PREPAIR: SPECULATIVE DRAFTER TRAINING FOR GENERATIVE UI UNDER SEMANTIC ACCEPTANCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative UI lets large language models answer with interactive interfaces instead of plain text, but an interface spans many more tokens than a text reply, limiting real-time interaction. Speculative decoding accelerates generation by letting a small drafter propose tokens that a large target verifies in parallel, keeping each only if it matches the target's token distribution. In Generative UI, however, we find that the drafter writes many components that are semantically correct but not token-identical to the target's, and some correct ones that the target gets wrong, so per-token acceptance both discards valid drafts and bounds the whole system by the target. In this paper, we lift speculative decoding from per-token acceptance to component-level semantic acceptance, in which the target checks all components against several criteria in one forward pass. Existing drafter training, built for per-token acceptance, imitates the target, which under semantic acceptance both rewrites components the drafter already writes correctly and passes on the target's errors. To overcome this, we propose PREPAIR, which reinforces accepted components and distills the target only on the divergent tokens of failed ones, located by chain repair, so the two objectives never share a token. With a 1.7B drafter and a 32B target, PREPAIR raises semantic acceptance over supervised fine-tuning from 81.0% to 87.8%, for a 3.5× end-to-end speedup over the target alone versus 2.3× for EAGLE-3, and on Macaron-A2UI the trained system outscores the 32B target writing directly under an independent evaluator.

1 INTRODUCTION

Generative UI lets a large language model answer with an interactive interface instead of plain text (Leviathan et al., 2026): the model writes a declarative document, such as A2UI (Google, 2025; Kong et al., 2026), that the client renders as cards, forms and buttons. The structured form that makes an interface renderable also makes it long: every component carries its type, properties, layout and data bindings, so an interface runs to several hundred tokens, far more than a text reply, which limits real-time interaction. Work on Generative UI has so far focused on interface quality rather than generation cost (Leviathan et al., 2026; Kong et al., 2026).

Speculative decoding is a common way to cut this latency (Leviathan et al., 2023; Chen et al., 2023): a small drafter proposes tokens that the target verifies in parallel, keeping each only while it matches the target's token distribution. Under this per-token acceptance the best drafter imitates the target most closely, and drafter training has accordingly been a matter of imitation, or model alignment in the words of Bachmann et al. (2025): distilling the target's distribution (Zhou et al., 2024), predicting its next token from its features (Li et al., 2024; 2025), aligning training and inference contexts (Zhang et al., 2025), or maximizing the acceptance of whole draft paths (Zou et al., 2026).

Generative UI exposes two limits of per-token acceptance. First, it rejects valid work: a document is a sequence of self-contained components, and 59.1% of the components an untrained 1.7B drafter writes are judged correct by the 32B target, yet only 39.5% of those are token-identical to the target's own (Appendix C), so most are cut at their first differing token. Second, being lossless, it outputs exactly what the target would have written, so even where the drafter writes a component correctly and the target does not, the target's version is kept, and the target bounds the whole system.

We therefore lift speculative decoding from per-token acceptance to component-level semantic acceptance (Figure 1). The target judges every component against several criteria, such as relevance,

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.

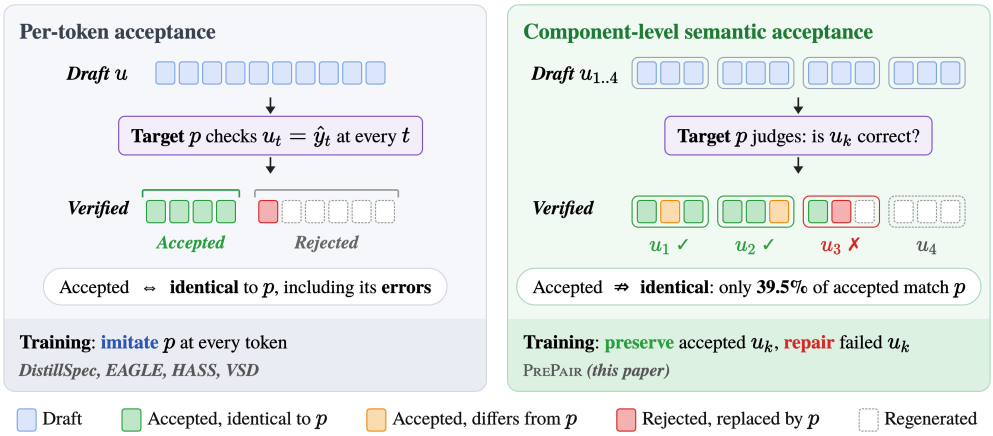


Figure 1: Why the training objective changes. Per-token acceptance cuts the draft at the first differing token; semantic acceptance keeps a whole component even where it differs, and repairs the first failed one from its first divergence, redrafting what follows. The 39.5% is for the untrained drafter over all 300 requests (Appendix C).

substance and non-repetition, and accepts it when it is correct rather than when it matches token by token. Semantic verification has been proposed for general-purpose generation (Bachmann et al., 2025; Yao et al., 2026), where output does not divide into self-contained units and is therefore verified token by token or span by span. Generative UI instead divides into components, each with explicit criteria, so no judge has to be trained; we verify a whole draft in one forward pass under a tree-structured attention mask that gives every component several isolated probes. Semantic acceptance gives up token-level losslessness, but at the component level it can improve on the target: a component the target would write wrongly is rejected, and a correct one worded differently is kept.

Under semantic acceptance, every rejection is costly: a rejected component is usually repaired by letting the target continue writing from it, which brings back the decoding that speculation exists to avoid, and left untrained, Qwen3-1.7B (Yang et al., 2025) has 40.9% of its components rejected. The drafter therefore has to be trained, but not with the token-level objectives built for per-token acceptance, which imitate the target indiscriminately: they rewrite correct components just because their wording differs from the target's, and copy the target's mistakes. Instead, it should preserve the components that are accepted and learn from the target only where a component fails. Standard post-training objectives each serve one side: a reward on the verdict (Shao et al., 2024; Rafailov et al., 2023) preserves accepted components but cannot say how a failed one should be written, while distillation from the target (Hinton et al., 2015; Liu et al., 2024; Lei et al., 2026; Bhansali & Heck, 2025) repairs failed components but also rewrites accepted ones toward the target. Simply summed, they pull the tokens where an accepted component differs from the target in opposite directions, the reward toward the drafter's wording and distillation toward the target's.

Our key observation is that the verification forward already provides the two signals needed to assign supervision per position: a semantic verdict for every component, which marks what to preserve, and, for every failed component, the first token at which the draft departs from the target together with the target's distribution there, which marks where repair starts. We propose PrePair (*Preserve-and-Repair*), which assigns supervision by these two signals: accepted components receive a verdict-weighted policy objective that preserves them, the divergent tokens of failed components receive distillation toward the target, and all other tokens receive no gradient. One forward, however, exposes only the first divergence of a failed component, so PrePair finds the rest by *chain repair*: it replaces that token with the target's choice, lets the drafter continue, and re-verifies until the component is accepted; the target never decodes, and each round yields one supervised position.

