WHEN MISMATCHES OVERLAP: CONCURRENT VERIFICATION FOR SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding (SD) accelerates large language model inference by verifying multiple draft tokens in a single target-model pass. Strict verification stops at the first mismatch and discards all remaining drafts, even though the target model has already computed predictions for later positions. Deferred verification addresses this waste by treating subsequent target outputs as evidence: instead of rejecting a mismatch immediately, the verifier continues and resolves it based on what follows. However, a later mismatch can arise before an earlier one is resolved, leaving multiple verification decisions pending simultaneously. Existing methods focus on the earliest pending mismatch: a later mismatch triggers rejection instead of becoming an independent verification decision, limiting draft reuse. We introduce PEND, a concurrent verifier that maintains a separate state for each pending mismatch while sharing subsequent target predictions across them. PEND adapts evidence requirements to target-relative support, weights subsequent observations by target confidence, and commits a draft segment only after all pending mismatches resolve. Using only cached target outputs from the current verification pass, PEND improves throughput by 46.6-70.3% (57.2% on average across the evaluated settings) relative to vanilla SD across two model families and four benchmarks, while keeping task quality close to strict verification and achieving better quality-efficiency trade-offs than recent deferred-verification baselines.

1 INTRODUCTION

Speculative decoding (SD) accelerates autoregressive large language model (LLM) inference by using a lightweight draft model to propose multiple tokens and a larger target model to verify them in parallel (Leviathan et al., 2023; Xia et al., 2023; Chen et al., 2023). Under strict verification, the decoder stops at the first mismatch: it accepts all preceding draft tokens, emits the target prediction at the mismatch position, and discards the remaining draft suffix. Because the target has already computed predictions for every later draft position in the same forward pass, this first-mismatch rule leaves already-computed target information unused.

Relaxed verification methods allow some mismatches to survive strict rejection. One approach uses richer evidence available at the mismatch itself, such as learned validity signals or target-model confidence (Bachmann et al., 2025; Song et al., 2026). While more permissive than exact matching, these methods still resolve each mismatch locally, without considering subsequent target predictions. Deferred-verification methods postpone mismatch resolution and treat subsequent target outputs as additional evidence (Li et al., 2026; Wang et al., 2026a). This continuation strategy brings already-computed post-mismatch information into the verification decision. When the evidence is strong enough, a deferred mismatch can be accepted; otherwise, it remains pending or gets rejected. Nevertheless, deferral introduces a new challenge: another mismatch may arise before an earlier one resolves, leaving multiple verification decisions pending simultaneously.

This concurrent challenge is not an edge case. As shown in Figure 1, multiple mismatches remain simultaneously unresolved in 10.39% of verification spans on HotpotQA and 15.00% on HumanEval+, affecting 25.60% and 28.04% of processed token positions, respectively (Yang et al., 2018; Liu et al., 2023). Tracking only the earliest pending mismatch yields a similar mean number of emitted tokens per draft round (MAT), but it leads to lower task quality than independently tracking all pending mismatches throughout verification.

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.

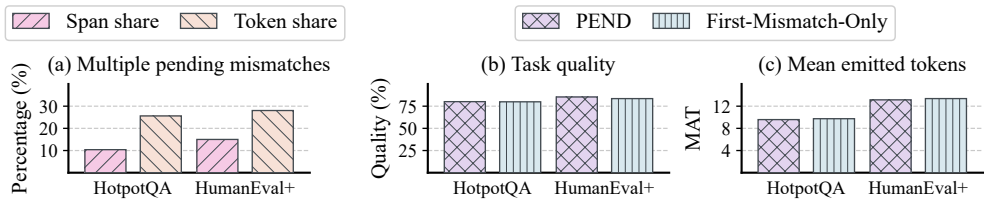


Figure 1: Concurrent pending mismatches occur in practice, and mismatch-specific tracking matters. (a) Under the evaluated continuation setting, the fractions of verification spans and processed token positions containing multiple simultaneously pending mismatches. (b, c) Full PEND versus First-Mismatch-Only: tracking only the earliest pending mismatch yields lower task quality at similar MAT. Quality is measured by answer F1 on HotpotQA and Pass@1 on HumanEval+, respectively.

Existing deferred-verification methods do not address how to handle multiple mismatches that remain pending simultaneously (Li et al., 2026; Wang et al., 2026a). This concurrent setting poses three coupled challenges. First, mismatches differ in how strongly the target supports the draft token relative to its top choice. A draft token far below the target's top choice should require stronger continuation evidence than one that is nearly competitive, but how should the required evidence scale with this disagreement? Second, subsequent predictions differ in informativeness. A target distribution with a large separation between its top two choices is more decisive than one whose leading choices are nearly tied, yet a binary match signal treats them identically. Third, later observations may simultaneously inform multiple pending mismatches. Because these mismatches arise at different positions and accumulate different evidence histories, they may require different amounts of evidence and resolve at different times. A concurrent verifier must therefore account for mismatch-specific verification burdens and observation-specific informativeness while coordinating overlapping decisions.

We introduce PEND to address concurrent mismatch verification. PEND tracks each deferred mismatch separately while sharing subsequent target evidence across pending mismatches. *Target-support-adaptive mismatch verification* uses target-relative support to determine whether a mismatch should be deferred and how much continuation evidence it requires. Mismatches with weaker target-relative support face a higher verification burden. Once deferred, *confidence-weighted evidence accumulation* weights subsequent target predictions by the separation between the target's top two choices, allowing more decisive observations to contribute more strongly. When several mismatches remain pending, *mismatch-specific concurrent resolution* maintains a separate evidence history and completion condition for each while allowing the same later observation to update multiple earlier pending mismatches. This separation-with-sharing design lets overlapping mismatches resolve independently and at different times. PEND commits a draft segment only after all pending mismatches resolve. If any remain unresolved when the fixed continuation window ends, PEND rejects the segment. PEND operates entirely on target outputs from the current verification pass, requiring neither additional forward passes nor a learned verifier.

In summary, our contributions are threefold. *First*, we identify and characterize concurrent pending mismatches as a distinct challenge in deferred verification, and show that tracking only the earliest pending mismatch yields similar MAT but lower task quality. *Second*, we introduce PEND, a concurrent verifier that tracks pending mismatches separately while sharing subsequent target evidence across them, allowing overlapping mismatch decisions to resolve independently. *Third*, across two draft-target model families and four benchmarks, PEND improves throughput over vanilla SD by 46.6-70.3% (57.2% on average) while maintaining task quality close to strict verification and achieving better quality-efficiency trade-offs than recent deferred-verification baselines.

2 Related Work

Early SD methods adopt strict verification, where draft reuse ends at the first mismatch (Leviathan et al., 2023; Xia et al., 2023; Chen et al., 2023). Subsequent work has relaxed this boundary in two ways. Distribution-preserving methods broaden or restructure speculative acceptance while retaining exact target-distribution guarantees. Representative approaches include block- and tree-

based verification (Sun et al., 2025; Weng et al., 2025), optimal-transport-based acceptance (Sun et al., 2023), and multi-draft verification (Sun et al., 2024; Hu et al., 2025; Khisti et al., 2025). Such guarantees are important when exact sampling fidelity is required, but they also constrain how freely mismatching draft tokens can be retained. Relaxed verification instead allows departures from exact distributional equivalence in exchange for more flexible draft reuse and potentially higher decoding efficiency. PEND belongs to this relaxed-verification family. We organize relaxed verifiers along a key axis: whether a mismatch is resolved using only evidence available at that position (local-evidence relaxed verification) or additionally using subsequent target outputs (continuation-evidence relaxed verification).

Local-Evidence Relaxed Verification. Local-evidence methods resolve each mismatch using only information available at the mismatch position. Learned approaches introduce supervised or learned verification mechanisms (Bachmann et al., 2025; Garipov et al., 2025; Yoon et al., 2026; Wang et al., 2026b). Training-free approaches instead derive verification rules directly from model outputs, using signals such as draft-target distribution divergence, probability thresholds, decision margins, alignment information, or draft-to-top-1 probability ratios (Kim et al., 2023; Holsman et al., 2025; Wang et al., 2025a; Song et al., 2026; Urbán et al., 2026). Although these signals differ, verification remains position-local: target outputs at later positions do not enter the decision for that mismatch.

Continuation-Evidence Relaxed Verification. Continuation-evidence methods defer mismatch resolution, allowing subsequent target outputs to contribute additional evidence. FLy (Li et al., 2026) uses an entropy gate to determine whether a mismatch may be deferred and then evaluates a fixed lookahead window. Crucially, when a later mismatch is encountered, FLy rejects the earlier deferred mismatch rather than creating a second pending verification state. LinguaSpec (Wang et al., 2026a) instead adapts probability thresholds and verification-window lengths using linguistic categories such as content and function words. Neither formulation addresses the case where a later mismatch arises before an earlier one has resolved: multiple verification decisions are then simultaneously pending, yet no separate mismatch-specific evidence states are maintained. Prior studies also exploit future or nonlocal information through future-prefix lookahead, retrospective assessment of draft continuations, and semantic- or reasoning-level signals (Kingetsu et al., 2026; Wang et al., 2025b; Fu et al., 2025). By contrast, PEND separates decisions while sharing evidence: each pending mismatch maintains its own evidence history and completion condition, while the same causally subsequent cached target outputs may update multiple earlier pending mismatches.

3 Problem Statement

Given a context x and a token vocabulary $\mathcal{Y}$, the draft model proposes a length-K candidate span $d_{1:K} = (d_1, \ldots, d_K) \in \mathcal{Y}^K$, where $K \geq 1$. For any positive integer n, let $[n] := \{1, \ldots, n\}$. For $1 \leq a \leq b \leq K$, write $d_{a:b} := (d_a, \ldots, d_b)$, and for $t \in [K+1]$, write $d_{<t} := d_{1:t-1}$, with $d_{<1}$ denoting the empty sequence. After accounting for the remaining generation budget and any end-of-sequence token produced by the draft model, let $V \in [K]$ denote the valid verification length, so that $d_{1:V}$ is the valid candidate span.

Let $p_T(\cdot \mid \cdot)$ denote the target model's next-token conditional distribution. A single target-model verification pass produces, for each $t \in [V+1]$,

$$q_t(\cdot) := p_T(\cdot \mid x, d_{<t}). \tag{1}$$

In particular, $q_{V+1}(\cdot) = p_T(\cdot \mid x, d_{1:V})$ is the next-token distribution conditioned on the full valid candidate span. For each $t \in [V+1]$, let $y_t^\star \in \mathcal{Y}$ be the token selected from $\arg\max_{v \in \mathcal{Y}} q_t(v)$ by a fixed deterministic tie-breaking rule.

