature from 0 to 0.3 leaves most of the overall speedup slightly increased (The agent trajectories on different tempuratures are different) Moving to Qwen3-32B reduces the gains of every evaluated method, but THRIFT retains the highest speedup at $1.53\times$ overall and $1.68\times$ on long prompts. With larger batches, THRIFT's decode-stage speedup declines less than those of the other retrieval baselines, widening its lead over SGLang N-Gram. Its accept length remains nearly unchanged, indicating that it continues to find reusable drafts even as serving conditions make speculation less profitable.

4.4 Draft Source Contribution

We measure how much each draft source contributes to THRIFT and whether combining them improves accept length. Table 3 compares the prompt suffix automaton, the generation Trie, and their scored union by turn content, and attributes accepted tokens to each source. The suffix automaton contributes most of the reused text from earlier turns: on rewrites it reaches 8.40 tokens per forward, and on reruns and stalls 87–94% of accepted tokens come from it alone. The Trie contributes patterns repeated within the current turn: it uniquely supplies 23.2% of accepted tokens on fresh output and 21.4% on periodic repetition. On fresh output, combining the sources raises accept length to 2.81, above either source alone. The scored union preserves most of their contributions: only 2.1%

of the tokens accepted by either single source are lost when they are combined. The weak region is fresh output, where both sources fall to about two tokens per forward against 8.69 over all turns (Table 3); that class covers 8.9% of generated tokens (Figure 1b), and the gains come from the large fraction of agent output that revisits the agent's own earlier text.

More detailed, Table 8 in Appendix C.3 shows the per-step accept distributions further distinguish the two sources' contributions. The Trie's 90th-percentile acceptance is one token per step, while the automaton's reaches the full 19-token draft.

Table 3: The two sources catch different drafts. Offline replay of the recorded Qwen3-14B temperature-0 trajectories at request scope, budget 20 per source. Tokens per forward for each source alone and for their scored union by turn class, then, of the completion tokens either single source accepts, the share accepted by exactly one of them and by both. The per-step accept distribution is in Appendix C.3.

	Tokens per forward			Share of accepted tokens (%)		
Turn content	Trie	SAM	Union	Trie only	SAM only	Both
Fresh output	1.89	2.12	2.81	23.2	31.3	45.5
Partial reuse	1.35	5.23	5.39	2.5	68.8	28.7
Rewrite (small edit)	1.23	8.40	8.40	0.9	78.7	20.4
Rerun (result changed)	1.05	9.84	9.84	0.1	94.4	5.5
Stall (no progress)	1.13	11.76	11.76	0.4	87.5	12.0
Periodic repetition	5.81	4.26	15.66	21.4	14.9	63.7
All turns	1.55	5.84	8.69	8.2	60.6	31.3

4.5 Hardware Cost Analysis

Table 4: Corpus lifecycle, Qwen3-14B, batch size 8, all turns, mean over four GPU pairs. Admission is the per-request cost of after-prefill steps beyond one forward, minus autoregressive decoding's; latency speedup is matched request latency against autoregressive decoding on the same pair.

	$T = 0$				$T = 0.3$			
Method	Adm. (ms)	Acc	Decode	Latency	Adm. (ms)	Acc	Decode	Latency
SGLang N-Gram	3.5	6.92	3.31	2.11	3.3	6.22	2.84	1.93
Ours, synchronous corpus	26.4	11.21	4.16	1.66	23.0	8.24	3.09	1.50
+ capped Trie pool, cost floor	17.3	11.59	4.59	1.98	15.3	8.74	3.49	1.79
Ours, asynchronous corpus	4.4	11.65	4.22	**2.28**	4.1	8.64	3.23	**1.99**
with fixed width 24	3.7	10.41	3.95	2.19	3.4	8.57	3.24	**1.99**

Longer accepted drafts improve speed only if they repay the cost of constructing and verifying them. We therefore separate corpus management costs from recurring drafting and verification costs. With synchronous construction, THRIFT accepts 62% more tokens per forward than SGLang N-Gram but achieves lower latency speedup ($1.66\times$ versus $2.11\times$; Table 4). Here, admission overhead outweighs the benefit of higher acceptance. Capping the Trie pool and adding the cost-aware floor reduces admission overhead from 26.4 to 17.3 ms; asynchronous construction and release further reduce it to 4.4 ms, raising latency speedup to $2.28\times$.