With a 1.7B drafter and a 32B target (Section 4), PrePair raises acceptance over its SFT starting point from 81.0% to 87.8%, runs 3.5× faster than the target alone against at most 2.3× for EAGLE-3, and scores 3.74 on Macaron-A2UI against 3.31 for the target writing directly. Ablations show both losses are needed and that an unassigned sum collapses the drafter.

Our contributions are:

- We introduce a tree-structured semantic verifier that, in one forward pass, judges every component with isolated probes and exposes each failed component's first divergence from the target, and chain repair, which fully repairs failed components from these signals without target decoding.
- We show that component-level semantic acceptance replaces imitation with a two-part objective, preserving correct components and repairing failed ones, that neither distillation nor reward-based training, nor their sum, can meet.
- We formulate PREPAIR as two losses on disjoint, verdict-assigned position sets, and validate it on Generative UI, where it raises acceptance from 81.0% to 87.8% and cuts target forwards per document by 36% over SFT at higher quality, runs 3.5× faster end to end than the target alone against 2.3× for EAGLE-3 and 1.7× for the same drafter under per-token acceptance, and the trained system exceeds its 32B target writing directly.

2 RELATED WORK

Drafter training for speculative decoding. Since Leviathan et al. (2023) and Chen et al. (2023), drafters have been trained to agree with the target token by token: DistillSpec (Zhou et al., 2024) distills the target's next-token distribution with a task-specific divergence, the EAGLE family (Li et al., 2024; 2025) predicts the target's next token from its intermediate features, HASS (Zhang et al., 2025) aligns the contexts seen at training and at inference, and VSD (Zou et al., 2026) maximizes the probability that a whole draft path is accepted. AdaSPEC (Hu et al., 2025) and SKD (Xu et al., 2025) further restrict distillation to selected positions, chosen by learnability under a reference model or by the teacher's ranking of the student's token. These objectives are designed for bitwise verification, where agreement with the target is the acceptance criterion itself. Under a semantic criterion, agreement and correctness no longer coincide, and selection rules based on the model pair's confidence or learnability do not by themselves separate the positions where a draft is wrong from those where it is merely worded differently.

Semantic acceptance. A second line replaces token agreement with a judgment of correctness. Judge Decoding (Bachmann et al., 2025) trains a classifier on the target's last-layer representation from human-labeled tokens, and AutoJudge (Garipov et al., 2025) learns a similar classifier without labels; in both, the criterion resides in learned weights tied to a particular model pair. PARSE (Yao et al., 2026) instead states the criterion in the prompt and verifies cumulative prefixes of a draft, cut every 40 tokens, in one masked forward pass, judging each by a single Correct/Incorrect readout and letting the target resume past the longest valid prefix, a design suited to tasks with one reference answer. These works focus on the verifier and leave the drafter as it is; how a drafter should be trained once acceptance is semantic remains open in this line.

Post-training methods. Reward-based methods reinforce outputs that pass a check, whether through policy optimization with a verdict as reward (Schulman et al., 2017; Shao et al., 2024), rejection-sampling fine-tuning (Yuan et al., 2023), or preference optimization (Rafailov et al., 2023), and are a natural fit for consolidating components a drafter already produces correctly. A scalar reward, however, indicates that a component failed without indicating what it should have been, so improving failed components is left to resampling. SFT and knowledge distillation (Hinton et al., 2015) supply that missing target, but apply it uniformly across tokens; variants that confine the loss to positions exposed by verification (Liu et al., 2024; Qian et al., 2026; Lei et al., 2026; Bhansali & Heck, 2025) narrow the scope, with positions still chosen by token-level disagreement. Combining a reward with distillation is natural, yet the two act on overlapping positions with opposite pulls, and neither family on its own offers a way to assign supervision component by component.

3 METHOD

PREPAIR trains a drafter for component-level speculative decoding; Figure 2 gives an overview. Section 3.1 fixes the setting: how a whole draft is verified in one forward pass and what that pass returns. Section 3.2 shows that the two requirements of Section 1, preserving correct components and repairing failed ones, need only what that pass provides, and repairs a failed component when the pass reveals only its first divergence; Section 3.3 turns them into two losses on disjoint positions.

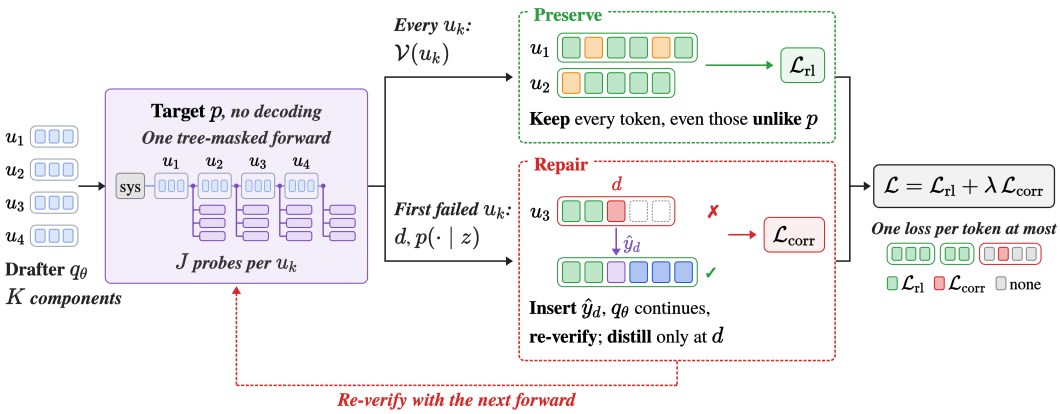


Figure 2: PREPAIR overview. The target p reads the drafter's components once under a tree mask: after each u_k come J probe branches B_k, one criterion q_j each, which see only the system prompt, $u_{<k}$ and themselves and together yield the verdict, and the trunk yields the first divergence d and $p(\cdot \mid z)$. Accepted components are kept through $\mathcal{L}_{\text{rl}}$, including tokens that differ from the target; failed ones are repaired from d through $\mathcal{L}_{\text{corr}}$.

3.1 TREE-STRUCTURED SEMANTIC VERIFICATION

Notation. Let p be the target model and q_θ the drafter. Given a request, the drafter writes a document as a sequence of components $u_1, \ldots, u_K$; s_k denotes the committed prefix before u_k, that is, the request together with every component accepted so far.

