RISK-CONTROLLED SPECULATIVE PREFETCHING FOR TOOL-AUGMENTED DIALOGUE

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding accelerates LLM inference by predicting future tokens. We apply the same principle to future *user actions* in tool-augmented dialogue: **PRESTO** (Predictive Speculative Tool Orchestration) predicts K diverse hypotheses for the next user query, pre-executes read-only tools while the user reads the current response, and serves cached results on exact-key match when the actual query arrives — or discards with no degradation. We make three contributions. (1) The *proxy reversal* and its constructive converse. Offline embedding commit rate is a gameable proxy: predictors that maximize it degrade in-loop. The fix is *coverage-WTA*, adaptation supervised by the in-loop objective (labeling winners by realized cache hits); it beats the zero-shot predictor at equal planner budget, and no-training retrieval even when that retrieval is given a larger prefetch budget, in *both* τ-bench domains, reaching 70.1%/32.2% turn-level cache hits vs. 59.4%/23.7% zero-shot. (2) The *entity-revelation boundary*: an a-priori, per-domain ceiling on speculation (88.4% retail, 47.3% airline) that explains most remaining misses and scopes the *magnitude*, not the applicability, of adaptation's win. (3) A *risk-controlled speculation gate*: conformal risk control over a learned score operationalizing that boundary lifts fired precision to 82–87% (retail) and 50–59% (airline) over 69–70%/29–33% always-fire at the certified ε=0.10 operating point; served-result correctness comes from exact-key matching, not the gate. A $\sim$600K-parameter MCL predictor ($<$1ms) enables the study: offline-stronger frontier predictors (82.6% vs. 54.4% commit) serve *0%* in-loop at short prediction-time slack, where MCL serves 32–35%; with a 15s speculation window, MCL hides 32.6% of retail per-turn tool latency. Speculative prefetching must be evaluated, adapted, and gated *in-loop*.

1 INTRODUCTION

Multi-turn conversation with tool-augmented agents is increasingly common in production systems (Ye et al., 2025; Li et al., 2025). Each user turn may trigger multiple tool calls (database queries, API requests, document retrieval) taking seconds to tens of seconds each. This creates a tension: richer tool context improves response quality, but sequential execution makes real-time interaction impractical. Yet the user's next query is often *predictable* from context — a regularity deployed systems leave unexploited, executing tools only *after* the query arrives and paying full latency.

Speculative Prefetching. We propose to hide tool latency behind user reading time: while the user reads the current response, a background thread *speculatively* predicts the next query, pre-executes the likely read-only tools, and stores the results. When the actual query arrives, the system *commits* any matching prefetched result (the LLM still generates a fresh response) or discards and runs fresh (no worse than synchronous). The key challenge is the *commit/discard decision* under uncertainty: committing incorrectly serves stale content, discarding forfeits the savings.

Our Approach. We formalize this as a cost-sensitive decision problem motivating a cost-informed thresholding policy (Proposition 1). The framework is predictor-agnostic: any model producing K diverse predictions plugs in. We instantiate a lightweight MCL (Multiple Choice Learning) model ($\sim$600K parameters, $<$1ms) and identify what it must, and must not, be optimized for. *Zero-shot*

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.