To isolate the adaptive width's contribution, we compare AIMD with fixed width 24 under the same asynchronous lifecycle. The fixed-width variant matches AIMD's latency on the cleaned view despite 12% lower accept length on all turns. The adaptive ladder primarily improves acceptance on high-repetition turns; the lifecycle and cost-aware floor drive the latency gains.

THRIFT's non-target time is 3.1–4.1 ms per step, below SGLang N-Gram in every bucket (Table 5). A full step costs 1.54–1.64 autoregressive forwards, requiring roughly 1.6 committed tokens to break even. CPU corpus construction takes 3.1–14.0 ms across prompt lengths, versus under 0.1 ms per step for retrieval and insertion (Appendix C.1), supporting its removal from the decoding path.

Table 5: Per-step cost at batch size 8, all turns, mean over four GPU pairs, in ms. Draft is the step minus the target forward and includes sampling, done inside the forward only for autoregressive decoding. Step is Draft plus Target over the autoregressive forward (the denominator of Equation 1).

	8K			16K			24K			32K		
Method	Draft	Target	Step	Draft	Target	Step	Draft	Target	Step	Draft	Target	Step
Autoregressive	–	9.03	1.00	–	9.44	1.00	–	10.32	1.00	–	10.54	1.00
SGLang N-Gram	3.68	11.40	1.67	3.46	10.76	1.51	4.41	11.47	1.54	4.87	11.86	1.59
SGLang SAM	3.84	11.60	1.71	3.75	11.07	1.57	4.65	11.73	1.59	5.12	12.13	1.64
EAGLE-3	4.42	8.83	1.47	4.44	9.08	1.43	4.53	9.74	1.38	4.56	9.98	1.38
DFlash	3.67	9.68	1.48	3.29	9.91	1.40	3.41	10.84	1.38	3.40	10.82	1.35
THRIFT	3.25	11.00	1.58	3.08	11.50	1.54	3.60	12.80	1.59	4.08	13.20	1.64

5 Related Work

Learned drafters. EAGLE-3 and DFlash are the strongest learned drafters available for our target models (Li et al., 2025; Chen et al., 2026b). Both run a draft network whose cost and KV cache grow with context, and in our replay their accept length is flat in context length while the retrieval baselines' rises.

Retrieval drafting. Prompt lookup decoding drafts from the current prompt (Saxena, 2023), LLMA from a reference document (Yang et al., 2023), REST from an external datastore (He et al., 2024), and N-Gram Trie speculative decoding from a trie over the input context (Chen et al., 2025). PLD+ ranks retrieved spans with target-model artifacts (Somasundaram et al., 2025), SAM-Decoding follows the longest suffix match through automata over a static corpus and the generated text (Hu et al., 2025). SuffixDecoding maintains a global suffix tree of prior outputs and a per-request tree of the current prompt and generation, updates the latter online, and evaluates eight concurrent SWE-Bench tasks (Oliaro et al., 2025). It adjusts draft length from the match length. THRIFT uses different structures for long prompt copies and short within-turn recurrences, and measures each source by agent turn class (Sections 3.1 and 4.4). Its primary results exclude stalls and periodic repetition (Section 4.1). THRIFT also builds the request corpus during prefill and constrains draft width using profiled verification cost for the active batch alongside acceptance feedback (Sections 3.3 and 3.2). Our SGLang suffix-automaton baseline uses an external corpus; the experiments do not include a direct SuffixDecoding comparison.