One forward pass. The target reads the whole draft once (Figure 5, Appendix A). Its input is an interleaved sequence: the trunk carries the draft tokens, and after each component u_k we append J short probe branches, one per criterion, each ending at a readout position. The criteria are stated once in the system prompt; for UI components they ask whether the component is relevant to the request, whether it carries substantive content, whether it repeats the preceding context, whether it has gone stale against the dialogue state, and whether it is static where the request calls for interaction. A tree-structured attention mask lets branch j of u_k attend to the system prompt, to the trunk up to the end of u_k, and to itself only. Branches never see one another, and the trunk never sees a branch, so a later component is judged exactly as it would be without the probes before it, and probes do not condition on each other's answers. At the readout of branch j we read the margin $m_k^{(j)} = \text{logit}(\text{Yes}) - \text{logit}(\text{No})$. The verdict $\mathcal{V}(u_k, s_k) \in \{0, 1\}$ accepts u_k when it passes a set of zero-cost hard gates (schema validity, resolvable references, no verbatim repetition), every probe margin clears a threshold calibrated on held-out accepted components, and the component's mean log-likelihood under p lies above a calibrated floor.

Divergence. Because the trunk is teacher-forced, the same forward provides $p(\cdot \mid s_k, u_{k,<t})$ at every draft position. For a failed component we take the first position at which the draft departs from the target's greedy choice,

$$d(u, s) \;=\; \min\{\, t :\; u_t \neq \hat{y}_t \;\wedge\; \Phi(t) \,\}, \qquad \hat{y}_t = \arg\max_v \, p\big(v \mid s, u_{<t}\big), \tag{1}$$

where $\Phi(t)$ excludes positions whose disagreement does not bear on the verdict: values of free-form identifier fields, whitespace-only tokens, and positions at which the two top logits lie within the numerical noise margin of bf16 inference. If no such position exists we write $d = \perp$: the component matches the target token for token. At position d we also keep the target's distribution $p(\cdot \mid z)$ over its top-K tokens, with $z = s \,\|\, u_{<d}$. The target performs no decoding at any point.

3.2 CHAIN REPAIR

Supervision component by component needs two things: which components to keep, and what a failed one should become. The forward just described supplies both, with no generation by the

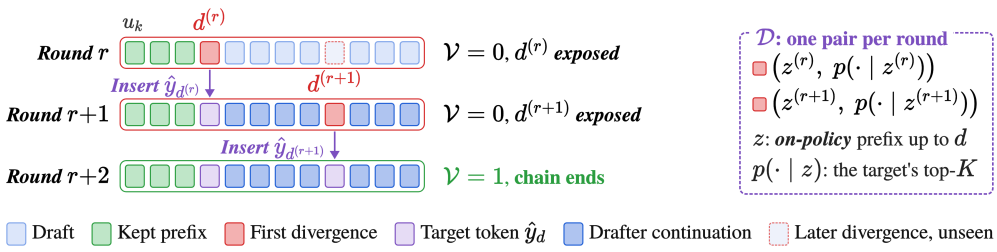


Figure 3: Chain repair. Only the divergent token comes from the target; the drafter continues and the result is re-verified, until the component is accepted. Each round yields one supervised position.

target and no second pass: the verdict partitions the draft, and for a failed component d and $p(\cdot \mid z)$ say where repair starts and what belongs there.

A single divergence rarely repairs a component. A failed u may differ from the target at several positions, and the forward exposes only the first; the rest surface only after the first is fixed and the component is rewritten. The usual remedy is to let the target rewrite everything after the divergence, as in prefix-level verification (Yao et al., 2026). This returns generation to the target and, if the rewrite is used as a training target, teaches the drafter the whole span including the parts it had right, which is the imitation problem of Section 1 again.

We repair in a chain (Figure 3). Let $u^{(0)} = u$. In round r, let $d^{(r)} = d(u^{(r)}, s)$ and let $\hat{y}_{d^{(r)}}$ be the target's choice there; the next candidate is

$$u^{(r+1)} \;=\; \Big(\underbrace{s \,\|\, u^{(r)}_{<d^{(r)}}}_{z^{(r)}} \,\|\, \hat{y}_{d^{(r)}}\Big) \;\oplus\; q_\theta, \tag{2}$$

where $\oplus\, q_\theta$ denotes continuation by the drafter until the component closes. The candidate is re-verified with the next forward. In a draft, components before u are committed, while those after it were judged against the rejected u, so they are discarded and the drafter rewrites them after the repair. The loop ends when $\mathcal{V}(u^{(r)}, s) = 1$ or $d(u^{(r)}, s) = \bot$, or after a fixed number of rounds. Throughout, the target only verifies: each round costs one forward rather than autoregressive decoding of the remainder, and the drafter learns only the divergent tokens, not a rewritten span.

The chain is also the training data. Each round r leaves a pair $\big(z^{(r)}, p(\cdot \mid z^{(r)})\big)$: a prefix that is on-policy up to $d^{(r)}$ and endorsed by the target up to there, and the target's distribution at the next token. A component with several divergences contributes one pair per round; the set of all pairs over a batch of requests is $\mathcal{D}$.

Why the loop needs a trained drafter. Both mechanisms are inference-time only, and each assumes something the drafter must supply: verification asks J criteria at once, so a component is accepted only if it is right on all of them, and chain repair assumes a component converges in a few rounds rather than drifting into a new error each time it continues. An untrained Qwen3-1.7B meets neither: 59.1% of its components are accepted, it spends 6.7 target forwards per document, and it repairs one component three or more times in 70% of documents, so it is only 1.12× faster than the target writing alone (Tables 2, 3). The drafter has to be trained for the loop it runs in.

3.3 The PrePair objective

Preserve. Accepted components are the drafter's own outputs that the target approved, so the first requirement is met by raising their probability without moving toward the target's wording. For a component slot with committed prefix s we form a group $\mathcal{G}$ of candidates, the greedy draft together with $G - 1$ samples, each verified in the same forward and rewarded by

$$R(u) \;=\; \mathbf{1}\big[\mathcal{V}(u,s) = 1\big]\Big(\tfrac{\min(|u|, L_{\text{cap}})}{\bar{L}} + \alpha\, \bar{m}(u)\Big) \;-\; \mathbf{1}\big[\mathcal{V}(u,s) = 0\big] \;-\; \mathbf{1}\big[\text{hard-gate failure}\big], \tag{3}$$

where the first term measures the target computation the component saves (its length, normalized by the mean accepted length $\bar{L}$ and capped at L_{cap}), $\bar{m}(u)$ is the mean probe margin with each probe's sign oriented so that larger is better, and a hard-gate failure is penalized twice. Within the group we

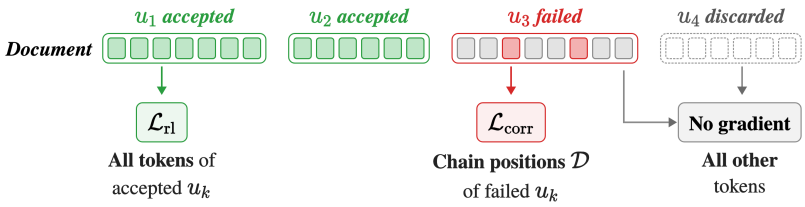


Figure 4: Supervision dispatch. Accepted components receive $\mathcal{L}_{\text{rl}}$ (Eq. 4), chain positions of failed ones receive $\mathcal{L}_{\text{corr}}$ (Eq. 5), everything else no gradient.

compute an advantage $A_i = (r_i - \mu)/(\sigma + \epsilon)$, weighting duplicate candidates by their multiplicity w_i, and update on positive advantages only:

$$\mathcal{L}_{\text{rl}}(\theta) \;=\; -\sum_i w_i \,[A_i]_+ \,\overline{\log q_\theta(u_i \mid s)}. \tag{4}$$

The overline denotes the mean token log-likelihood of u_i. Restricting the update to positive advantages keeps it on forms the drafter itself produced and the target accepted.