Under greedy verification, define the mismatch set $\mathcal{M} := \{t \in [V] : d_t \neq y_t^\star\}$. If $\mathcal{M} = \varnothing$, strict verification accepts all V draft tokens and additionally emits $y_{V+1}^\star$ as a bonus token. Otherwise, let $m_0 < \cdots < m_{J-1}$ denote the elements of $\mathcal{M}$, where $J := |\mathcal{M}|$. Strict verification stops at the first mismatch m_0: it accepts $d_{<m_0}$, emits $y_{m_0}^\star$, and discards the remaining draft suffix $d_{m_0:V}$.

Importantly, although strict verification stops draft reuse at m_0, the same target-model pass has already computed q_t for every $t \in [V+1]$. For any $t > m_0$, Equation (1) shows that q_t remains conditioned on the draft prefix $d_{<t}$ rather than on a prefix in which an earlier mismatch has been corrected. We refer to these post-mismatch distributions as *cached target information* along the current draft trajectory.

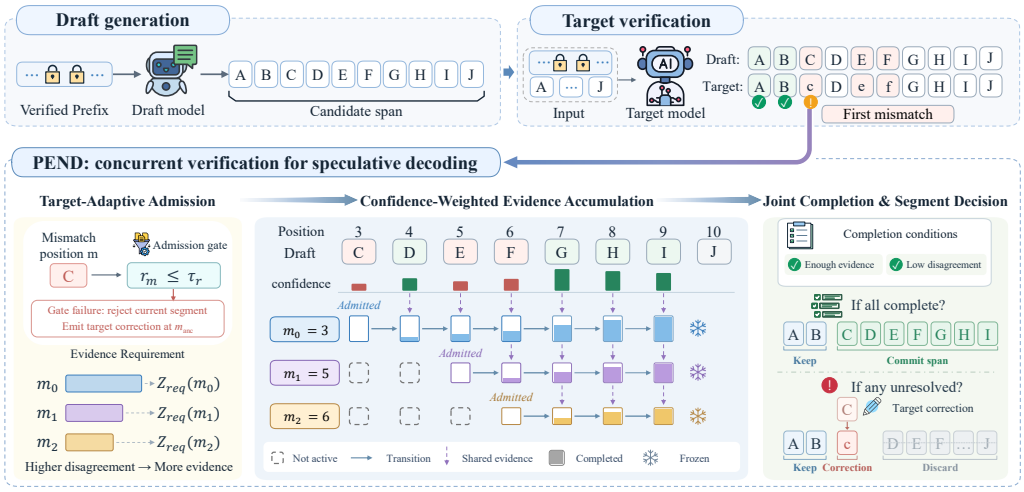


Figure 2: Overview of PEND within a single verification pass. Starting from the first mismatch after the accepted prefix, PEND anchors an unresolved segment, admits subsequent mismatches based on target-relative support, and assigns mismatch-specific evidence requirements. Causally later cached target observations provide confidence-weighted evidence to all eligible pending mismatches, while their states remain separate and may resolve at different positions. The segment is committed only when all pending mismatches resolve; admission failure or infeasibility within the shared verification boundary triggers rejection at the anchoring mismatch and emission of the target correction.

To formalize deferred verification, for $m \in \mathcal{M}$ and $t \in [V]$ with $t \geq m$, we say that the mismatch at m is *pending* at t if its verification decision remains unresolved after considering cached target information through position t. Concurrent pending mismatches occur at t when at least two distinct mismatches are pending simultaneously. This is distinct from merely having multiple mismatches in $d_{1:V}$: their verification decisions must overlap in time.

Because pending mismatches may originate at different draft positions, a later cached target distribution can be causally subsequent to multiple unresolved mismatches and thus provide evidence for several pending decisions at once. We therefore study verification beyond the first mismatch using only the valid candidate span $d_{1:V}$ and the cached target distributions $\{q_t\}_{t=1}^{V+1}$ from the current target-model pass, without additional target-model calls, while allowing multiple mismatch decisions to remain pending concurrently. Our goal is to improve draft reuse and decoding efficiency while preserving generation quality as closely as possible relative to strict verification.

4 Method

4.1 Overview

Concurrent verification raises three coupled questions: which mismatches are eligible for deferral and how much continuation evidence each should require, how subsequent cached target outputs should be weighted, and how overlapping pending mismatches should be resolved. PEND addresses these questions with three mechanisms. **Target-Support-Adaptive Mismatch Verification** (§4.2) uses target-relative support to determine mismatch admission and the required continuation evidence. **Confidence-Weighted Evidence Accumulation** (§4.3) weights subsequent cached target outputs by target decisiveness. **Mismatch-Specific Concurrent Resolution** (§4.4) maintains a separate evidence history and completion condition for each pending mismatch while allowing the same causally later observations to contribute to multiple pending mismatches. Figure 2 illustrates how these mechanisms interact within a single verification pass of PEND. Each continuation-verification invocation uses a fixed continuation horizon and commits a draft segment only after every pending mismatch satisfies its completion conditions. PEND uses only cached target outputs from the current verification pass and requires no additional target-model forward pass.

4.2 TARGET-SUPPORT-ADAPTIVE MISMATCH VERIFICATION

Mismatches differ in how strongly the target model supports the draft token relative to its top choice. PEND uses this support both for admission to continuation verification and for setting the required continuation evidence.

For each draft position $t \in [V]$, let $q_t^{(1)} := q_t(y_t^\star)$ and $q_t^{(d)} := q_t(d_t)$ denote the target probabilities of the top-1 token and the draft token. We define the target-relative local disagreement as

$$r_t := 1 - \frac{q_t^{(d)}}{q_t^{(1)}}, 0 \le r_t < 1. \tag{2}$$

An exact match has $r_t = 0$; larger r_t indicates weaker target-relative support for the draft token.

A mismatch at position m is admitted only if

$$r_m \le \tau_r, \tag{3}$$

where $0 < \tau_r < 1$ is the local admission threshold. Gate failure rejects the current unresolved segment as described in Section 4.4; otherwise, the mismatch remains pending.

For an admitted mismatch, define the normalized local disagreement

$$\rho_m := \frac{r_m}{\tau_r}, \tag{4}$$

and the required continuation-evidence mass

$$Z_{\mathrm{req}}(m) := Z_0 + \lambda_Z \rho_m, \tag{5}$$

where $Z_0 > 0$ is the base requirement and $\lambda_Z \ge 0$ controls its dependence on local disagreement. Thus, admitted mismatches with weaker target-relative support require more continuation evidence.

We also define the target-relative local log regret $\ell_m := \log \frac{q_m^{(1)}}{q_m^{(d)}} = -\log(1 - r_m)$. Equation (3) is equivalently expressed as $\ell_m \le -\log(1 - \tau_r)$, and $Z_{\mathrm{req}}(m)$ is nondecreasing in ℓ_m over the admissible range. Appendix B.1 gives the corresponding deterministic local-regret bounds.

4.3 CONFIDENCE-WEIGHTED EVIDENCE ACCUMULATION

Later target predictions vary in informativeness. PEND weights each cached observation by the separation between the target model's top two choices.

For each draft position $t \in [V]$, define the runner-up probability $q_t^{(2)} := \max_{v \in \mathcal{Y} \setminus \{y_t^\star\}} q_t(v)$. Using $q_t^{(1)}$ from Section 4.2, define the confidence weight

$$c_t := 1 - \frac{q_t^{(2)}}{q_t^{(1)}}, 0 \le c_t < 1. \tag{6}$$

Thus, c_t is the relative probability margin between the target model's top two choices, a monotone function of the corresponding logit gap. We use it to weight target decisiveness rather than as a calibrated correctness probability.

For an admitted mismatch at position m and any $t \in [V]$ with $t \ge m$, define

$$N_m(t) := \sum_{j=m+1}^{t} c_j r_j, \quad Z_m(t) := \sum_{j=m+1}^{t} c_j, \tag{7}$$

where an empty sum is zero, and the normalized continuation disagreement

$$D_m(t) := \frac{N_m(t)}{Z_m(t) + \epsilon}, \tag{8}$$

with numerical stabilizer $\epsilon > 0$.

Here, $Z_m(t)$ is the accumulated evidence mass and $D_m(t)$ its confidence-weighted disagreement. A zero-disagreement observation ($r_j = 0$) increases Z_m without increasing N_m; otherwise, position j contributes $c_j r_j$ to N_m. These statistics are updated only while mismatch m remains pending and, together with $Z_{\mathrm{req}}(m)$, determine completion in Section 4.4. Appendix B.2 gives the corresponding deterministic continuation-regret certificate.

4.4 MISMATCH-SPECIFIC CONCURRENT RESOLUTION

When multiple mismatches are pending, PEND maintains a separate verification state for each mismatch while sharing causally subsequent cached observations across eligible states.

Shared boundary and concurrent updates. Each continuation-verification invocation is anchored at the next mismatch after the currently accepted or committed prefix and terminates when the anchored unresolved segment is committed or rejected. Let m_{anc} denote the anchoring mismatch; for the first invocation, $m_{\mathrm{anc}} = m_0$. The shared verification boundary is $B := \min\{m_{\mathrm{anc}} + W_{\max}, V\}$, where $W_{\max} \geq 1$ bounds the number of positions examined strictly after m_{anc}. Later mismatches instantiate additional pending states but do not reset or extend B.

PEND scans $t = m_{\mathrm{anc}}, \ldots, B$ in order. At each position, the cached observation summarized by (c_t, r_t) first updates every pending mismatch $m < t$ through Equation (7). If t is itself a mismatch, PEND then applies the admission rule in Equation (3): gate failure rejects the segment anchored at m_{anc}, whereas admission initializes a new pending state with $N_t(t) = Z_t(t) = 0$ and evidence requirement $Z_{\mathrm{req}}(t)$. Completion is tested only after this admission step, so position t may support earlier pending mismatches but cannot serve as continuation evidence for the state created at t itself.

Despite sharing observations, the states remain mismatch-specific. For each mismatch m, $Z_{\mathrm{req}}(m)$ is fixed at admission from its local disagreement, while $N_m(t)$ and $Z_m(t)$ accumulate only causally subsequent observations while m remains pending. Appendix B.3 formalizes this separability.

Per-mismatch completion and joint commitment. A pending mismatch m completes after processing position t when

$$Z_m(t) \geq Z_{\mathrm{req}}(m) \quad \text{and} \quad D_m(t) \leq \tau_D, \tag{9}$$

where $0 \leq \tau_D \leq 1$ is the continuation-disagreement threshold.

Segment-level commitment is joint. Let $\mathcal{A}_t$ denote the set of mismatches still pending after position t is processed, including any mismatch admitted at t. PEND commits the current segment when

$$\mathcal{A}_t = \varnothing. \tag{10}$$

When this condition holds, PEND commits $d_{m_{\mathrm{anc}}:t}$ and resumes from $t+1$ using the remaining cached target outputs. Once all V draft positions have been accepted or committed, PEND additionally emits $y^\star_{V+1}$ as the target-model bonus token.