Hardware-aware verification. MagicDec and LongSpec exploit the compute slack of memory-bound long-context decoding (Sadhukhan et al., 2025; Yang et al., 2026); TurboSpec, AdaServe, BanditSpec and Nightjar adapt the speculative configuration to serving load or service objectives (Liu et al., 2025; Li et al., 2026c; Hou et al., 2025; Li et al., 2026a). These systems study hardware-aware control for general serving workloads. THRIFT combines a profiled verification-cost bound for the active batch with acceptance feedback on a fixed width ladder (Section 3.2). It applies this control to prompt and generation sources tailored to agent reuse, with a per-request corpus built during prefill.

6 Conclusion

Agent turns are not chat turns: the output is largely self-repetitive, and the text to be generated sits in a prompt ten to a hundred times longer than the decode burst that follows. THRIFT turns this asymmetry into a training-free speculative decoder that specializes in-context: a suffix automaton over the prompt for the long spans the prompt supplies, a Trie over the current generation for the short patterns that recur while it writes, one scored expansion that merges the two with accept statistics recorded only where the verifier reached, and a lifecycle that builds and releases each request's corpus off the decode path. On BeyondSWE replays through SGLang it wins most cells beyond 8K of context, its accept length grows across a session while learned drafters' accept lengths change little, and building each request's corpus off the scheduler thread drives the latency gain. Evaluating this regime took a protocol of its own, which we also contribute.

AI USE STATEMENT

We used generative AI coding agents to write the experiment and plotting scripts, and to tabulate and help interpret the measured results. No generative AI tool produced evaluation data. The same tools also drafted and edited parts of the text for readability, translated drafts into English, wrote the TikZ source of the method figures, and helped search the literature and check bibliography entries.

The authors reviewed all AI-assisted work: code changes were reviewed before any run entered the paper, and every reported number was traced to an exported result file whose run metadata the authors checked against the source logs.

REFERENCES

Anselm Blumer, Janet Blumer, David Haussler, Andrzej Ehrenfeucht, M. T. Chen, and Joel Seiferas. The smallest automaton recognizing the subwords of a text. *Theoretical Computer Science*, 40: 31–55, 1985. doi: 10.1016/0304-3975(85)90157-4.

Guoxin Chen, Fanzhe Meng, Jiale Zhao, Minghao Li, Daixuan Cheng, Huatong Song, Jie Chen, Yuzhi Lin, Hui Chen, Xin Zhao, Ruihua Song, Chang Liu, Cheng Chen, Kai Jia, and Ji-Rong Wen. BeyondSWE: Can current code agent survive beyond single-repo bug fixing? arXiv preprint arXiv:2603.03194, 2026a. URL https://arxiv.org/abs/2603.03194.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026b.

Jinglin Chen, Qiwei Li, Zuchao Li, Baoyuan Qi, Guoming Liu, Haojun Ai, Hai Zhao, and Ping Wang. Faster in-context learning for LLMs via N-gram trie speculative decoding. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 18040–18051. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.911. URL https://aclanthology.org/2025.emnlp-main.911/.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. 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)*, pp. 1582–1595. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.88. URL https://aclanthology.org/2024.naacl-long.88/.

Yunlong Hou, Fengzhuo Zhang, Cunxiao Du, Xuan Zhang, Jiachun Pan, Tianyu Pang, Chao Du, Vincent Y. F. Tan, and Zhuoran Yang. BanditSpec: Adaptive speculative decoding via bandit algorithms. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24045–24079. PMLR, 2025. URL https://proceedings.mlr.press/v267/hou25h.html.

Yuxuan Hu, Ke Wang, Xiaokang Zhang, Fanjin Zhang, Cuiping Li, Hong Chen, and Jing Zhang. SAM decoding: Speculative decoding via suffix automaton. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12187–12204, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.595.

Maxime Labonne. Open-PerfectBlend. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/mlabonne/open-perfectblend.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 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*, pp. 19274–19286. PMLR, 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. *Journal of Systems Architecture*, 178:103889, 2026a. doi: 10.1016/j.sysarc.2026.103889.