Repair. Chain positions receive distillation toward the target:

$$\mathcal{L}_{\text{corr}}(\theta) \;=\; \sum_{(z,\,p)\,\in\,\mathcal{D}} D_{\text{KL}}\big(p(\cdot \mid z) \,\big\|\, q_\theta(\cdot \mid z)\big), \tag{5}$$

computed over the target's top-K tokens at z after renormalization. The loss touches only positions in $\mathcal{D}$; every other token of a failed component, in particular the prefix before d that the target endorsed, receives no gradient from this term.

Objective.

$$\mathcal{L}(\theta) \;=\; \mathcal{L}_{\text{rl}}(\theta) \;+\; \lambda\,\mathcal{L}_{\text{corr}}(\theta). \tag{6}$$

The two terms act on disjoint sets of positions (Figure 4): Eq. (4) on the tokens of accepted candidates, Eq. (5) on chain positions of failed components, and every other token on nothing. Assignment is made by the verdict and the chain rather than by token disagreement, so the semantically valid but differently worded majority is left untouched, and the preservation term never opposes the correction term. Every quantity in Eqs. (3)–(6) is read from the verification forward; training adds no target-model computation beyond verification itself. Algorithm 1 in Appendix B summarizes one iteration.

4 Experiments

4.1 Setup

Models and task. The drafter is Qwen3-1.7B and the target is Qwen3-32B (Yang et al., 2025) in bf16, which also serves as the verifier; both run under vLLM (Kwon et al., 2023) on a single A100. Documents are A2UI component streams generated for three types of request: requests drawn from multi-turn dialogues (*atomic*; MultiWOZ (Budzianowski et al., 2018), SGD (Rastogi et al., 2020), ESConv (Liu et al., 2021), AnnoMI (Wu et al., 2022)), and synthesized multi-intent (*width*) and multi-turn (*depth*) requests. The target reads $J = 5$ probes per component, relevance, substance and non-redundancy plus two context-dependent checks, state continuity and interactivity, with thresholds calibrated on held-out accepted components (Appendix G); hard gates check schema validity, reference resolution and verbatim repetition at zero cost.

Training. PrePair starts from a light SFT checkpoint (SFT): Qwen3-1.7B trained for one epoch on 5,869 atomic, 870 width and 900 depth gold documents produced offline by the target and filtered by the verifier. Training then alternates collection and optimization for eight rounds. Each round collects 60 requests, 15 each of MultiWOZ and SGD atomic requests and of width and depth requests, with groups of $G = 4$ candidates at each rejected slot and at the first slot of a forward when it is accepted outright, and chain repair on every failed component; groups whose members

Table 1: Quality on the Macaron-A2UI Bench, with Avg the mean of L1–L3 and V1–V3. Direct rows write the document themselves; drafter rows are verified by Qwen3-32B.

	Model	schema err.	L1	L2	L3	V1	V2	V3	Avg
direct	GPT-5.4	**4**	4.37	**2.62**	2.29	4.72	**4.45**	**4.53**	**3.83**
	Qwen3-235B	15	4.21	2.53	**2.51**	4.66	4.35	4.49	3.79
	Qwen3-32B (target)	76	3.31	2.05	2.05	4.37	3.97	4.13	3.31
	Macaron-A2UI-Tall (30B)	81	3.60	2.16	2.11	4.31	3.72	4.02	3.32
1.7B drafter	untrained	143	2.91	1.52	1.69	4.57	3.92	3.66	3.05
	SFT	21	4.48	1.95	2.40	4.77	4.04	4.27	3.65
	ONE BATCH	30	4.39	1.97	2.35	4.80	4.09	4.29	3.65
	PREPAIR	9	**4.69**	1.95	2.38	**4.84**	4.17	4.43	3.74

Table 2: Deployment behaviour on the Macaron-A2UI Bench.

Drafter	accept	inter.	comp/doc	uiOk	EOS	fwd/doc	fwd/comp	tok/doc
untrained 1.7B	59.1%	18%	2.8	54%	65%	6.65	2.34	2648
SFT	81.0%	46%	2.7	68%	88%	4.20	1.57	1095
ONE BATCH	85.4%	47%	2.7	69%	90%	3.37	1.23	969
PREPAIR	**87.8%**	**58%**	**3.4**	**78%**	**91%**	**2.68**	**0.78**	**864**

all fail are dropped. Accepted components are also replayed in $\mathcal{L}_{\mathrm{rl}}$ with unit weight, as is the end-of-sequence token where the target accepts the draft's ending. Round 1 yields 528 group samples, 793 replayed components and 296 chain positions; the first seven rounds together contribute 8,954 training samples. Each round optimizes Eq. (6) for one epoch at learning rate 10^{-6} with the AdamW (Loshchilov & Hutter, 2019) state carried over from the previous round; chain positions are normalized per chain. We set $\lambda = 0.5$, $\alpha = 0.5$, $L_{\mathrm{cap}} = 16$ and $\bar{L} = 11.4$ tokens, $K = 64$ for $\mathcal{L}_{\mathrm{corr}}$, at most 8 repair rounds per component (24 while the divergence still advances), and sample candidates at temperature 0.9. We deploy round 7, whose checkpoint has the highest acceptance on the fresh training requests of the following round (85.4%, Figure 6), so no evaluation request is used.

Evaluation. Quality is measured on the Macaron-A2UI Bench (Kong et al., 2026), 300 held-out requests (100 atomic, 100 width, 100 depth, none seen in training; a depth request is a multi-turn episode with one document per turn, 477 documents in all): L1 (protocol correctness), L2 (task construction) and L3 (user experience) on a 1–5 scale by a GPT-5.4 judge, and visual quality V1–V3 by Qwen3-VL-235B (Bai et al., 2025) on rendered pages, averaged over documents that render an interface where one is expected; we also count schema errors, interfaces that fail the A2UI schema check, parsing or rendering. Macaron-A2UI does not release its renderer, so pages are rendered by our own implementation, and every score, baselines included, is re-measured by us; the evaluators use neither the verifier's model nor its prompts, only Macaron-A2UI's own rubric. Deployment behaviour is measured on all 300 requests for every drafter: acceptance (components accepted at first judgment, end-of-document votes excluded), share of documents with an interactive component (inter.), components per document, natural end-of-sequence rate (EOS), renderability (uiOk), target forwards per document and per component (fwd), tokens fed to the target (tok); the target never decodes a token. End-to-end latency is measured on all three request types against the target decoding autoregressively, which averages 20.5 s per document, and against EAGLE-3 (Li et al., 2025) on the same target, both with its released head and with the head retrained on the target's own unfiltered outputs for the training requests; every system reuses the shared system prompt across requests, by vLLM's prefix cache for the baselines and by pinning it in the target's cache for ours. Every system, including the target decoding alone, runs in eager mode.