The update order is essential: if position t both completes an earlier mismatch and is itself a mismatch, testing joint commitment before admitting t would commit through the new mismatch even though it has accumulated no post-mismatch evidence.

Bounded verification and early rejection. The shared boundary B limits the evidence available to each pending mismatch. After processing position t, let $R_t := B - t$ denote the number of remaining positions. Under an optimistic continuation in which every remaining position contributes the maximum evidence mass and no additional disagreement, define

$$Z_m^{\mathrm{best}}(t) := Z_m(t) + R_t, \quad D_m^{\mathrm{best}}(t) := \frac{N_m(t)}{Z_m(t) + R_t + \epsilon}. \tag{11}$$

These quantities respectively upper-bound the evidence mass that can be accumulated and lower-bound the normalized disagreement that can be attained by the boundary. If

$$Z_m^{\mathrm{best}}(t) < Z_{\mathrm{req}}(m) \quad \text{or} \quad D_m^{\mathrm{best}}(t) > \tau_D, \tag{12}$$

then mismatch m cannot complete by B, so PEND rejects the segment anchored at m_{anc}.

Appendix B.4 shows that Equation (12) can only advance a rejection that is already unavoidable under the fixed boundary. Otherwise, verification continues until all pending mismatches complete or B is reached; any state still pending at B also triggers rejection. On rejection, PEND emits $y^\star_{m_{\mathrm{anc}}}$ and discards $d_{m_{\mathrm{anc}}:V}$.

Appendix A provides the complete procedure in Algorithm 1. Across commit-and-resume invocations, a cached position already scanned by one invocation is not revisited by a later continuation scan, and at most $O(W_{\max})$ mismatch states coexist. The verifier therefore requires $O(VW_{\max})$ control work and $O(W_{\max})$ auxiliary state, with no additional target-model forward pass. Appendix B.5 gives the formal argument.

Table 1: Main results across two model pairs and four benchmarks. Values are means over three runs; throughput is reported as mean ± sample standard deviation.

Model	Method	HumanEval+			GSM8K			HotpotQA			MT-Bench		
		Pass@1	MAT	Throughput	EM	MAT	Throughput	F1	MAT	Throughput	JS	MAT	Throughput
Qwen	vanilla SD	87.80	9.16	24.76 ±0.54	95.75	9.80	26.50 ±0.23	79.79	6.92	18.83 ±0.23	7.53	6.04	16.10 ±0.20
	SelfJudge	75.00	15.39	41.46 ±0.42	91.81	14.87	39.52 ±0.78	74.43	14.02	37.62 ±0.86	6.85	15.05	39.62 ±0.75
	MARS	82.93	12.87	35.85 ±0.31	94.84	13.24	40.02 ±0.33	79.91	10.06	31.78 ±0.41	7.11	9.83	26.23 ±0.33
	FLy	86.59	12.05	34.06 ±0.60	94.54	13.03	39.12 ±0.86	79.23	9.34	28.37 ±0.69	7.37	8.39	22.30 ±0.42
	LinguaSpec	84.76	11.69	33.08 ±0.33	95.00	12.14	37.31 ±0.47	80.03	8.49	27.28 ±0.71	7.51	7.68	20.23 ±0.79
	PEND	**86.59**	**13.85**	**40.18 ±0.31**	**95.53**	**13.23**	**40.70 ±0.89**	**80.05**	**9.62**	**30.55 ±0.37**	**7.49**	**9.66**	**26.19 ±0.60**
Llama	vanilla SD	73.78	10.78	27.35 ±0.39	95.38	9.87	25.56 ±0.45	80.15	7.62	19.46 ±0.18	7.04	7.65	18.75 ±0.19
	SelfJudge	66.26	15.26	38.08 ±0.56	87.92	15.25	39.64 ±0.07	67.44	15.39	38.36 ±0.12	6.15	15.11	34.97 ±0.14
	MARS	74.39	13.74	38.38 ±0.72	95.07	13.17	39.00 ±0.61	79.13	11.01	34.75 ±0.34	6.84	11.49	30.21 ±0.32
	FLy	75.61	13.38	38.10 ±0.02	94.95	12.56	37.33 ±1.66	77.80	10.57	31.41 ±0.22	6.42	10.57	27.98 ±0.11
	LinguaSpec	73.78	12.86	35.71 ±0.32	95.15	12.20	36.28 ±0.18	79.81	9.50	30.27 ±0.02	6.95	9.46	24.51 ±0.27
	PEND	**76.83**	**14.14**	**41.12 ±0.22**	**95.22**	**13.10**	**38.19 ±0.47**	**80.00**	**10.45**	**33.15 ±1.23**	**6.94**	**10.68**	**27.49 ±0.11**

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Models and Benchmarks. We evaluate PEND using two draft-target model pairs: Qwen2.5-7B-Instruct → Qwen2.5-32B-Instruct (Yang et al., 2024) and Llama-3.1-8B-Instruct → Llama-3.1-70B-Instruct (Grattafiori et al., 2024). We evaluate on four benchmarks spanning code generation, mathematical reasoning, multi-hop question answering, and open-ended instruction following: HumanEval+ (Chen et al., 2021; Liu et al., 2023), GSM8K (Cobbe et al., 2021), HotpotQA (Yang et al., 2018), and MT-Bench (Zheng et al., 2023).

Baselines and Evaluation Metrics. We compare PEND against vanilla SD (Leviathan et al., 2023), SelfJudge (Yoon et al., 2026), MARS (Song et al., 2026), FLy (Li et al., 2026), and LinguaSpec (Wang et al., 2026a). These baselines cover strict verification (vanilla SD) and three complementary relaxed-verification paradigms: learned verification (SelfJudge), training-free local-evidence verification (MARS), and continuation-evidence verification (FLy and LinguaSpec). For task quality, we report Pass@1 on HumanEval+, exact match (EM) on GSM8K, answer F1 on HotpotQA (all in %), and judge score (JS) on MT-Bench (1-10). For efficiency, we report throughput (tokens/s), time per output token (TPOT; ms/token), speedup (×) relative to target-only decoding, and MAT (tokens/round), defined as the mean number of emitted tokens per draft round, including the target-model bonus token when present.

Implementation Details. All experiments use greedy decoding, BF16 inference, and a batch size of 1. In the main comparison, all speculative methods use the same proposal length of $K = 15$. Shared decoding settings are standardized across methods, while verifier-specific baseline rules and parameters follow the corresponding papers or released implementations when available. Qwen experiments use two NVIDIA A800-SXM4-80GB GPUs, whereas Llama experiments use two NVIDIA H20-96GB GPUs; all methods within each model-benchmark setting use the same hardware configuration. All main-comparison metrics are averaged over three independent runs; throughput additionally reports the sample standard deviation to characterize runtime variability. Additional implementation details are provided in Appendix C.1.

5.2 PERFORMANCE COMPARISON

Comparison with Strict Verification. Across all eight model-benchmark settings, PEND increases throughput over vanilla SD by 46.6-70.3%, with an average gain of 57.2% (Table 1). The largest improvement occurs on Llama/HotpotQA, where throughput increases from 19.46 to 33.15 tokens/s. At the same time, task quality remains close to strict verification across benchmarks: PEND achieves higher scores on Qwen/HotpotQA and Llama/HumanEval+, while showing only small variations in the remaining settings. PEND also increases MAT across all settings, confirming that the end-to-end speedups are accompanied by more effective reuse of draft tokens beyond the first mismatch. Over-

all, these results demonstrate that PEND consistently translates additional post-mismatch draft reuse into substantial decoding acceleration while largely preserving the task quality of strict verification.

Comparison with Relaxed Verifiers. Among the relaxed-verification baselines, FLy and LinguaSpec are the closest methodological comparators because both exploit post-mismatch continuation evidence. PEND achieves higher throughput than LinguaSpec in all eight settings and than FLy in seven of eight, while matching or exceeding FLy in task quality. This consistent advantage indicates that explicitly resolving overlapping pending mismatches improves the use of continuation evidence beyond what existing deferred-verification strategies provide. Among complementary verification paradigms, SelfJudge reaches a more aggressive reuse regime, with higher MAT, but consistently lower task quality than PEND. MARS provides a training-free local-evidence comparison, for which PEND achieves higher task quality across all eight settings while retaining competitive efficiency. These results show that PEND improves the quality-efficiency balance of continuation-based verification while remaining competitive across broader relaxed-verification paradigms.

Quality-Throughput Trade-off. We further characterize the quality-throughput trade-off over five operating points per method on four benchmarks (Figure 3; Appendix C.2). At matched or closely comparable task quality, PEND consistently achieves higher throughput than FLy, LinguaSpec, and MARS. On HumanEval+, PEND gains 13.6% and 17.3% over FLy and LinguaSpec, respectively, at 86.59% Pass@1; on GSM8K, the gains are 4.7% over FLy at 94.47% EM and 14.6% over LinguaSpec at 95.30% EM. Within 0.5 F1 on HotpotQA, the throughput gains over FLy, LinguaSpec, and MARS are 3.7%, 17.5%, and 7.3%, respectively; within 0.1 judge score on MT-Bench, they are 19.5%, 38.1%, and 13.2%. SelfJudge achieves higher throughput, but at substantially lower task quality across all four benchmarks, so matched-quality comparisons focus on the other three baselines. Overall, PEND's efficiency advantage persists across operating points rather than arising from a single verifier configuration.

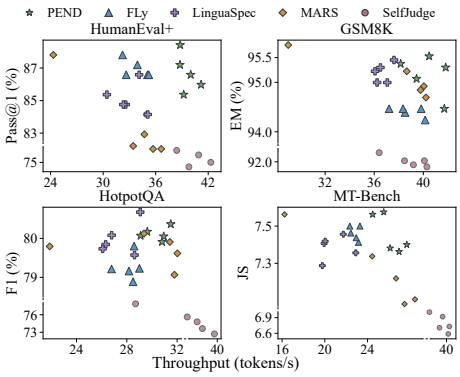


Figure 3: Quality-throughput trade-offs on Qwen2.5-7B→32B across five operating points per method and four benchmarks.

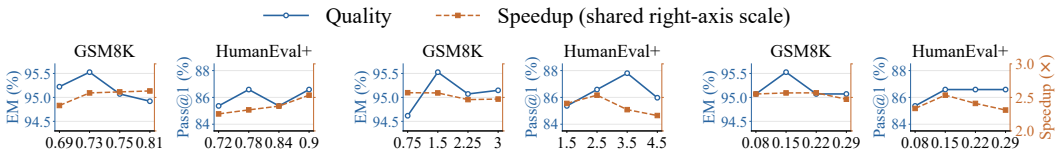
5.3 Verifier Parameter Analysis

We focus on the three verifier parameters controlling mismatch admission, evidence requirements, and continuation-based completion. Figure 4 summarizes the quality-speedup trends, with detailed quality, MAT, and TPOT results in Table 6. Appendix C.4 further extends the sensitivity analysis to proposal length K, verification window $W_{\max}$, and base evidence requirement Z_0.