Shenggui Li, Chao Wang, Yikai Zhu, Yubo Wang, Fan Yin, Shuai Shi, Yefei Chen, Xiaomin Dong, Qiaoling Chen, Jin Pan, et al. SpecForge: A flexible and efficient open-source training framework for speculative decoding. arXiv preprint arXiv:2603.18567, 2026b. URL https://arxiv.org/abs/2603.18567.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Zikun Li, Zhuofu Chen, Remi Delacourt, Gabriele Oliaro, Zeyu Wang, Qinghan Chen, Shuhuai Lin, April Yang, Zhihao Zhang, Zhuoming Chen, Yi-Hsiang Lai, Xinhao Cheng, Xupeng Miao, and Zhihao Jia. AdaServe: Accelerating multi-SLO LLM serving with SLO-customized speculative decoding. In *Proceedings of the 21st European Conference on Computer Systems*, EuroSys '26, pp. 36–54, New York, NY, USA, 2026c. Association for Computing Machinery. doi: 10.1145/3767295.3769315. URL https://dl.acm.org/doi/10.1145/3767295.3769315.

Xiaoxuan Liu, Jongseok Park, Langxiang Hu, Woosuk Kwon, Zhuohan Li, Chen Zhang, Kuntai Du, Xiangxi Mo, Kaichao You, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. TurboSpec: Closed-loop speculation control system for optimizing LLM serving goodput. *arXiv preprint arXiv:2406.14066*, 2025. URL https://arxiv.org/abs/2406.14066.

Gabriele Oliaro, Zhihao Jia, Daniel Campos, and Aurick Qiao. SuffixDecoding: Extreme speculative decoding for emerging AI applications. In *Advances in Neural Information Processing Systems*, volume 38, pp. 126326–126354. Curran Associates, Inc., 2025. doi: 10.52202/085713-4211. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/b7aea253ab34a773967f1e4cdea9e4fb-Abstract-Conference.html.

Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian Yen, Avner May, Tianqi Chen, and Beidi Chen. MagicDec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. In *The Thirteenth International Conference on Learning Representations*, pp. 6835–6850, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/13f972adf12bdf886583d48cd528002f-Abstract-Conference.html.

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

Shwetha Somasundaram, Anirudh Phukan, and Apoorv Saxena. PLD+: Accelerating LLM inference by leveraging language model artifacts. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 6090–6104. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.338. URL https://aclanthology.org/2025.findings-naacl.338/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025. URL https://arxiv.org/abs/2505.09388.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R. Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652. Curran Associates, Inc., 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. Inference with reference: Lossless acceleration of large language models, 2023.

Penghui Yang, Cunxiao Du, Fengzhuo Zhang, Haonan Wang, Tianyu Pang, Chao Du, and Bo An. LongSpec: Long-context lossless speculative decoding with efficient drafting and verification. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1826–1844. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.83. URL https://aclanthology.org/2026.acl-long.83/.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-2000. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/724be4472168f31ba1c9ac630f15dec8-Abstract-Conference.html.

A NOTATION

Table 6 lists the symbols used in the paper.

Table 6: Notation used in the paper.