Baselines. Direct generation by the target itself (Qwen3-32B), by Qwen3-235B and GPT-5.4, and by Macaron-A2UI-Tall (Kong et al., 2026), a 30B model fine-tuned for this task, all with the same system prompt; the SFT drafter without PREPAIR; PREPAIR trained as a single large batch instead of in rounds (ONE BATCH, 5,520 samples, same objective); the untrained Qwen3-1.7B placed directly in the verification loop; and, for latency, EAGLE-3 on the same target.

Table 3: End-to-end latency over the three request types of the Macaron-A2UI Bench, means per document; the target decoding alone takes 20.5 s.

Acceptance	Drafter	wall (s)	speedup			
			atomic	width	depth	all
per-token	EAGLE-3 head, released, k=4	13.0	1.77×	1.47×	1.59×	1.58×
	EAGLE-3 head, retrained, k=4	8.80	2.68×	2.11×	2.37×	2.33×
	Qwen3-1.7B, PREPAIR, k=4	11.8	1.88×	1.64×	1.77×	1.74×
semantic	Qwen3-1.7B, untrained	18.3	1.14×	0.91×	1.29×	1.12×
	Qwen3-1.7B, PREPAIR	**5.88**	**4.99**×	**2.45**×	**4.05**×	**3.49**×

4.2 MAIN RESULTS

Table 1 reports quality, Table 2 deployment behaviour and Table 3 end-to-end latency. Four observations follow.

The untrained drafter starts far from acceptable, and SFT lifts quality but still leaves heavy repair. Placed directly in the loop, Qwen3-1.7B has 59.1% of its components accepted, renders 54% of its documents and scores 3.05, below the target alone: verification filters, it does not rescue. Repairing its rejected components takes 6.65 target forwards per document, so it feeds the target 2,648 tokens, three times what PREPAIR feeds. One epoch of SFT brings acceptance to 81.0%, renderability to 68% and quality to 3.65, above every direct-generation baseline on L1, with L2 and L3 close to those of every other 1.7B drafter. Yet 19.0% of its components are still rejected and a document still takes 4.20 target forwards: imitating documents the verifier has already accepted does not teach the drafter how to repair the components it gets wrong.

PREPAIR raises acceptance, cuts target work, and writes more. Relative to SFT, PREPAIR raises acceptance from 81.0% to 87.8%, and cuts target forwards per document from 4.20 to 2.68 and tokens fed to the target from 1,095 to 864. A forward is a whole-document pass: the first judges the whole draft and each repair round costs one more, and 62% of documents need only the first. Quality moves the other way: Avg 3.65 to 3.74, with L1 up from 4.48 to 4.69, every visual dimension up, and L2 and L3 essentially unchanged. Since these scores come from evaluators independent of the verifier, the drafter has learned to write better components, not only ones the verifier accepts more easily: ONE BATCH raises acceptance to 85.4% and leaves Avg at 3.65. It also writes more: 3.4 components per document against 2.7, and 58% of documents are interactive against 46%.

The trained system exceeds its own 32B target writing directly. Qwen3-32B writing directly scores 3.31 on average with 76 schema errors; the same model verifying the 1.7B draft yields 3.74 with 9. It scores higher on every request type (4.06/3.66/3.58 against 3.87/3.56/2.74) and on every dimension but L2, and, paired by request, the verified 1.7B system produces higher-scoring outputs on more held-out requests than direct Qwen3-32B generation: 165 against 104, most clearly on depth (74 against 21; Appendix F). L2 alone accounts for more than the gap in Avg to GPT-5.4 and Qwen3-235B, 0.09 and 0.05; it leads both on L1 and V1.

End to end, the verified drafter is 3.5× faster than the target across all three request types. Over the 477 documents of the three request types (Table 3), PREPAIR averages 5.9 s against 20.5 s for the target writing alone. The gain is largest where drafts are accepted at once: a document accepted in its first verification forward takes 2.6–3.3 s for every request type, 6.9–8.3× faster than the target, while one that needs repair is only 1.8–2.2× faster, since every repair round adds a forward and a redraft. The speedups per request type therefore follow the share of documents accepted at once: 80% on atomic gives 5.0× and 66% on depth 4.1×, while on width, where a document carries 5.5 components and takes 4.2 forwards on average, only 36% are, and the speedup falls to 2.5×. The remaining time is mostly drafting: the drafter's decoding takes three quarters to four fifths of PREPAIR's wall clock and verification the rest. Per-token acceptance caps every baseline, and its speedups vary far less across request types: EAGLE-3 reaches 1.6× with its released head and 2.3× once retrained on the target's own outputs, where 39% and 70% of drafted tokens are accepted, and the same drafter under per-token acceptance 1.7×, so the criterion, not the drafter, is what scales. Left untrained, the drafter gives only 1.12×: the criterion buys little without a good drafter.

Table 4: Ablations on the Macaron-A2UI Bench; speedup is over the target alone.

Variant	accept	fwd/doc	comp/doc	Avg	atomic	width	depth	speedup
PREPAIR	**87.8%**	**2.68**	**3.44**	**3.74**	**4.06**	**3.66**	**3.58**	**3.49**×
$-\mathcal{L}_{\text{corr}}$	85.5%	4.12	3.22	3.66	4.01	3.54	3.47	2.54×
$-\mathcal{L}_{\text{rl}}$	80.8%	4.29	2.45	3.54	3.97	3.51	3.23	1.94×
plain sum	63.5%	6.59	1.37	3.41	3.82	3.21	3.24	0.22×

4.3 ANALYSIS

The verdict is reliable enough to train on. The costly error is a false accept: a false reject costs one more repair round, but a falsely accepted component is emitted as is. We therefore fix each probe's false-reject rate at 5% and measure false accepts directly (Appendix G): relevance, substance and redundancy reach AUC 0.95–0.99 and accept 2.8%, 5.6% and 20.0% of synthetic defects. The two context-dependent checks are less reliable for a 32B target, and this shows in L2, whose dimensions grade interactive components, their actions and the use of dialogue state: it stays at the SFT level (1.95) while L1 and V rise. Three safeguards limit misses: a component must clear every probe, the hard gates and the likelihood floor; each probe sits on its own branch, so no Yes carries over; and the hard gates check schema, references and repetition without the target.

Acceptance rises while agreement with the target does not. Table 5 counts accepted components token-identical to the target's greedy continuation. On the atomic requests, training raises the accepted share from 61.5% to 93.7%, and about two thirds of what is accepted still differs from the target token for token. Chasing agreement would spend most of the budget rewriting accepted components. Over all 300 requests it is 87.8% (93.7/82.1/88.9% on atomic/width/depth).