Local admission threshold τ_r. Increasing τ_r admits mismatches with weaker target-relative support. Across the tested values, MAT consistently increases and TPOT decreases on both tasks, while task quality varies non-monotonically; GSM8K reaches its highest EM at $\tau_r = 0.73$ (Figure 4(a)). Thus, τ_r primarily controls admission aggressiveness: more permissive admission improves draft reuse and efficiency, with task-dependent effects on quality.

Disagreement scaling λ_Z. Increasing λ_Z imposes a larger continuation-evidence requirement on mismatches with weaker target-relative support. Quality and efficiency vary non-monotonically across the tested values (Figure 4(b)), as changing λ_Z alters which pending mismatches accumulate sufficient evidence within the continuation window. Thus, λ_Z acts as a mismatch-specific evidence-calibration parameter rather than a simple global strictness control.

Continuation-disagreement threshold τ_D. Increasing τ_D relaxes the completion criterion by allowing larger normalized continuation disagreement. Moving from $\tau_D = 0.08$ to 0.15 improves quality and reduces TPOT on both benchmarks; at $\tau_D = 0.15$, GSM8K reaches its highest EM and HumanEval+ its lowest TPOT (Figure 4(c)). Further relaxation provides little additional quality benefit, indicating that a moderate threshold offers a favorable balance between efficient completion and continuation consistency.



(a) Admission threshold τ_r (b) Disagreement scaling λ_Z (c) Continuation-disagreement threshold τ_D

Figure 4: Sensitivity of task quality and decoding speedup to the three verifier parameters on Qwen2.5-7B→32B across GSM8K and HumanEval+.

Table 2: Core design ablations on Qwen2.5-7B→32B across GSM8K and HumanEval+.

Variant	GSM8K			HumanEval+		
	EM	MAT	TPOT	Pass@1	MAT	TPOT
w/o Local Admission	93.48	15.09	21.83	81.71	14.43	24.19
Local Admission Only	94.77	13.75	24.20	82.93	13.96	26.35
Constant Evidence Requirement	94.62	13.49	24.04	84.76	13.62	26.64
Uniform Evidence Weights	94.69	13.43	25.17	84.15	13.40	27.03
First-Mismatch-Only	94.92	13.45	23.94	84.76	13.39	27.24
PEND (Full)	**95.53**	**13.24**	**23.97**	**86.59**	**13.85**	**24.67**

5.4 Ablation Study

Target-Support-Adaptive Mismatch Verification. Removing local admission increases MAT and reduces TPOT on both tasks, but substantially degrades task quality (Table 2), showing that unconstrained admission trades verification reliability for more aggressive draft reuse. Local Admission Only also lowers quality, indicating that local target support is useful for deciding whether a mismatch may be deferred but is insufficient for determining its final acceptance. Replacing the adaptive requirement $Z_{\text{req}}(m)$ with the constant Z_0 further reduces quality. These ablations validate the two roles of target-relative support: admission gating and mismatch-specific evidence calibration.

Confidence-Weighted Evidence Accumulation. Uniform Evidence Weights removes the confidence weighting in Equation (6). Despite similar MAT to full PEND, it lowers task quality and increases TPOT on both benchmarks. This indicates that the benefit comes not only from accumulating continuation evidence, but also from weighting each observation by target decisiveness.

Mismatch-Specific Concurrent Resolution. First-Mismatch-Only tracks only the earliest pending mismatch, so later mismatches do not maintain independent evidence histories or completion conditions. It yields lower task quality on both tasks at similar MAT, with comparable or higher TPOT than full PEND. Because the quality gap emerges at similar MAT, it cannot be explained by greater draft reuse; instead, it highlights the benefit of maintaining mismatch-specific verification states for overlapping decisions while sharing causally subsequent evidence. Appendix C.3 further decomposes the completion criteria, cross-mismatch evidence sharing, and stopping rules.

6 Conclusion

Deferring mismatch decisions allows later mismatches to arise before earlier ones resolve, turning speculative verification into a concurrent decision problem. PEND addresses this setting by maintaining separate verification states for pending mismatches while sharing subsequent cached target outputs as evidence. Across all eight evaluated settings, PEND improves throughput over vanilla SD while maintaining task quality close to strict verification and achieving better quality-efficiency trade-offs than prior continuation-based verifiers. The First-Mismatch-Only ablation yields lower task quality despite similar MAT, supporting the need to resolve overlapping mismatches independently. Effective deferred verification, therefore, depends not only on using post-mismatch evidence but also on how concurrent mismatch decisions are represented and resolved.

AI USE STATEMENT

During the preparation of this manuscript, an LLM was used solely for language editing, including grammatical correction and improvements to wording, clarity, and readability.

REPRODUCIBILITY STATEMENT

We provide the methodological details, algorithmic procedures, implementation choices, experimental configurations, and evaluation protocols needed to reproduce our results. Appendix C.1 further documents the evaluation protocol and the final PEND configuration for each benchmark. For main comparisons, repeated measurements were performed after final configuration selection, with throughput variability reported using the sample standard deviation. We will release the accompanying code and experimental configurations to facilitate independent reproduction of our results.

REFERENCES

Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Edgar Schoenfeld, Ali Thabet, and Jonas Kohler. Judge decoding: Faster speculative sampling requires going beyond model alignment. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/656d6174fd5ffb01c843f269649ab5cb-Abstract-Conference.html.

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

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

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

Yichao Fu, Rui Ge, Zelei Shao, Zhijie Deng, and Hao Zhang. Scaling speculative decoding with lookahead reasoning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 191092–191124, 2025. doi: 10.52202/085713-5738. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/fb65f4aed3027871f349dbc91cd27ae4-Abstract-Conference.html.

Roman Garipov, Fedor Velikonivtsev, Ivan Ermakov, Ruslan Svirschevski, Vage Egiazarian, and Max Ryabinin. AutoJudge: Judge decoding without manual annotation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 104904–104941, 2025. doi: 10.52202/085713-3163. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/87e498bbb5a41b3d2e456218b4ceb939-Abstract-Conference.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 Rozier, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, et al. The Llama 3 herd of models. arXiv preprint arXiv:2407.21783, 2024. URL https://arxiv.org/abs/2407.21783.

Maximilian Holsman, Yukun Huang, and Bhuwan Dhingra. Fuzzy speculative decoding for a tunable accuracy-runtime tradeoff. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 26257–26273. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.1346. URL https://aclanthology.org/2025.findings-acl.1346/.

Zhengmian Hu, Tong Zheng, Vignesh Viswanathan, Ziyi Chen, Ryan Rossi, Yihan Wu, Dinesh Manocha, and Heng Huang. Towards optimal multi-draft speculative decoding. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/0907335ecf28faf15be54485dbcbe70e-Abstract-Conference.html.

Ashish Khisti, MohammadReza Ebrahimi, Hassan Dbouk, Arash Behboodi, Roland Memisevic, and Christos Louizos. Multi-draft speculative sampling: Canonical decomposition and theoretical limits. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/04cdf500730af7733a6b13cbbc230206-Abstract-Conference.html.

Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael W. Mahoney, Amir Gholami, and Kurt Keutzer. Speculative decoding with big little decoder. In *Advances in Neural Information Processing Systems*, volume 36, pp. 39236–39256, 2023. doi: 10.52202/075280-1705. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/7b97adeafa1c51cf65263459ca9d0d7c-Abstract-Conference.html.

Hiroaki Kingetsu, Kaoru Yokoo, Kenji Fukumizu, and Manohar Kaul. Look before you leap: A lookahead reasoning quality gate for speculative decoding. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7831–7847. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.eacl-long.367. URL https://aclanthology.org/2026.eacl-long.367/.

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.

Jinze Li, Yixing Xu, Guanchen Li, Shuo Yang, Jinfeng Xu, Xuanwu Yin, Dong Li, Edith Ngai, and Emad Barsoum. Training-free loosely speculative decoding: Accepting semantically correct drafts beyond exact match. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/acb94e709f02895fd98b5867f0b184f3-Abstract-Conference.html.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 21558–21572, 2023. doi: 10.52202/075280-0943. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/43e9d647ccd3e4b7b5baab53f0368686-Abstract-Conference.html.

Jingwei Song, Xinyu Wang, Hanbin Wang, Xiaoxuan Lei, Tianyu Shi, Shixin Han, Eric Yang, Xiao-Wen Chang, and Lynn Ai. MARS: Unleashing the power of speculative decoding via margin-aware verification. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 8652–8663. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.421. URL https://aclanthology.org/2026.findings-acl.421/.

Ryan Sun, Tianyi Zhou, Xun Chen, and Lichao Sun. SpecHub: Provable acceleration to multi-draft speculative decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 20620–20641. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.1148. URL https://aclanthology.org/2024.emnlp-main.1148/.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. SpecTr: Fast speculative decoding via optimal transport. In *Advances in Neural Information Processing Systems*, volume 36, pp. 30222–30242, 2023. doi: 10.52202/075280-1314. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/6034a661584af6c28fd97a6f23e56c0a-Abstract-Conference.html.

Ziteng Sun, Uri Mendlovic, Yaniv Leviathan, Asaf Aharoni, Jae H Ro, Ahmad Beirami, and Ananda Theertha Suresh. Block verification accelerates speculative decoding. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/3e710b42b1a9ed898f607ec0f4fcc971-Abstract-Conference.html.

Oszkár Urbán, Young D. Kwon, Stylianos I. Venieris, and Cecilia Mascolo. Margins, not windows: Training-free per-step lossy speculative decoding. arXiv preprint arXiv:2609.02897, 2026. URL https://arxiv.org/abs/2609.02897.

Jikai Wang, Zhenxu Tian, Juntao Li, Qingrong Xia, Xinyu Duan, Zhefeng Wang, Baoxing Huai, and Min Zhang. Alignment-augmented speculative decoding with alignment sampling and conditional verification. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6751–6763. Association for Computational Linguistics, 2025a. doi: 10.18653/v1/2025.emnlp-main.343. URL https://aclanthology.org/2025.emnlp-main.343/.

Jingyi Wang, Jiaqi Huang, Zunnan Xu, Jun Zhou, Kehong Yuan, and Xiang Qian. Training-free adaptive speculative decoding via linguistic priors. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 21184–21194. Association for Computational Linguistics, 2026a. doi: 10.18653/v1/2026.findings-acl.1065. URL https://aclanthology.org/2026.findings-acl.1065/.