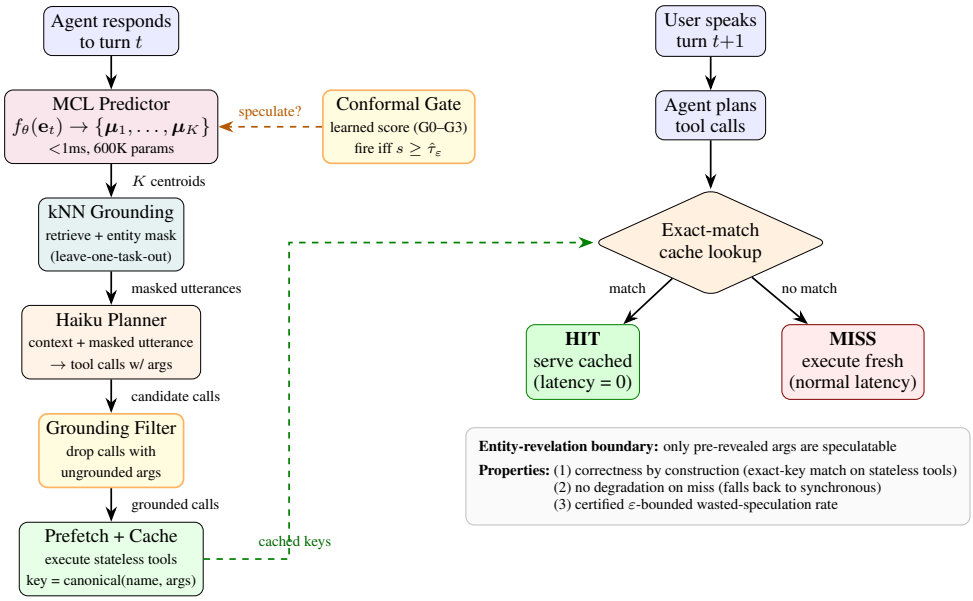


Figure 1: PRESTO architecture. **Left:** during user reading time, the MCL predictor generates K hypotheses, grounded via masked kNN retrieval and a planner LLM, filtered for entity availability, and cached by canonical key; the conformal gate decides per-turn whether to speculate, using a learned score over entity-availability features. **Right:** when the user speaks, each actual tool call is checked against the cache by exact key match — hits are served at zero latency; misses execute normally with no degradation.

MCL earns its place on activation, latency, and gateability, not raw accuracy; converting the predictor-addressable headroom below each domain's ceiling requires adaptation on the *in-loop* objective (*coverage-WTA*, §5.4), and adapting on the offline proxy instead degrades it. The same adaptation wins in both τ-bench domains, with the entity-revelation boundary scoping the win's magnitude, not its applicability (§5.4).

Contributions.

1. **Coverage-WTA — adaptation supervised by the in-loop objective**: Motivated by the *proxy reversal* — optimizing offline commit rate *degrades* in-loop hits (both $p<10^{-4}$) — we train the predictor *on* the in-loop metric: coverage-WTA reassigns the winner-take-all target by realized exact-key cache hits (self-labeled, no new supervision). It beats zero-shot MCL in *both* domains (70.1% vs. 59.4% retail, 32.2% vs. 23.7% airline, pooled) and no-training retrieval at *over*-matched budget (all $p \leq 4.3 \times 10^{-4}$; the gain is coverage, not volume), and transfers to human-authored MultiWOZ dialogue — the constraint-changing-turn gain replicates across three domains (7.3× on restaurant), no LLM in the loop.

2. **The entity-revelation boundary**: an a-priori, per-domain ceiling on speculation — set by when tool-call arguments first appear in the conversation — measured at 88.4% (retail) and 47.3% (airline) of eligible turns and dominating misses in both domains, validated on a 2-domain, 5-arm, 2-fold interventional τ-bench matrix. It scopes the *magnitude* of the win (coverage-WTA reaches 70.1%/32.2%), not which domains a predictor helps.

3. **Risk-controlled speculation gate**: a conformal gate (LTT with Clopper-Pearson bounds) over a learned score on entity-availability features, certifying a joint error rate $\Pr[\text{fire} \wedge \text{miss}] \leq \varepsilon$. On the coverage-WTA arm it gates both domains, lifting precision over always-fire in retail (82–87% vs. 69–70%) and airline (50–59% vs. 29–33%) at certified ε=0.10.

2 Related Work

Speculative Execution for LLMs and Agents. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) accelerates autoregressive LLM inference by having a small draft model propose tokens that a larger target model verifies in batches, achieving lossless speedup when drafts are accepted. Speculative Actions (Ye et al., 2025) generalizes this predict-verify pattern to agent-environment interactions, treating every API call as subject to prediction and parallelization. PASTE (Sui et al., 2026) applies pattern-aware speculation to tool-call chains, predicting the next *tool* an agent will invoke. Dynamic Speculative Planning (Guan et al., 2025) and Cost-Aware Speculative Execution (Fareed, 2026) extend this to resource-constrained settings.

PRESTO differs in three ways: (i) we speculate over *user utterances*, not tool chains or agent actions; (ii) we model the prediction as a multimodal distribution via MCL, not a point estimate; and (iii) we cast commit/discard as a cost-sensitive decision and evaluate, adapt, and gate *in-loop* rather than on an offline proxy.