4.4 ABLATIONS

Both terms are needed. Without $\mathcal{L}_{\text{corr}}$, accepted components are still reinforced but failed ones are never taught: acceptance falls to 85.5%, a document takes 4.12 target forwards instead of 2.68, and the speedup falls to 2.5×. Without $\mathcal{L}_{\text{rl}}$, including its replay and end-of-sequence supervision, only distillation on chain positions remains: acceptance falls to 80.8%, documents shrink from 3.44 to 2.45 components, depth quality falls from 3.58 to 3.23, and the speedup falls to 1.9×. Distillation alone repairs failures but erodes what the drafter wrote well.

Without assignment by position, preservation and repair collide. In the plain sum, negative advantages are admitted, so every token of a failed component is pushed down, including the prefix before its divergence that the target endorsed and that accepted candidates share; chain positions enter at full weight, unnormalized, and the reward has no margin term. The same tokens are pulled in opposite directions and the drafter collapses: by round 6 it accepts 63.5%, writes 1.37 components and spends 6.59 forwards per document, and runs at 0.22× the target's speed, since the collapsed drafter spends almost all of its 91 s per document drafting text that is thrown away.

5 CONCLUSION

Component-level semantic acceptance changes what a drafter should learn: not to imitate the target, but to keep what it gets right and repair what fails. PREPAIR does this with what one verification forward already provides, a per-component verdict and one supervised position per divergence, sent to disjoint positions so the two never compete. On Generative UI a 1.7B drafter trained this way lifts acceptance from 81.0% to 87.8%, and the resulting system runs 3.5× faster than its target and exceeds it writing alone. Our study has several limitations. First, the reliability of the semantic verifier is uneven across criteria: while relevance, substance, and redundancy are well separated, the more context-dependent criteria of state continuity and interactivity remain difficult for the current 32B target. This limits the quality of the semantic supervision available to PREPAIR, and a stronger target model may provide more reliable judgments and repair signals. Second, our experiments use a single drafter–target pair and focus on Generative UI; whether the same preserve-and-repair principle extends to other structured generation settings, such as tool calls and structured actions, remains to be studied.

REFERENCES

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

Shuai Bai et al. Qwen3-VL technical report. *arXiv:2511.21631*, 2025.

Shrenik Bhansali and Larry Heck. Draft, verify, and improve: Toward training-aware speculative decoding. *arXiv:2510.05421*, 2025.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. MultiWOZ – a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *EMNLP*, 2018.

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

Roman Garipov, Fedor Velikonivtsev, Ivan Ermakov, Ruslan Svirschevski, Vage Egiazarian, and Max Ryabinin. AutoJudge: Judge decoding without manual annotation. In *NeurIPS*, 2025.

Google. A2UI: Agent-to-UI protocol. https://a2ui.org, 2025.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015.

Yuezhou Hu, Jiaxin Guo, Xinyu Feng, and Tuo Zhao. AdaSPEC: Selective knowledge distillation for efficient speculative decoders. In *NeurIPS*, 2025.

Fancy Kong, Congjie Zheng, Murphy Zhuang, Rio Yang, Sueky Zhang, Hao Fu, Gene Jin, Song Cao, Kaijie Chen, Andrew Chen, and Pony Ma. Macaron-A2UI: A model for generative UI in personal agents. *arXiv:2605.24830*, 2026.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *SOSP*, 2023.

Haodi Lei, Yafu Li, Haoran Zhang, Shunkai Zhang, Qianjia Cheng, Xiaoye Qu, Ganqu Cui, Bowen Zhou, Ning Ding, Yun Luo, and Yu Cheng. Draft-OPD: On-policy distillation for speculative draft models. *arXiv:2605.29343*, 2026.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *ICML*, 2023.

Yaniv Leviathan, Dani Valevski, Matan Kalman, Danny Lumen, Eyal Segalis, Eyal Molad, Shlomi Pasternak, Vishnu Natchu, Valerie Nygaard, Srinivasan Venkatachary, James Manyika, and Yossi Matias. Generative UI: LLMs are effective UI generators. *arXiv:2604.09577*, 2026.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *ICML*, 2024.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *NeurIPS*, 2025.

Siyang Liu, Chujie Zheng, Orianna Demasi, Sahand Sabour, Yu Li, Zhou Yu, Yong Jiang, and Minlie Huang. Towards emotional support dialog systems. In *ACL*, 2021.

Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. Online speculative decoding. In *ICML*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Yu-Yang Qian, Hao-Cong Wu, Yichao Fu, Hao Zhang, and Peng Zhao. When drafts evolve: Speculative decoding meets online learning. In *ICML*, 2026.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, 2023.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *AAAI*, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv:2402.03300*, 2024.

Zixiu Wu, Simone Balloccu, Vivek Kumar, Rim Helaoui, Ehud Reiter, Diego Reforgiato Recupero, and Daniele Riboni. Anno-MI: A dataset of expert-annotated counselling dialogues. In *ICASSP*, 2022.

Wenda Xu, Rujun Han, Zifeng Wang, Long T. Le, Dhruv Madeka, Lei Li, William Yang Wang, Rishabh Agarwal, Chen-Yu Lee, and Tomas Pfister. Speculative knowledge distillation: Bridging the teacher-student gap through interleaved sampling. In *ICLR*, 2025.

An Yang et al. Qwen3 technical report. *arXiv:2505.09388*, 2025.

Yuncheng Yao, Yuxuan Xia, Shengjie Wang, and Danyang Zhuo. Parallel prefix verification for speculative generation. *arXiv:2605.04263*, 2026.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv:2308.01825*, 2023.

Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. Learning harmonized representations for speculative sampling. In *ICLR*, 2025.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. DistillSpec: Improving speculative decoding via knowledge distillation. In *ICLR*, 2024.

Xiandong Zou, Jianshu Li, Jing Huang, and Pan Zhou. Variational speculative decoding: Rethinking draft training from token likelihood to sequence acceptance. In *ICML*, 2026.

A THE VERIFICATION FORWARD IN DETAIL

Figure 5 shows the sequence layout and the attention mask behind Section 3.1. The draft components and their probe branches are packed into one sequence. The mask keeps the trunk causal, so a component is judged as though no probe existed, while each branch attends to the system prompt, to the trunk up to its own component, and to itself. Verdicts are read from the branches and divergences from the trunk, in the same pass.