Yixuan Wang, Yijun Liu, Shiyu Ji, Yuzhuang Xu, Yang Xu, Qingfu Zhu, and Wanxiang Che. Think before you accept: Semantic reflective verification for faster speculative decoding. arXiv preprint arXiv:2505.18629, 2025b. URL https://arxiv.org/abs/2505.18629.

Ziyi Wang, Siva Rajesh Kasa, Ankith M S, Santhosh Kumar Kasa, Jiaru Zou, Sumit Negi, Ruqi Zhang, Nan Jiang, and Qifan Song. DIVERSED: Relaxed speculative decoding via dynamic ensemble verification. In *Proceedings of The 29th International Conference on Artificial Intelligence and Statistics*, volume 300 of *Proceedings of Machine Learning Research*, pp. 2341–2349. PMLR, 2026b. URL https://proceedings.mlr.press/v300/wang26d.html.

Yepeng Weng, Qiao Hu, Xujie Chen, Li Liu, Dianwen Mei, Huishi Qiu, Jiang Tian, and Zhongchao Shi. Traversal verification for speculative tree decoding. In *Advances in Neural Information Processing Systems*, volume 38, pp. 86562–86587, 2025. doi: 10.52202/085713-2611. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/70596d70542c51c8d9b4e423f4bf2736-Abstract-Conference.html.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. Speculative decoding: Exploiting speculative execution for accelerating Seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3909–3925. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.257. URL https://aclanthology.org/2023.findings-emnlp.257/.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Kanghoon Yoon, Minsub Kim, Sungjae Lee, Joonhyung Lee, Sunghyeon Woo, Yeonjun In, Se Jung Kwon, Chanyoung Park, and Dongsoo Lee. SelfJudge: Faster speculative decoding via self-supervised judge verification. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026. URL https://openreview.net/forum?id=9DAX4oLVhy.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623, 2023. doi: 10.52202/075280-2020. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

A PEND VERIFICATION ALGORITHM

Algorithm 1 gives the complete PEND verification procedure using the cached target outputs from a single target-model verification pass. Each continuation-verification invocation is anchored at the first mismatch at or after the current resume position. The first invocation is anchored at $m_{\text{anc}} = m_0$; after a segment is committed, verification may resume from a later anchor. At each scanned position, the current observation first updates all causally earlier pending mismatches and is then used to admit a new mismatch, if present. If all V draft positions are eventually accepted or committed, PEND additionally emits $y^\star_{V+1}$ as the target-model bonus token. For brevity, Algorithm 1 suppresses the scan-position argument and uses N_m, Z_m, and A for the current values of $N_m(t)$, $Z_m(t)$, and A_t, respectively.

Algorithm 1 PEND Verification

Require: Verification span $d_{1:V}$, cached target distributions $\{q_t\}_{t=1}^{V+1}$, parameters τ_r, τ_D, Z_0, λ_Z, and $W_{\max}$

Set resume index $s \leftarrow 1$.
while $s \leq V$ **do**
  Let m_{anc} be the first position in $\{s, \dots, V\}$ such that $d_{m_{\text{anc}}} \neq y^\star_{m_{\text{anc}}}$.
  if no such position exists **then**
    Accept the remaining span $d_{s:V}$, emit $y^\star_{V+1}$ as a bonus token, and return.
  end if
  if $m_{\text{anc}} > s$ **then**
    Accept the exact-match prefix $d_{s:m_{\text{anc}}-1}$.
  end if
  Set $B \leftarrow \min\{m_{\text{anc}} + W_{\max}, V\}$ and $\mathcal{A} \leftarrow \varnothing$, where $\mathcal{A}$ stores the pending mismatch positions.
  for $t = m_{\text{anc}}, \dots, B$ **do**
    Compute local disagreement r_t using Equation (2).
    if $t > m_{\text{anc}}$ **then**
      Compute c_t using Equation (6); for every $m \in \mathcal{A}$ with $m < t$, set $N_m \leftarrow N_m + c_t r_t$ and $Z_m \leftarrow Z_m + c_t$.
    end if
    if $d_t \neq y^\star_t$ **then**
      if $r_t > \tau_r$ **then** reject at m_{anc} (emit $y^\star_{m_{\text{anc}}}$ and discard $d_{m_{\text{anc}}:V}$) and return.
      Set $\rho_t \leftarrow r_t/\tau_r$ and $Z_{\text{req}}(t) \leftarrow Z_0 + \lambda_Z \rho_t$.
      Initialize $N_t \leftarrow 0$ and $Z_t \leftarrow 0$, and add t to $\mathcal{A}$.
    end if
    Remove from $\mathcal{A}$ every mismatch m whose current accumulators satisfy Equation (9) at position t.
    if $\mathcal{A} = \varnothing$ **then**
      Commit $d_{m_{\text{anc}}:t}$ and set $s \leftarrow t + 1$.
      break.
    end if
    if $t = B$ **or** any $m \in \mathcal{A}$ satisfies Equation (12) **then**
      Reject at m_{anc} (emit $y^\star_{m_{\text{anc}}}$ and discard $d_{m_{\text{anc}}:V}$) and return.
    end if
  end for
end while
Emit $y^\star_{V+1}$ as a bonus token and return.

B THEORETICAL GUARANTEES FOR PEND

This appendix formalizes verifier-level properties of the three mechanisms introduced in Section 4: *Target-Support-Adaptive Mismatch Verification*, *Confidence-Weighted Evidence Accumulation*, and *Mismatch-Specific Concurrent Resolution*. Specifically, we establish target-relative local-regret control, a confidence-weighted continuation certificate, separability of mismatch-specific accumulators

under shared evidence, and per-mismatch completion under joint commitment. We further show that early rejection is sound and that verifier-side bookkeeping remains bounded without additional target-model forward passes.

We use the notation from Sections 3 and 4. In particular, for each draft position $t \in [V]$, $q_t^{(1)} := q_t(y_t^\star)$ and $q_t^{(d)} := q_t(d_t)$. All verifier statistics analyzed below depend only on the cached, draft-prefix-conditioned distributions $\{q_t\}_{t=1}^{V}$ produced by the single target-model verification pass; q_{V+1} is used only to produce the target-model bonus token when the full draft span is accepted or committed. None of the results assumes that these cached distributions are recomputed after correcting an earlier mismatch.

We assume

$$0 < \tau_r < 1, \quad 0 \le \tau_D \le 1, \quad Z_0 > 0, \quad \lambda_Z \ge 0, \quad W_{\max} \ge 1, \tag{13}$$

together with the numerical stabilizer $\epsilon > 0$ used in the normalized continuation disagreement. We analyze the ideal softmax probabilities induced by finite logits, so all probabilities appearing in logarithmic quantities are strictly positive. For readability, we omit the fixed floating-point comparison tolerance used in the implementation; the same tolerance is applied consistently to completion and rejection comparisons. Unless stated otherwise, the results concern the full verifier in Algorithm 1.

For later use, define the target-relative local log regret

$$\ell_t := \log \frac{q_t^{(1)}}{q_t^{(d)}} = -\log(1 - r_t), \tag{14}$$

and $L_r := -\log(1 - \tau_r)$, $\kappa_r := \frac{L_r}{\tau_r}$. An exact match satisfies $r_t = \ell_t = 0$. Under a top-probability tie, the converse need not hold: deterministic tie-breaking may classify d_t as a mismatch even when $q_t^{(d)} = q_t^{(1)}$, in which case $r_t = \ell_t = 0$. The results below remain valid in this case.

B.1 Target-Support-Adaptive Mismatch Verification: Local Regret Control

The local admission gate bounds the target-relative local log regret of every mismatch admitted to continuation verification. Moreover, the normalized local disagreement ρ_m upper-bounds the normalized log regret, giving the adaptive evidence requirement a deterministic interpretation.

Proposition 1 (Regret-controlled admission and evidence requirement)**.** *Let m be a mismatch admitted by the local admission gate. Then*

$$0 \le \ell_m \le L_r, \tag{15}$$

and, since an admitted mismatch satisfies $0 \le r_m \le \tau_r$,

$$\frac{\ell_m}{L_r} \le \rho_m = \frac{r_m}{\tau_r} \le 1. \tag{16}$$

Consequently,

$$Z_{\mathrm{req}}(m) = Z_0 + \lambda_Z \rho_m \ge Z_0 + \frac{\lambda_Z}{L_r} \ell_m. \tag{17}$$

Equivalently, for every admitted mismatch,

$$Z_{\mathrm{req}}(m) = Z_0 + \frac{\lambda_Z}{\tau_r} \left(1 - e^{-\ell_m}\right), \quad 0 \le \ell_m \le L_r. \tag{18}$$

Hence the continuation-evidence requirement is nondecreasing in the mismatch's target-relative local regret, and strictly increasing when $\lambda_Z > 0$.

Proof. From the definition of local disagreement, $1 - r_m = \frac{q_m^{(d)}}{q_m^{(1)}}$, so Equation (14) gives $\ell_m = -\log(1 - r_m)$. Because $f(r) = -\log(1 - r)$ is increasing on $[0, 1)$, $r_m \le \tau_r \implies \ell_m \le -\log(1 - \tau_r) = L_r$, which proves Equation (15).

To prove Equation (16), note that f is convex and $f(0) = 0$. For $0 \le r_m \le \tau_r$,

$$f(r_m) = f\left(\frac{r_m}{\tau_r}\tau_r + \left(1 - \frac{r_m}{\tau_r}\right) 0\right) \le \frac{r_m}{\tau_r} f(\tau_r). \tag{19}$$

Therefore

$$\frac{\ell_m}{L_r} = \frac{f(r_m)}{f(\tau_r)} \le \frac{r_m}{\tau_r} = \rho_m. \tag{20}$$

Since $\lambda_Z \ge 0$, substituting this inequality into $Z_{\text{req}}(m) = Z_0 + \lambda_Z \rho_m$ yields Equation (17). Finally, $r_m = 1 - e^{-\ell_m}$ gives Equation (18) directly. □

Lemma 1 (Cumulative target-relative local log regret)**.** *Let C be a segment committed by the verifier, and let $\mathcal{M}_C$ denote the mismatch positions contained in C. With each term evaluated under the corresponding target distribution q_m from Equation (1),*

$$R_{\text{loc}}(C) := \sum_{m \in \mathcal{M}_C} \log \frac{q_m^{(1)}}{q_m^{(d)}} \le |\mathcal{M}_C| L_r = |\mathcal{M}_C| \log \frac{1}{1-\tau_r}. \tag{21}$$

Proof. Every mismatch $m \in \mathcal{M}_C$ contained in a committed segment must have passed the local admission gate; otherwise, the unresolved segment would have been rejected rather than committed. Proposition 1 therefore gives $\ell_m \le L_r$ for every $m \in \mathcal{M}_C$. Summing over mismatch positions proves Equation (21). □

Equation (21) is a local bound under the cached, draft-prefix-conditioned target distributions $q_m(\cdot) = p_T(\cdot \mid x, d_{<m})$ of the current verification pass, rather than a sequence-level comparison with a counterfactual target-only trajectory.