Dialogue Planning and Latency. SCOPE (Chen et al., 2025) demonstrates that planning entirely within a dense semantic embedding space achieves a $70\times$ speedup over LLM-query-based simulation. SAPIENT (Du et al., 2025) applies MCTS to conversational recommendation with a self-training loop. CollabLLM (Wu et al., 2025) trains models with multi-turn aware rewards. None of these systems address the tool-execution latency bottleneck: they optimize *what* to say, not *when* to pre-compute the answer. PRESTO is orthogonal: it can wrap any dialogue planner and hide its tool latency behind speculative prefetching. No existing benchmark combines conversational turn structure with tool-execution latency: turn-based dialogue benchmarks mock tools at zero latency, while latency-realistic tool benchmarks have no conversational user whose reading time creates a speculation window. We therefore build PRESTOBench, which adds configurable tool-latency simulation and reading-time-grounded speculation windows to multi-turn scenarios (§5.1; Appendix A.14), and separately evaluate in-loop on τ-bench under the same latency model (§5.4).

Multiple Choice Learning and Multimodal Prediction. Multiple Choice Learning (Guzmán-Rivera et al., 2012; Lee et al., 2016) trains K predictors where only the winner receives gradient per example, producing diverse predictions without explicit diversity regularization; it has been applied to multimodal trajectory prediction (Cui et al., 2019; Thiede & Brahma, 2019). We apply MCL to conversational state prediction, treating user queries as the same kind of branching, multimodal future.

Decision Theory and Uncertainty in Dialogue. Cost-sensitive decision making (Elkan, 2001; Metz, 1978) turns on the cost-ratio-determined optimal threshold, well established but not previously applied to speculative execution in dialogue. Calibrated uncertainty in dialogue has targeted belief-state tracking (van Niekerk et al., 2020; 2021) and policy optimization (Tegho et al., 2017), but for *what to say*, not *whether to pre-compute*; our contribution connects commit-gate reliability to speculative regret optimality (Proposition 1).

3 Decision-Theoretic Formulation

We formalize the commit/discard decision as a cost-sensitive binary decision problem. At each turn t, after speculative prefetching has produced K candidate results, the system must choose: *commit* (serve a prefetched result instantly) or *discard* (run fresh synchronous execution).

Optimal Commit Policy. Let τ_{tool} denote tool latency (saved on commit) and C_{stale} the cost of a false commit; a discard costs τ_{tool} (the user waits). Given calibrated posteriors $p_k = P(\text{hypothesis } k \text{ correct})$, committing k costs $(1 - p_k)\, C_{\text{stale}}$ in expectation, so the Bayes-optimal policy commits iff:

$$\max_k p_k > 1 - \frac{\tau_{\text{tool}}}{C_{\text{stale}}} \triangleq \tau_{\text{cost}} \tag{1}$$

This is a normative decision rule under calibrated probabilities; our deployed commit score is cosine similarity, not a posterior, so τ_{cost} fixes no numeric threshold on the score; it motivates a permissive operating region only. We set $\tau^* = 0.3$ empirically, after confirming stable precision across the 0.25–0.35 sweep; in-loop, the binding fire decision is the calibrated gate of §5.5, not τ^*.

Regret Bound. The commit score $s_k = \cos(\mu_k, \phi(u_{t+1}))$ is a compatibility score, not a calibrated posterior; correctness comes from exact-key cache matching (§5.4), while the conformal gate (§5.5) bounds the rate of unproductive speculation. Call a commit gate ε_{rel}-*reliable* if $P(\text{stale} \mid \text{committed at } \tau^*) \leq \varepsilon_{\text{rel}}$ (Appendix A.1).

Proposition 1 (Regret under reliable speculation)**.** *For an ε_{rel}-reliable commit gate, the expected per-turn excess cost of the threshold policy relative to the no-speculation baseline is bounded by:*

$$\mathbb{E}[\textit{Regret}] \leq p_{\textit{commit}} \cdot \varepsilon_{\text{rel}} \cdot C_{\textit{stale}} \tag{2}$$

The only source of excess cost is false commits; discarded turns match the synchronous baseline exactly. Our deployed exact-key configuration (§5.4) satisfies the bound trivially; the predict-and-serve regime, deployed-mode risk accounting, and full proof are in Appendix A.1.