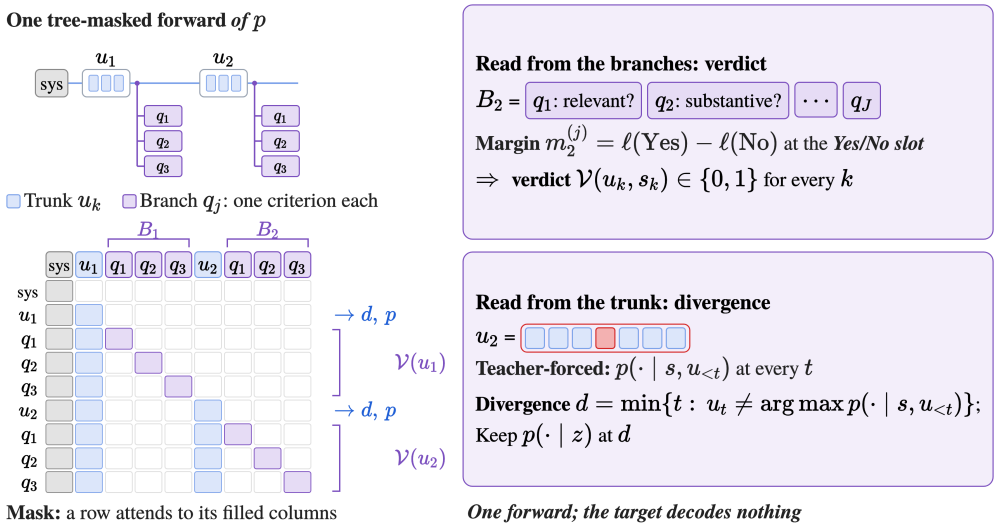


Figure 5: One verification forward. Left: after each component u_k come J probe branches B_k, one criterion q_j each, running parallel to the trunk; the attention mask keeps the trunk causal while each branch sees only the system prompt, $u_{\leq k}$ and itself. Right: the branches yield the verdict, and the teacher-forced trunk yields the first divergence and the target's distribution there. The target decodes nothing.

B ONE TRAINING ITERATION

Algorithm 1 states one iteration of the loop of Sections 3.2 and 3.3. Each pass of the inner loop verifies whatever is pending once, commits the accepted prefix and repairs only the first rejected component, so chain repair of one component unfolds over successive passes, each re-verifying the repaired component together with everything the drafter wrote after it; a component that the verifier still rejects after repair, or that does not converge within the round limit, is replaced once by a resampled draft that goes through normal verification, and if that is rejected too, the document ends at the committed prefix. Deployment runs the same loop without group sampling and without collecting $\mathcal{D}$. Every quantity the algorithm uses is read from the verification forward, and the target is never asked to decode.

Algorithm 1 One PREPAIR training iteration

$\mathcal{D} \leftarrow \emptyset$; $\mathcal{G} \leftarrow \emptyset$
for each request c in the batch **do**
  $s \leftarrow c$; draft components $u_{1:K} \sim q_\theta(\cdot \mid s)$
  while the document is not finished **do**
    one verification forward over $u_{1:K}$; $u_j \leftarrow$ first rejected ($j = K{+}1$ if none)
    commit $u_{<j}$ to s and add them to $\mathcal{G}$ as accepted components
    if u_1 was accepted outright, add a group of G candidates at slot 1 to $\mathcal{G}$
    if $j \le K$ **then**
      add a group of G candidates at slot j to $\mathcal{G}$
      add $\big(z_j,\, p(\cdot \mid z_j)\big)$ at the first divergence d_j to $\mathcal{D}$
      $u_j \leftarrow u_{j,<d_j} \,\|\, \hat{y}_{d_j}$ (Eq. 2); discard $u_{j+1:K}$, judged against the rejected u_j
    end if
    the drafter continues after s and any repaired u_j, giving new pending $u_{1:K}$
  end while
end for
update θ on $\mathcal{L} = \mathcal{L}_{\mathrm{rl}}(\mathcal{G}) + \lambda\, \mathcal{L}_{\mathrm{corr}}(\mathcal{D})$, groups rewarded by R (Eqs. 3–6)

C ADDITIONAL ABLATION AND ANALYSIS

SFT is not optional. Applied to the untrained drafter (with the one-batch recipe), PREPAIR produces a model that writes no component at all: in all 100 atomic documents it emits a bare text reply with an empty component array, nothing renders, and its 98.0% acceptance counts only that text reply, not any component. The training data already show why: on the same prompts the untrained drafter yields 29 groups with any reward variance and 58 policy-gradient samples, so there is almost nothing accepted to preserve and the reward is minimized by not drafting. PREPAIR refines a drafter within the distribution SFT gives it; it does not create that distribution.

Table 5 gives the counts behind Section 4.3: components judged at first judgment, the share accepted, and how many of the accepted ones equal the target's greedy continuation token for token. Over all 300 requests, the untrained drafter has 1,359 of its 2,300 components accepted (59.1%), and 537 of those (39.5%) are token-identical; these are the figures quoted in Section 1 and Figure 1.

Table 5: Accepted components and token agreement on the 100 atomic requests (first judgment, end-of-document votes excluded). identical: accepted components whose tokens equal the target's greedy continuation; content components carry visible text, the rest are containers and layout; rej.∧id.: rejected components that are nevertheless token-identical.

Drafter	judged	accepted	accepted %	identical % of accepted	content, identical %	rej.∧id.
untrained 1.7B	405	249	61.5%	33.3%	26.0%	0
SFT	436	376	86.2%	26.6%	41.6%	0
ONE BATCH	448	418	93.3%	34.9%	50.2%	0
PREPAIR	491	460	93.7%	32.0%	45.4%	0

D RECIPE ABLATIONS

Table 6: Recipe ablations, all starting from SFT, with changes added cumulatively; accept is on the atomic requests, at the round given in parentheses.

	Variant	accept	Avg	atomic	width	depth
	SFT (start)	86.2%	3.65	**4.07**	3.55	3.42
	ONE BATCH	93.3%	3.65	**4.07**	3.49	3.45
iterative	cold-start optimizer, 20/20/10/10 mix, chain weight 1	92.4% (8)	3.66	4.02	3.56	3.46
	+ optimizer carried across rounds	93.6% (4)	3.68	**4.07**	3.53	3.47
	+ equal request mix	92.3% (6)	3.66	3.94	3.51	3.53
	+ chain weight $0.5\times$ (= PREPAIR)	**93.7%** (7)	**3.74**	4.06	**3.66**	**3.58**

The three changes work only together. Naive on-policy iteration, restarting the optimizer every round on a 20/20/10/10 mix of MultiWOZ, SGD, width and depth requests, ends at 3.66 after eight rounds, level with one large batch and with SFT (both 3.65). Carrying the optimizer state (3.68) or adding the equal request mix (3.66) each stays within 0.02 of that baseline; only with the chain weight lowered to $0.5\times$ does the loop reach 3.74, 0.09 above its starting point. The equal mix raises depth requests to a quarter of every round, gaining 0.06 on depth and losing 0.13 on atomic against carrying the optimizer alone, and at full weight the chain positions from those harder requests overcorrect.

Late rounds lose depth quality. Where the final round is not the best, with the optimizer carried and with the equal mix, it loses 0.09 on depth against 0.07 and 0.01 overall, and PREPAIR's round 8 falls 2.7 points in acceptance below round 7. Training is scheduled for eight rounds; four further rounds raise atomic acceptance by one point while quality falls by 0.04 and depth by 0.07.

E TRAINING CURVES AND QUALITY PROFILE

Figure 6 plots the acceptance of each round's PREPAIR checkpoint on the fresh training requests it drafts in the following round, together with the six quality dimensions of Table 1 drawn on one scale. Round 7 is the highest and is deployed.