B.2 Confidence-Weighted Evidence Accumulation: Continuation Certificates

Recall from Section 4.3 that

$$q_t^{(2)} := \max_{v \in \mathcal{Y} \setminus \{y_t^\star\}} q_t(v), \quad c_t = 1 - \frac{q_t^{(2)}}{q_t^{(1)}}. \tag{22}$$

We first characterize c_t as a relative target margin, then relate the completion statistic D_m to a confidence-weighted continuation log regret.

Lemma 2 (Relative target-margin interpretation)**.** *For every draft position $t \in [V]$,*

$$c_t = \frac{q_t^{(1)} - q_t^{(2)}}{q_t^{(1)}}. \tag{23}$$

If $z_t^{(1)}$ and $z_t^{(2)}$ denote the logits associated with $q_t^{(1)}$ and $q_t^{(2)}$, respectively, then

$$c_t = 1 - \exp\left(-\left[z_t^{(1)} - z_t^{(2)}\right]\right). \tag{24}$$

Hence c_t is a strictly increasing transformation of the top-two logit gap. If position t is a mismatch, then $c_t \le r_t$.

Proof. Equation (23) is immediate from the definition of c_t. Softmax probability ratios satisfy $\frac{q_t^{(2)}}{q_t^{(1)}} = \exp\left(z_t^{(2)} - z_t^{(1)}\right)$, which proves Equation (24).

If t is a mismatch, then $d_t \ne y_t^\star$, so by definition of $q_t^{(2)}$, $q_t^{(d)} \le q_t^{(2)}$. Hence $\frac{q_t^{(d)}}{q_t^{(1)}} \le \frac{q_t^{(2)}}{q_t^{(1)}}$, and therefore $r_t = 1 - \frac{q_t^{(d)}}{q_t^{(1)}} \ge 1 - \frac{q_t^{(2)}}{q_t^{(1)}} = c_t$. □

For an admitted mismatch m, define its confidence-weighted continuation log regret after processing position $t > m$ by

$$\overline{L}_m(t) := \frac{\sum_{j=m+1}^{t} c_j \ell_j}{Z_m(t) + \epsilon}. \tag{25}$$

Proposition 2 (Target-support-adaptive continuation certificate)**.** *Consider a mismatch m contained in a segment eventually committed by PEND, and let h_m denote the position at which mismatch m completes. Then*

$$Z_m(h_m) \geq Z_0 + \frac{\lambda_Z}{L_r}\ell_m, \tag{26}$$

and

$$\overline{L}_m(h_m) \leq \kappa_r D_m(h_m) \leq \kappa_r \tau_D. \tag{27}$$

Thus every committed mismatch satisfies both a target-support-adaptive evidence requirement and an explicit bound on its confidence-weighted continuation log regret.

Proof. At completion, $Z_m(h_m) \geq Z_{\mathrm{req}}(m)$, so Proposition 1 yields Equation (26).

Now consider any $j \in \{m+1, \ldots, h_m\}$. If j is an exact match, then $r_j = \ell_j = 0$. If j is a mismatch, then it must have passed the local admission gate; otherwise the unresolved segment containing m would have been rejected and could not later be committed. Hence, for every such j, $0 \leq r_j \leq \tau_r$. Using the same convexity argument as in Proposition 1,

$$\ell_j = -\log(1 - r_j) \leq \frac{-\log(1-\tau_r)}{\tau_r} r_j = \kappa_r r_j. \tag{28}$$

Multiplying by $c_j \geq 0$ and summing gives

$$\sum_{j=m+1}^{h_m} c_j \ell_j \leq \kappa_r \sum_{j=m+1}^{h_m} c_j r_j = \kappa_r N_m(h_m). \tag{29}$$

Dividing by $Z_m(h_m) + \epsilon > 0$ yields $\overline{L}_m(h_m) \leq \kappa_r D_m(h_m)$. The completion rule in Equation (9) gives $D_m(h_m) \leq \tau_D$, proving Equation (27). □

Since $N_m(h_m) \leq Z_m(h_m)$ and $Z_m(h_m) \geq Z_0$ at completion, the numerical stabilizer changes the corresponding unstabilized disagreement statistic $N_m(h_m)/Z_m(h_m)$ by at most ϵ/Z_0.

B.3 Mismatch-Specific Concurrent Resolution: Accumulator Separability and Joint Commitment

PEND allows a later observation to update several earlier pending mismatches while keeping each mismatch's evidence history and completion condition separate. We first show that sharing evidence does not numerically couple the mismatch-specific accumulators, and then show that joint commitment certifies every mismatch contained in a committed segment.

For a mismatch m, consider the hypothetical standalone accumulator

$$\widetilde{N}_m(t) := \sum_{j=m+1}^{t} c_j r_j, \quad \widetilde{Z}_m(t) := \sum_{j=m+1}^{t} c_j, \tag{30}$$

initialized at m and evaluated on the same cached target observations as PEND.

Proposition 3 (Mismatch-specific accumulator separability)**.** *Fix an admitted mismatch m. At every position $t > m$ at which an evidence update is applied while mismatch m remains pending and before the current invocation terminates,*

$$N_m(t) = \widetilde{N}_m(t), \quad Z_m(t) = \widetilde{Z}_m(t). \tag{31}$$

Consequently, whenever Algorithm 1 reaches the completion test for mismatch m at position t, its mismatch-specific accumulators satisfy Equation (9) if and only if the standalone accumulator satisfies the same conditions.

The accumulators and completion test for mismatch m therefore depend only on its own position, local disagreement, and causally subsequent cached observations; they do not depend on the accumulated values of any other pending mismatch.

Proof. When mismatch m is admitted, $N_m(m) = Z_m(m) = 0$, which agrees with Equation (30) before any eligible observation has been incorporated.

Assume Equation (31) holds after the preceding eligible position. At position $t > m$, Algorithm 1 updates the accumulators of every earlier mismatch that remains pending by $N_m \leftarrow N_m + c_t r_t$, $Z_m \leftarrow Z_m + c_t$. The standalone accumulator receives exactly the same increment, so Equation (31) continues to hold by induction.

If t is itself a mismatch, it is admitted only after the observation at t has updated earlier pending mismatches. It therefore begins with $N_t(t) = Z_t(t) = 0$ and does not consume its own mismatch-position observation. Finally, the completion test for mismatch m uses only its own $N_m(t)$, $Z_m(t)$, $Z_{\mathrm{req}}(m)$, and τ_D, none of which contains another mismatch's accumulated statistics. This proves the stated separability property. □

Proposition 3 formalizes the distinction between *shared evidence* and *separate decisions*. A later target observation may enter several mismatch-specific accumulators, but it does not merge their evidence histories, transfer accumulated evidence between mismatches, or change another mismatch's completion requirement. Coupling occurs only through invocation-level stopping rules: the shared boundary, joint commitment, and segment-level rejection.

Proposition 4 (Causal per-mismatch completion)**.** *For every mismatch m contained in a committed segment, its mismatch-specific accumulators use evidence only from positions strictly after m and, upon completion, individually satisfy both conditions in Equation (9).*

Proof. The first claim follows from Proposition 3: an admitted mismatch m starts with zero evidence and receives only contributions indexed by $j > m$. Now consider a committed segment $d_{m_{\mathrm{anc}}:t}$. Every mismatch contained in this segment must have passed its local admission gate and been added to the pending set; a failed gate would reject the segment anchored at m_{anc}. The commit rule in Equation (10) is triggered only when the pending set is empty. Because completion is the only non-rejection operation that removes a mismatch from this set, every mismatch in $d_{m_{\mathrm{anc}}:t}$ must have completed individually and therefore satisfies Equation (9).

As a boundary consequence, a mismatch created at $m = B$ has no eligible position $j > m$ within the current invocation. Its evidence mass remains zero, while $Z_{\mathrm{req}}(B) = Z_0 + \lambda_Z \rho_B \geq Z_0 > 0$. Such a mismatch cannot complete within the invocation and cannot use evidence from positions at or before its own mismatch position. □

Corollary 1 (Per-mismatch deterministic completion certificate)**.** *For every mismatch m contained in a committed segment, there exists a completion position $h_m > m$ such that*

$$\sum_{j=m+1}^{h_m} c_j \geq Z_0 + \lambda_Z \frac{r_m}{\tau_r} \geq Z_0 + \frac{\lambda_Z}{L_r} \ell_m, \tag{32}$$

$$\sum_{j=m+1}^{h_m} c_j r_j \leq \tau_D \left(\sum_{j=m+1}^{h_m} c_j + \epsilon \right), \tag{33}$$

and

$$\overline{L}_m(h_m) \leq \kappa_r \tau_D. \tag{34}$$

Thus every committed mismatch has its own causal evidence history, target-support-adaptive evidence requirement, and completion certificate, even when some later observations are simultaneously reused by other pending mismatches.

Proof. For an admitted mismatch, $0 \leq \frac{r_m}{\tau_r} \leq 1$, so Equation (4) gives $Z_{\mathrm{req}}(m) = Z_0 + \lambda_Z \frac{r_m}{\tau_r}$. The first inequality in Equation (32) is the evidence-mass condition in Equation (9); the second follows from Proposition 1. Multiplying the continuation-disagreement condition by the strictly positive denominator $Z_m(h_m) + \epsilon$ gives Equation (33). Equation (34) follows from Proposition 2. □

The need for mismatch-specific completion is structural rather than merely implementational.

Proposition 5 (Earliest-mismatch completion is insufficient for joint commitment)**.** *There exist valid verifier statistics for which the earliest pending mismatch completes at a position t that is itself a newly admitted mismatch with zero post-mismatch evidence. Therefore, a verifier that commits solely because the earliest pending mismatch has completed cannot, in general, guarantee a completion certificate for every committed mismatch.*

Proof. Consider $\tau_r = 0.8$, $\tau_D = 0.7$, $Z_0 = 0.4$, $\lambda_Z = 0$, with any $\epsilon > 0$. Let the current invocation be anchored at an admitted mismatch $m_{\mathrm{anc}} = 1$. At position $t = 2$, suppose the target distribution satisfies $q_2^{(1)} = 0.50$, $q_2^{(2)} = 0.25$, $q_2^{(d)} = 0.20$, with the remaining probability assigned to other tokens. Position 2 is a mismatch and $r_2 = 1 - \frac{0.20}{0.50} = 0.60 \le \tau_r$, $c_2 = 1 - \frac{0.25}{0.50} = 0.50$. Before mismatch 2 is added to the pending set, position 2 updates the accumulators of the earlier anchoring mismatch m_{anc}. Hence $Z_{m_{\mathrm{anc}}}(2) = 0.50 \ge Z_0$, and $D_{m_{\mathrm{anc}}}(2) = \frac{0.50 \times 0.60}{0.50+\epsilon} < 0.60 < \tau_D$. Thus the anchoring mismatch can complete at position 2.