Symbol	Meaning
Speculative step	
K	Draft width: number of draft tokens submitted for verification in one step
α, $\bar{\alpha}(K)$	Tokens committed by one speculative step (bonus token included), and its mean at width K
T_{AR}	Time of a one-token autoregressive decode forward
$T_{\mathrm{prop}}(K)$, $T_{\mathrm{ver}}(K)$	Time to build and to verify a K-token draft tree
$S_{\mathrm{decode}}(K)$	Decode-stage speedup, Equation 1
Sources and scoring	
d	Maximum depth of the current-turn Trie; a committed token is inserted with its $d-1$ predecessors
v, c	A drafted node and one of its candidate children
e	The edge from v to c, identified by the three tokens preceding c and c itself
$P(c \mid v)$	Conditional frequency of c after v, pooled over the suffix automaton and the Trie
R_e, A_e	Times the verifier reached edge e (its parent was accepted) and times it then accepted c
$\hat{q}_e$	Smoothed reached-node accept rate $(A_e+1)/(R_e+2)$, Equation 2
Width control	
$\mathcal{L}$	Width ladder $\{8, 16, 24, 32, 40, 50\}$; one CUDA graph is captured per level
w_t	Batch-wide draft width at decode step t, drawn from $\mathcal{L}$
Roofline	
$z = (B, L)$	Serving state: active batch size B and prefix-length profile L
τ_K^{mem}, τ_K^{comp}	Memory and compute time of a verification forward at width K in a fixed state z
$K^*_{\mathrm{MB}}(z)$	Largest width that remains memory-bound in state z

B EXPERIMENTAL DETAILS

Protocol. Five rules make the comparison faithful to agent serving. (i) Batch k of the replay holds turn k of every instance, so a worker-global corpus is shared across instances exactly as in production. (ii) Every turn is classified by its relation to the agent's earlier outputs and the tables use a cleaned view (details below). (iii) End-to-end speedup, request tokens per second against autoregressive decoding with prefill included, is reported next to decode-stage speedup from synchronized runs (details under Timing). (iv) Position in a session is the tertile of the instance's own turns, so long looping sessions do not dominate late buckets. (v) Accepted draft tokens are attributed to their source (Section 4.4). A request stalling beyond ten seconds in any method is dropped from every method; cells with fewer than five requests are `n/a`.

Methods. Autoregressive decoding is the reference. The SGLang N-Gram drafter uses PROB matching over the worker-global Trie with 20 draft tokens. The SGLang SAM drafter adds an external suffix automaton over the task statements of the benchmark with a SAM budget of 10 inside the same 20-token draft. EAGLE-3 runs five draft steps with top-1 expansion and six draft tokens; three steps was slower ($2.19\times$ against $2.28\times$ decode speedup). DFlash is a 14B draft retrained with SpecForge (block size 16) and served with SGLang's default draft attention; forcing the Triton backend cut its decode speedup from $2.16\times$ to $1.35\times$. THRIFT runs SCORED matching with the width ladder $\{8, 16, 24, 32, 40, 50\}$, up-step fraction 0.5, down threshold 1, cost-aware floor at inflation 1.10, a request Trie pool capped at 16,384 nodes, and the asynchronous corpus lifecycle; its source scope is the request only, so earlier turns reach it through the prompt.

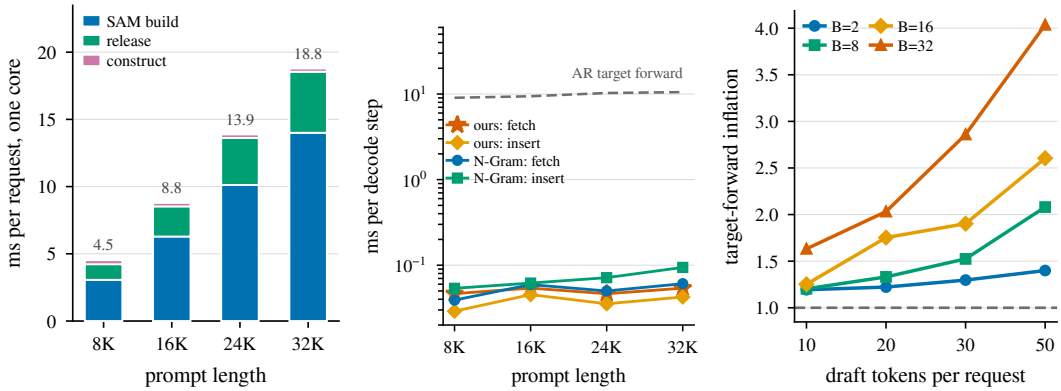
Cleaned view. A turn is a *stall* when it reissues an earlier tool call of the same instance and the tool result is unchanged after masking digit runs, hexadecimal strings of seven or more characters

and `0x` object addresses; it is *periodic repetition* when its completion is an insert-only extension of the previous call for three or more consecutive turns. The cleaned view drops both classes and keeps reruns whose result changed and edits of earlier outputs. At temperature 0 it retains 1,252 of 2,311 turns and at temperature 0.3 839 of 1,955.