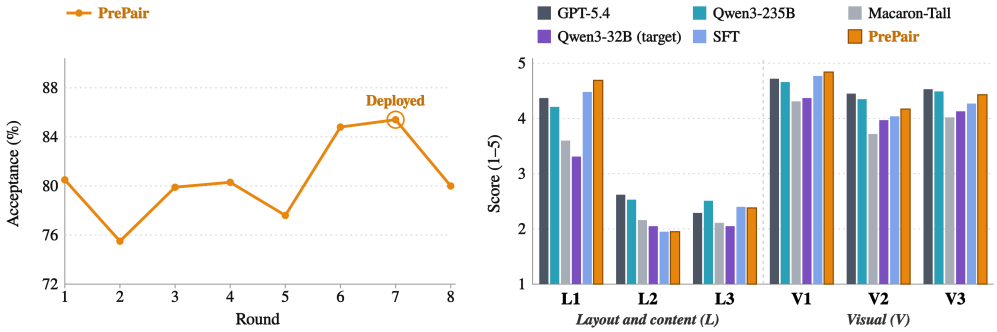


Figure 6: Left: acceptance of each round's PREPAIR checkpoint on the fresh training requests it drafts in the following round; round 7 is deployed. Right: the six quality dimensions of Table 1; PREPAIR exceeds the large direct-generation models on L1 and V1, trails them on L2, V2 and V3, and trails Qwen3-235B on L3.

Figure 7 breaks the L and V means down by request type. Against its 32B target, PREPAIR is higher on L for every type, most on depth (2.62 against 1.73); on V it is level on atomic (4.67), lower on width (4.28 against 4.42) and higher on depth (4.53 against 3.76). Against GPT-5.4 and Qwen3-235B it matches or exceeds them on width L (3.04 against 2.73 and 3.03) and on depth V (4.53 against 4.45 and 4.25), and trails them most on width V (4.28 against 4.75).

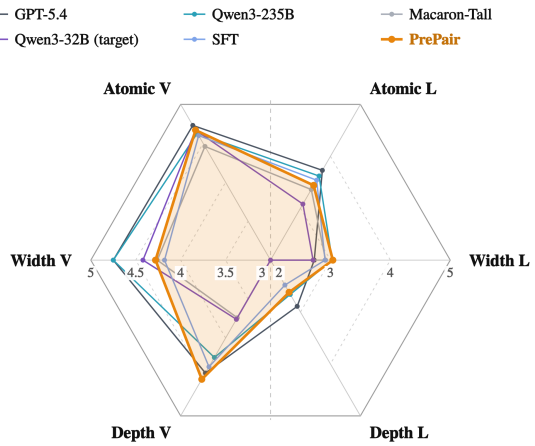


Figure 7: Mean L (L1–L3) and V (V1–V3) per request type on the Macaron-A2UI Bench. L axes (right) span 2–5 and V axes (left) 3–5; the two depth-L values below 2 are drawn at the center.

F PAIRED COMPARISON WITH THE TARGET

Table 7 pairs, request by request, the document Qwen3-32B writes directly with the one PREPAIR produces under its verification on the Macaron-A2UI Bench. The L comparison uses the mean of L1–L3 from the independent judge and covers the 278 requests scored on all three. Schema errors are counted on all 300; since the hard gates also check the schema at verification, this half compares the outputs of the two systems rather than the two models alone.

Table 7: Request-level comparison of PREPAIR with Qwen3-32B writing directly on the Macaron-A2UI Bench.

		all	atomic	width	depth
L1–L3 mean	PREPAIR higher	**165**	37	**54**	**74**
	Qwen3-32B higher	104	**39**	44	21
	tie	9	2	2	5
schema error	Qwen3-32B only	72	6	18	48
	PREPAIR only	5	1	2	2
	both	4	0	0	4

G VERIFIER RELIABILITY

Figure 8 reports, for the three content probes, the AUC of the readout margin $m^{(j)}$ at separating acceptable components from defective ones, with 95% intervals by the Hanley–McNeil formula, and the false-accept rate at the threshold that rejects 5% of the acceptable ones, the operating point used in deployment. Relevance, substance and redundancy are tested in the deployed configuration, on 107 accepted content components from the SFT drafter's documents for the 100 atomic requests, against synthetic defects: a content component copied from another document, quoted content replaced by placeholders, and a verbatim repeat of the preceding content component (buttons and links are exempt, since generic button text is not a defect). Shuffling the labels 200 times gives 0.50 for every probe. The three synthetic-defect probes clear the pre-registered threshold of 0.8 (0.95–0.99) and accept 2.8%, 5.6% and 20.0% of defects at the operating point. The two context-dependent checks, state continuity and interactivity, are less reliable for a 32B target, so the load is carried by these three probes and by the hard gates. Thresholds for all five are set at the 5% quantile of accepted components of the SFT drafter, not on the defect sets, so the verdict's false-reject rate is fixed by construction and its false-accept rate is what the figure measures.

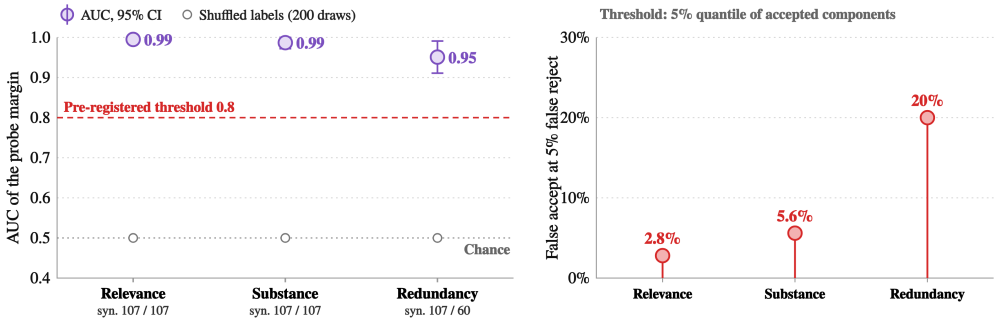


Figure 8: Reliability of the three content probes. Left: AUC of the probe margin against labeled components with 95% confidence intervals; the hollow marker is the shuffled-label control, the dashed red line the pre-registered threshold. Right: false-accept rate of the three synthetic-defect probes at the threshold that rejects 5% of acceptable components, the operating point of the deployed verdict.

H Reproducibility Statement

Upon publication we will release the code, the trained drafter checkpoints and our evaluation scripts; the setup and hyperparameters are given in Section 4.1 and Appendix B.

I Use of Large Language Models

We used large language models to aid and polish the writing of this paper, for grammar, wording and presentation, and for retrieval and discovery during the literature search, to find and summarize related work. The authors reviewed all such output, checked cited works against their sources, and take full responsibility for the content of the paper. Separately, GPT-5.4 and Qwen3-VL-235B serve as evaluators in our experiments (Section 4.1); that use is part of the evaluation protocol, not of the writing.