4 Architecture

PRESTO has two components: an MCL predictor (below) and a speculative prefetching pipeline (§4.2).

4.1 MCL Transition Model

The decision-theoretic framework requires diverse predictions over K future hypotheses. We implement this via Multiple Choice Learning (MCL) (Guzmán-Rivera et al., 2012): $f_\theta(\mathbf{e}_t) \to \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$, where $\mathbf{e}_t = \phi(s_t)$ is the current state embedding (encoded via Sentence-BERT (Reimers & Gurevych, 2019), $d = 768$) and each $\boldsymbol{\mu}_k \in \mathbb{S}^{d-1}$ is an L2-normalized centroid.

Architecture. A shared encoder (3-layer MLP, $d_h = 128$, ReLU) followed by K independent linear heads projecting to $\mathbb{R}^d$ with L2-normalization: $\boldsymbol{\mu}_k = \text{normalize}(W_k \cdot \text{encoder}(\mathbf{e}_t) + b_k)$. Total: $\sim$600K parameters, $<$1ms forward pass.

Winner-Take-All Training. Each context $\mathbf{e}_t$ is paired with multiple plausible next-utterance embeddings. The loss assigns each target to its nearest centroid and only updates that centroid:

$$\mathcal{L}_{\text{WTA}} = \min_{k \in \{1,\ldots,K\}} \|\boldsymbol{\mu}_k - \mathbf{e}_{t+1}\|^2 \tag{3}$$

This produces diverse centroids: each specializes to its nearest targets; a non-winning centroid receives no gradient on that example, so specialization emerges across examples rather than within any single update. We train on 4,428 multi-target examples from 1,404 unique conversational contexts across 9 domains; after training all 5 centroids are active and near-orthogonal (win-rate and diversity detail in Appendix A.3).

These training contexts are template-generated, so a fair question is whether the method is an artifact of synthetic data. It is not: the same coverage-WTA objective (§5.4) trained on *human-authored* MultiWOZ dialogue reproduces the win (§5.4), and the predictor transfers across scenario generators (Appendix A.8) and held-out domains (Appendix A.12). The template data buys diversity cheaply for the sandbox; the load-bearing results do not depend on it.

4.2 Multi-Hypothesis Speculative Prefetching

A "commit" serves pre-computed tool results; the LLM still generates the response, but with tool outputs already available. **Safety constraint:** PRESTO speculates only over *idempotent, read-only* tools; side-effecting operations run synchronously after confirmation (100% read-only in PRESTOBench; a classifier partitions them in deployment).

After responding to turn t, PRESTO launches K parallel background threads: (1) query the MCL for K centroids, (2) prefetch tools for each hypothesis, (3) store results, all hidden behind the user's reading time — a median 11–15s and 90th percentile 23–31s in our trajectories (45-word median turns at 238–175 wpm (Brysbaert, 2019); Appendix A.14) — a floor on the inter-turn gap, since comprehension and reply composition add to it. This covers the domains' 2–10s tools; PRESTOBench's longer 3–35s tools rely on that additional slack. Total speculation cost = $\max_k(\tau_{\text{tool},k})$, not $\sum_k$; the MCL forward pass adds $<$1ms (measured: 0.32ms on CPU; Appendix A.15).

Commit Phase. When the user submits turn $t{+}1$, the system computes $s_k = \cos(\boldsymbol{\mu}_k, \phi(u_{t+1}))$ for each k and commits the best hypothesis $k^* = \arg\max_k s_k$ if $s_{k^*} \geq \tau^*$, else discards all (pseudocode in Algorithm 1, Appendix A.2). This cosine commit is the predict-and-serve apparatus used for offline predictor evaluation (Algorithm 1). The deployed pipeline (Algorithm 2) instead fires *before* u_{t+1} on prediction-time features and establishes correctness *after* u_{t+1} by exact-key cache matching (§5.4), falling back to synchronous execution on a miss; §5 quantifies commit precision.