Timing. End-to-end speedup uses untimed runs. Decode-stage speedup and the per-step costs of Table 5 come from separate runs with per-step device synchronization, which compare correctly with each other but not with untimed throughput. A batch that takes more than ten seconds in any method is dropped from every method.

C Additional Results

C.1 What a Per-Request Corpus Costs to Build



(a) Per-request corpus cost. (b) Per-step cost (log scale). (c) Cost of a wider draft.

Figure 4: What a per-request corpus costs. (a) CPU microbenchmark on one core, 40 sampled turns per bucket: building the suffix automaton from the prompt dominates. (b) Per-step fetch and insert against the target forward they must hide inside. (c) Target-forward inflation from the decode-forward profile at tensor parallel 2, at KV lengths below 20K.

Figure 4a gives the once-per-request cost: 4.5 ms at 8K rising to 18.8 ms at 32K, of which seeding the automaton from the prompt is 3.1 to 14.0 ms and releasing the corpus 1.2 to 4.5 ms; constructing the objects themselves costs about 0.25 ms at every length. Panel (b) covers the recurring cost: fetching a draft costs 0.046 to 0.054 ms per step and recording verification outcomes 0.029 to 0.045 ms, against a target forward of 9.0 to 10.5 ms, and both are flat in prompt length since the automaton is built once. Upstream SGLang's worker Trie fetches in the same range, 0.039 to 0.061 ms, but its insert cost grows with context, from 0.054 to 0.094 ms. Panel (c) shows the price of verification: at batch 2 a 50-token draft inflates the forward by $1.40\times$, at batch 32 by $4.04\times$.

C.2 Effect of the Acceptance Estimate on Draft Quality

The scored expansion of Section 3.2 mixes corpus frequency $P(c \mid v)$ with a Bayesian accept estimate $\hat{q}_e$. Table 7 removes the estimate by setting its weight to zero, ranking children by frequency alone.

Table 7: Offline tokens per forward at request scope, budget 20, Qwen3-14B temperature 0, capped at 256 tokens per request to match the GPU runs. "Freq-only" sets $accept_weight = 0$; "Scored" is the paper's $\frac{1}{2}\,P + \frac{1}{2}\,\hat{q}$ mix.

	Tokens per forward	
Config	All turns	Cleaned view
Freq-only (P alone)	8.63	5.08
Scored ($\frac{1}{2}P + \frac{1}{2}\hat{q}$)	8.64	5.08
AIMD (scored + ladder)	—	4.78
Trie only	—	1.20
SAM only	—	4.54

The two rankings produce the same draft quality: mean tokens per forward differs by less than 0.02 on all turns and is identical on the cleaned view. At request scope only 23% of edges carry acceptance history, so the prior dominates $\hat{q}_e$ and the ranking rests on the frequency term. The reached-node counting described in Section 3.2 therefore contributes calibration accuracy—it avoids

the systematic underestimate that arises when unaccepted parents inflate the denominator—but does not change aggregate draft quality at the scope and budget the system uses.

C.3 Accept Distribution per Verify Step

Table 8: Accepted draft tokens per verify step, offline replay at request scope, Qwen3-14B temperature 0, budget 20 per source. The ladder allows up to 49.

Configuration	Mean	p50	p90	Steps ≥ 8 (%)
Trie only	0.55	0	1	3.4
SAM only	4.84	0	19	29.6
Scored union	7.69	6	19	48.2
Scored union + AIMD ladder	7.97	4	23	33.5