However, because position 2 is itself a mismatch, PEND then admits it with $N_2(2) = Z_2(2) = 0$. Since $Z_{\mathrm{req}}(2) = Z_0 = 0.4 > 0$, the new mismatch cannot complete at its admission position. A verifier that committed through position 2 solely because the anchoring mismatch m_{anc} had completed would therefore commit mismatch 2 without any post-mismatch completion certificate. PEND avoids this failure by adding mismatch 2 to the pending set before evaluating the global condition $\mathcal{A}_t = \varnothing$. □

The proposition is structural: it establishes the failure of per-mismatch certification, not necessarily a task-level error. It also explains the update order in Algorithm 1: at position t, PEND first applies the observation to causally earlier pending mismatches, then admits t if it is itself a mismatch, and only afterward tests whether the pending set is empty.

Joint commitment also makes commitment atomic at the unresolved-segment level: within an invocation anchored at m_{anc}, rejection never requires rolling back a token already committed from that unresolved segment.

B.4 Soundness of Early Rejection

PEND can reject an unresolved segment before reaching the shared boundary when even an optimistic continuation cannot satisfy a pending mismatch's completion conditions. The following result shows that this pruning rule is sound and becomes monotonically tighter as the shared boundary approaches.

Proposition 6 (Monotone feasibility envelope and sound early rejection)**.** *Let mismatch m remain pending after the completion test at position t. For every later position $u \in \{t+1, \dots, B\}$ reached before termination, any continuation of mismatch m satisfies*

$$Z_m(u) \le Z_m^{\mathrm{best}}(t), \tag{35}$$

and

$$D_m(u) \ge D_m^{\mathrm{best}}(t). \tag{36}$$

Therefore, if

$$Z_m^{\mathrm{best}}(t) < Z_{\mathrm{req}}(m) \quad \textit{or} \quad D_m^{\mathrm{best}}(t) > \tau_D, \tag{37}$$

then mismatch m cannot satisfy Equation (9) at any position up to the shared boundary.

Moreover, while mismatch m remains pending, $Z_m^{\mathrm{best}}(t)$ is non-increasing in t, whereas $D_m^{\mathrm{best}}(t)$ is non-decreasing. Relative to an otherwise identical verifier that disables only early rejection, Equation (12) therefore cannot change the commit/reject outcome of the segment anchored at m_{anc} and can only advance an inevitable rejection.

Proof. Fix a mismatch m that remains pending after the completion test at position t, and define the future increments for any $u \in \{t+1, \dots, B\}$ as

$$\Delta Z_{t:u} := \sum_{j=t+1}^{u} c_j, \quad \Delta N_{t:u} := \sum_{j=t+1}^{u} c_j r_j. \tag{38}$$

Since $0 \le c_j \le 1$ and $r_j \ge 0$,

$$0 \le \Delta Z_{t:u} \le u - t \le R_t, \quad \Delta N_{t:u} \ge 0. \tag{39}$$

Hence

$$Z_m(u) = Z_m(t) + \Delta Z_{t:u} \le Z_m(t) + R_t = Z_m^{\text{best}}(t), \tag{40}$$

which proves Equation (35).

For normalized continuation disagreement,

$$D_m(u) = \frac{N_m(t) + \Delta N_{t:u}}{Z_m(t) + \Delta Z_{t:u} + \epsilon} \ge \frac{N_m(t)}{Z_m(t) + \Delta Z_{t:u} + \epsilon} \ge \frac{N_m(t)}{Z_m(t) + R_t + \epsilon} = D_m^{\text{best}}(t), \tag{41}$$

proving Equation (36). Therefore either branch of Equation (12) certifies that mismatch m cannot satisfy its completion conditions before B.

For monotonicity, note that $R_{t+1} = R_t - 1$. If mismatch m remains pending after the completion test at position $t+1$, then

$$Z_m^{\text{best}}(t+1) = Z_m(t) + c_{t+1} + R_t - 1 = Z_m^{\text{best}}(t) + c_{t+1} - 1 \le Z_m^{\text{best}}(t). \tag{42}$$

For the disagreement envelope, the numerator changes from $N_m(t)$ to $N_m(t) + c_{t+1} r_{t+1}$ and is therefore non-decreasing, while the denominator changes by $c_{t+1} - 1 \le 0$ and is therefore non-increasing. Since the denominator remains strictly positive because of ϵ, $D_m^{\text{best}}(t)$ is non-decreasing.

Finally, compare two verifiers that are identical except that one disables early rejection. Their mismatch-specific accumulators agree up to the first early-rejection trigger. At that point, the preceding argument shows that at least one pending mismatch cannot complete by B. Because commitment requires the pending set to become empty, the verifier without early rejection cannot commit the segment anchored at m_{anc} either. It must eventually reject because of a later gate failure or because the boundary is reached with an unresolved mismatch. Early rejection therefore preserves the final segment decision and changes only the stopping time. □

B.5 Whole-Pass Control Complexity and Target-Forward Count

For an invocation anchored at m_{anc}, $B = \min\{m_{\text{anc}} + W_{\max}, V\}$, so at most $B - m_{\text{anc}} + 1 \le W_{\max} + 1$ positions are processed within that invocation, including the anchor. The mismatch-specific bookkeeping is therefore confined to a fixed-width region of the cached verification span.

Proposition 7 (Whole-pass verifier control complexity and forward count)**.** *Given the target distributions produced by one verification pass, all PEND invocations induced by commit-and-resume within a valid verification span of length V require $O(VW_{\max})$ verifier-side control work in the worst case and $O(W_{\max})$ auxiliary storage. For a single continuation-verification invocation, the mismatch-specific bookkeeping requires $O(W_{\max}^2)$ control work. No additional target-model forward pass is required.*

Proof. Within one invocation, all pending mismatches lie inside a window containing at most $W_{\max} + 1$ inspected positions. Therefore the number of simultaneously stored constant-size mismatch records is $O(W_{\max})$.

At any processed continuation position, the current observation can update all earlier pending mismatches, while the completion and early-rejection checks inspect at most the same number of states. The verifier-side control work per such position is therefore $O(W_{\max})$.

When an invocation commits at position t, Algorithm 1 resumes from position $t+1$. Hence no continuation position already processed by a completed invocation is revisited by a later invocation. Across all invocations, at most V cached positions participate in continuation scans, giving $O(VW_{\max})$ accumulator-update and completion-check work.

Locating each next anchoring mismatch m_{anc} also scans forward from the current resume index. Because the resume index is monotone and accepted exact-match positions are not revisited, these searches contribute $O(V)$ additional comparisons over the full verification span. Since $W_{\max} \ge 1$, this does not change the overall $O(VW_{\max})$ control bound.

The number of simultaneously pending mismatches is at most $O(W_{\max})$. For a single invocation, at most $W_{\max} + 1$ continuation positions are processed, yielding the special-case bound $O(W_{\max}^2)$. All required quantities, including the top-1 probability, top-2 probability, draft-token probability, r_t, c_t, and the mismatch-specific accumulators, are obtained from or computed from the cached target outputs produced by the existing verification pass. The verifier therefore requires no additional target-model forward pass. □

When $W_{\max}$ is treated as a fixed inference-time constant, verifier-side control cost is linear in the cached verification-span length V. These bounds concern verifier-side control operations after the relevant target probabilities are available; end-to-end latency additionally depends on draft generation, target verification, probability extraction, memory traffic, and the realized emitted length per draft round.

B.6 Scope of the Guarantees

The results above characterize PEND under the cached, draft-prefix-conditioned target distributions from the current verification pass. They establish target-relative local-regret control, a confidence-weighted continuation certificate, mismatch-specific accumulator separability and per-mismatch completion under joint commitment, sound early rejection, and bounded verifier-side control without additional target-model forward passes.

These are verifier-level guarantees for the relaxed-verification setting. They do not imply exact preservation of the target model's output distribution, nor do they compare against a counterfactual target-only trajectory after an earlier mismatch has been corrected. Accordingly, task-level generation quality is evaluated empirically in Section 5 and Appendix C.

C Additional Experiments

C.1 Experimental Settings

Evaluation Protocol. We use the full evaluation sets of HumanEval+, GSM8K, and MT-Bench. For HotpotQA, all methods are evaluated on the same fixed subset of 1,024 examples from the distractor validation split, selected deterministically by proportional stratification over question type and answer form. All methods use greedy decoding and identical task prompts within each model-benchmark setting. The maximum generation length is 1,024 tokens for HumanEval+, 512 tokens for GSM8K and HotpotQA, and 1,536 tokens per turn for MT-Bench. For MT-Bench, we follow the FastChat single-answer judging protocol using `GPT-5.6-luna` with temperature 0. The judge prompt preserves the official rubric, reference-answer policy, multi-turn structure, and 1-10 rating scale, while adding prompt-injection safeguards and stricter output-format instructions. The same judging and aggregation procedure is applied to all methods. Unless otherwise stated, the main comparison reports averages over three runs, whereas the ablation and parameter analyses report single-run measurements.

PEND Operating-Point Configuration. For the main comparison, PEND uses a benchmark-specific operating point shared across all model-hardware settings. We fix $K = 15$, $W_{\max} = 8$, $\tau_D = 0.15$, and $Z_0 = 1.00$ globally; only τ_r and λ_Z vary by benchmark (Table 3). For each benchmark, τ_r and λ_Z are selected on a held-out development set disjoint from the evaluation examples and then frozen; the resulting benchmark-specific configuration is shared across both model-hardware settings.

Table 3: PEND operating points.

Benchmark	τ_r	λ_Z
HumanEval+	0.90	2.50
GSM8K	0.73	1.50
HotpotQA	0.72	2.00
MT-Bench	0.75	1.50

Baseline Configurations. We standardize all shared decoding and evaluation settings across methods, including the draft-target model pairs, proposal length K, generation limits, numerical precision, batch size, hardware, and evaluation protocol. Verifier-specific rules and parameters follow the

Table 4: Prespecified operating points for the quality-throughput comparison. Following the original baseline evaluation protocols, each baseline uses a task-shared threshold grid; PEND uses task-specific (τ_r, λ_Z) grids centered on the development-selected operating points.

Method	Task	Parameter	Operating points				
			1	2	3	4	5
SelfJudge	Task-shared	θ_{SJ}	0.10	0.30	0.50	0.70	0.90
MARS	Task-shared	θ_{MARS}	0.80	0.85	0.90	0.95	1.00
FLy	Task-shared	θ_{FLy}	0.10	0.20	0.30	0.40	0.50
LinguaSpec	Task-shared	θ_{LS}	0.10	0.20	0.30	0.40	0.50
PEND	HumanEval+	(τ_r, λ_Z)	(0.85, 2.00)	(0.85, 3.00)	(0.95, 2.00)	(0.95, 3.00)	(0.90, 2.50)
	GSM8K		(0.68, 1.00)	(0.68, 2.00)	(0.78, 1.00)	(0.78, 2.00)	(0.73, 1.50)
	HotpotQA		(0.67, 1.75)	(0.67, 2.25)	(0.77, 1.75)	(0.77, 2.25)	(0.72, 2.00)
	MT-Bench		(0.70, 1.00)	(0.70, 2.00)	(0.80, 1.00)	(0.80, 2.00)	(0.75, 1.50)

Table 5: Additional verifier decompositions on Qwen2.5-7B→32B.

Variant	GSM8K			HumanEval+		
	EM	MAT	TPOT	Pass@1	MAT	TPOT
w/o Evidence-Mass Requirement	94.84	13.63	24.95	82.32	13.95	26.20
w/o Continuation-Disagreement Criterion	95.07	13.36	25.10	86.59	13.16	27.95
w/o Cross-Mismatch Evidence Sharing	95.30	13.23	24.84	83.54	12.76	27.43
w/o Commit-and-Resume & Early Rejection	95.15	12.85	26.36	86.59	12.30	28.95
Commit-and-Resume Only	95.07	13.40	24.20	86.59	13.14	28.24
Early Rejection Only	95.15	12.85	25.16	86.59	12.30	29.59
PEND (Full)	**95.53**	**13.24**	**23.97**	**86.59**	**13.85**	**24.67**

corresponding papers or released implementations when available. We use the official FLy implementation with its reported verifier configuration. For MARS, we apply the published margin-aware verification rule and reported threshold within our common draft-model evaluation pipeline. For SelfJudge, we train a model-pair-specific verifier following the paper and select its threshold on the validation split. For LinguaSpec, we implement the published part-of-speech adaptive verification rule with the reported category-specific thresholds and verification windows. For the main comparison, we additionally report the sample standard deviation of throughput over three runs.

C.2 Quality-Throughput Trade-off

We compare PEND with SelfJudge, FLy, LinguaSpec, and MARS on HumanEval+, GSM8K, HotpotQA, and MT-Bench, using five prespecified operating points per method and task. PEND jointly varies (τ_r, λ_Z) using the four corners and center of a task-specific parameter grid, while fixing $\tau_D = 0.15$, $Z_0 = 1.00$, and $W_{\max} = 8$. SelfJudge varies its acceptance threshold θ_{SJ}. FLy varies its entropy threshold θ_{FLy}, LinguaSpec its content-word probability threshold θ_{LS}, and MARS its logit-ratio threshold θ_{MARS}, with all remaining verifier and drafting settings fixed as specified in Appendix C.1. Following the task-shared parameterization used in the original baseline evaluations, we use the same threshold grid across tasks for each baseline, whereas PEND uses task-specific grids centered on the development-selected operating points described in Appendix C.1. Table 4 reports the prespecified operating points.

C.3 Additional Mechanism Decompositions

Table 5 extends the core ablations in Table 2 by isolating three design components of mismatch-specific concurrent resolution: the two completion conditions, causal cross-mismatch evidence sharing, and the invocation-level stopping rules.

Completion Conditions. Unlike the Constant Evidence Requirement ablation in Table 2, removing the evidence-mass requirement here eliminates the evidence-mass completion condition itself. This

Table 6: Verifier parameter measurements corresponding to the main-text sensitivity analyses on Qwen2.5-7B→32B across GSM8K and HumanEval+.

Parameter	GSM8K				HumanEval+			
	Value	EM	MAT	TPOT	Value	Pass@1	MAT	TPOT
τ_r	0.69	95.22	12.97	25.86	0.72	85.37	12.78	27.73
	0.73	95.53	13.24	23.97	0.78	86.59	13.14	27.01
	0.75	95.07	13.40	23.83	0.84	85.37	13.53	26.37
	0.81	94.92	13.70	23.70	0.90	86.59	13.85	24.67
λ_Z	0.75	94.62	13.46	23.94	1.50	85.37	13.39	25.93
	1.50	95.53	13.24	23.97	2.50	86.59	13.85	24.67
	2.25	95.07	13.31	24.94	3.50	87.80	12.88	26.97
	3.00	95.15	13.16	24.86	4.50	85.98	12.46	28.03
τ_D	0.08	95.07	13.33	24.12	0.08	85.37	13.06	26.76
	0.15	95.53	13.24	23.97	0.15	86.59	13.85	24.67
	0.22	95.07	13.41	23.96	0.22	86.59	13.16	25.93
	0.29	95.07	13.41	24.88	0.29	86.59	13.16	27.04

lowers quality on both tasks, most sharply on HumanEval+, and increases TPOT despite comparable or slightly higher MAT. The degradation at similar draft reuse indicates the importance of accumulating sufficient evidence mass before a pending mismatch can resolve. Removing the continuation-disagreement condition together with its paired disagreement-based early-rejection check also increases TPOT; GSM8K EM decreases, whereas HumanEval+ Pass@1 remains unchanged. The two completion conditions therefore play complementary roles: evidence mass controls the quantity of accumulated continuation evidence, while continuation disagreement controls its consistency. Because the latter ablation also removes its paired feasibility check, its efficiency effect reflects the disagreement branch as a whole.

Causal Cross-Mismatch Evidence Sharing. Removing cross-mismatch evidence sharing lowers quality and increases TPOT on both benchmarks, with a larger quality degradation on HumanEval+. Notably, GSM8K MAT remains essentially unchanged relative to full PEND while both quality and TPOT worsen. The degradation therefore cannot be explained simply by a change in draft reuse. Instead, it highlights the value of reusing the same causally eligible cached observation across multiple pending mismatches while keeping their accumulators and completion conditions mismatch-specific.

Invocation-Level Stopping Rules. Disabling both commit-and-resume and early rejection lowers MAT and increases TPOT on both benchmarks without a consistent improvement in task quality. Retaining commit-and-resume alone recovers nearly all of the GSM8K efficiency but remains less efficient than full PEND on HumanEval+, whereas retaining early rejection alone is less efficient than full PEND on both tasks. The two rules play complementary roles: commit-and-resume ends an invocation once all pending mismatches have completed and resumes verification from the remaining cached target outputs, whereas early rejection stops processing once completion becomes infeasible within the shared verification boundary.

C.4 Additional Verifier Parameter Analysis

Table 6 reports the full quality, MAT, and TPOT measurements for the τ_r, λ_Z, and τ_D sweeps shown in Figure 4. Table 7 complements these main-text analyses with proposal length K, verification window $W_{\max}$, and base evidence requirement Z_0.

Proposal Length K. Increasing K enlarges the candidate span and substantially increases MAT, but also raises draft-generation cost. $K = 15$ gives the lowest observed TPOT on both tasks; increasing to $K = 20$ yields further MAT gains without improving end-to-end efficiency. These results indicate diminishing efficiency returns as the proposal length increases within the tested range.

Verification Window $W_{\max}$. Increasing $W_{\max}$ from 2 to 8 substantially reduces TPOT and increases MAT on both tasks by providing more subsequent positions for evidence accumulation.

Table 7: Additional parameter analysis on Qwen2.5-7B→32B across GSM8K and HumanEval+.

Parameter	GSM8K				HumanEval+			
	Value	EM	MAT	TPOT	Value	Pass@1	MAT	TPOT
K	8	95.22	8.02	27.24	8	85.98	7.84	30.08
	12	95.38	11.15	24.80	12	85.37	11.03	27.15
	15	95.53	13.24	23.97	15	86.59	13.85	24.67
	20	95.15	16.76	24.07	20	85.98	16.55	27.58
$W_{\max}$	2	95.00	11.29	30.74	2	85.98	10.17	36.78
	4	95.07	13.08	25.28	4	83.54	12.15	29.60
	8	95.53	13.24	23.97	8	86.59	13.85	24.67
	12	95.07	13.41	23.89	12	85.98	13.27	26.91
Z_0	0.50	94.54	13.48	24.31	0.50	85.37	13.43	26.39
	1.00	95.53	13.24	23.97	1.00	86.59	13.85	24.67
	1.50	94.92	13.28	24.31	1.50	88.41	12.86	27.37
	2.00	95.15	13.09	24.48	2.00	87.20	12.58	27.67

Extending the window to 12 yields little additional benefit on GSM8K and increases HumanEval+ TPOT, indicating that the efficiency gains largely saturate by $W_{\max} = 8$.

Base Evidence Requirement Z_0. Z_0 sets the common minimum evidence mass required by every admitted mismatch. $Z_0 = 1.00$ gives the lowest TPOT on both tasks, whereas larger values can improve HumanEval+ quality at the cost of efficiency. Unlike λ_Z, which controls how the evidence requirement varies with normalized local disagreement, Z_0 determines the common evidence floor shared by all admitted mismatches.

D Case Study

Figure 5 presents a HotpotQA example whose final answer, *Newcastle upon Tyne, England*, matches the reference answer. The figure highlights selected token-level draft-target mismatches that are ultimately retained in the generated response. In the underlying verifier trace, later observations provide shared evidence to multiple causally eligible pending mismatches, each of which satisfies its own completion conditions before commitment. The figure shows token-level draft-target differences rather than the full verifier state; mismatch-specific evidence histories are omitted for clarity.

Input Prompt (context excerpted):

```
Answer the multi-hop question using only the provided context. Reason step by step and identify the evidence
that connects the relevant facts. End with exactly one line in the form 'Final answer: <short answer>'.
Context excerpts (original numbering):
[4] 2016 Brit Awards: The 2016 Brit Awards were held on 24 February 2016 and was the 36th edition of the
British Phonographic Industry's annual pop music awards. The awards ceremony was held at The O2 in London and
were presented by Ant & Dec.
[5] Ant & Dec: Anthony McPartlin, OBE (born 18 November 1975) and Declan Donnelly, OBE (born 25 September
1975), known collectively as Ant & Dec, are an English comedy TV presenting, television producing, acting and
former music duo from Newcastle upon Tyne, England.
Question: Where were the presenters of the 2016 Brit Awards from?
```

Output:

To answer the question, we need to identify who (the) the presenters of the 2016 Brit Awards were and then determine their origin.
Step 1: Identify the presenters of the 2016 Brit Awards. From [4], we learn that the 2016 Brit Awards were (ceremony) presented by Ant & Dec.
Step 2: Determine (Identify) the origin of Ant & Dec. From [5], we find that Ant & Dec are (,) an English comedy TV presenting duo from Newcastle upon Tyne, England.
Final answer: Newcastle upon Tyne, England.

Figure 5: A HotpotQA continuation-verification example in input-output format. Blue marks selected draft tokens that are ultimately committed; green parentheses indicate the target model's local top-1 predictions at the corresponding positions conditioned on the preceding draft prefix.