5 Experiments

5.1 Setup and Benchmark

PRESTOBench. Existing benchmarks (FaithDial (Dziri et al., 2022), MultiWOZ (Zang et al., 2020), ToolBench (Qin et al., 2023)) evaluate conversational quality but do not measure speculative capabilities. PRESTOBench contains 300 scenarios across 9 domains (Appendix A.4) with 1,210 total user turns (910 eligible for speculation). Conversations range from 2 to 15 turns with 38 annotated topic pivots (4.2% of eligible turns) — a pivot/ predictability structure a 783-worker crowd study independently confirms (Appendix A.22) — and simulated tool latencies of 3–35s per call (deterministic responses, configurable delay). We report three metrics: **SLR** (Speculative Latency Ratio, fraction of tool latency the user experiences; 1=no savings, 0=perfect speculation), **Commit Rate** (fraction of eligible turns where speculation commits successfully), and **PDR** (Prediction-Decision Rate: commit precision among committed turns). PRESTOBench scenarios were authored independently of MCL training data, and 332 LLM-generated scenarios from three model families confirm consistent commit rates (49.7–52.2% at $K{=}5$, spread <3pp; Appendix A.8). τ-bench (Yao et al., 2024) provides fully external validation (460 retail + 200 airline conversations, 4,580 turns).

Models. State embeddings: `all-mpnet-base-v2` ($d = 768$). MCL: 3-layer encoder, $d_h = 128$, $K = 5$ centroids, trained on 4,428 multi-target examples. LLM baseline: Claude Sonnet 4.5 with temperature sampling ($T = 0.7$) for $K > 1$.

5.2 Main Results

PRESTOBench commit rate measures predictor quality and diversity in a controlled *sandbox* — the very metric §5.4 shows *inverts* in-loop; deployed value is the τ-bench cache-hit rate. We report it first to establish the predictor, then test whether it survives contact with the loop.

$K = 5$ achieves $3.0\times$ the commit rate of $K = 1$ (54.4% vs. 18.2%; Table 1), all pairwise differences significant ($p < 10^{-4}$, McNemar's); latency-weighted SLR $= 0.494$ (50.6% of tool time saved on committed turns, charging false commits at full re-query cost).

The pivot-based PDR (0.956) mainly reflects the benchmark's 4.2% pivot base rate (always-commit: 0.958); true tool-match precision is substantially lower (Appendix A.16). $K = 5$ is optimal when background compute is free; as compute cost rises the optimum shifts toward smaller K, and by 2s/hypothesis all $K{>}1$ are net-negative and $K = 1$ is preferred (Table 6, Appendix A.5).

5.3 Predictor Landscape

We compare MCL against three predictors spanning the accuracy–latency frontier, each in its best-case format (Table 1): a fine-tuned small LM (Qwen2.5-0.5B, LoRA), Sonnet with temperature sampling ($K{=}5$ calls), and Sonnet with a single diversity-forcing prompt (1 call).

Diversity is recoverable via prompting: naïve temperature sampling mode-collapses at every scale (Qwen 3.5/5, Sonnet 3.3/5), wasting 1.5–1.7 slots on duplicates (MCL gains +20.2pp from $K{=}3{\to}5$ vs. Qwen's +7.9pp), but a single structured prompt ("list 5 distinct predictions") recovers full diversity (5.0/5) and lifts Sonnet 65.7%→82.6%: WTA's diversity is not a moat against explicit prompting. **But this 82.6% is an *offline* rate that does not transfer in-loop**: the moat is *latency*, not accuracy — MCL owns the low-slack regime at <1ms, where a multi-second predictor is infeasible (§5.4).

Table 1: PRESTOBench results (300 scenarios, 910 turns, τ^*=0.3). All pairwise K comparisons significant ($p < 10^{-4}$, McNemar's); MCL K=5 beats persistence (p=0.04); Wilson CIs and full K progression in Table 14. Pairwise cosine = inter-hypothesis similarity (lower = more diverse); unique/5 = distinct predictions/turn; K<5 rows use a fixed prefix (lower bound).

System	Commit Rate	Pairwise cos	Unique/5	Latency	Calls
Persistence (identity)	49.8%	—	—	<1ms	0
MCL K=1	18.2%	—	—	<1ms	0
MCL K=3	34.2%	—	—	<1ms	0
MCL K=5	**54.4%**	**0.137**	**5.0**	**<1ms**	0
Qwen-0.5B FT	46.9%	0.382	3.5	∼1,150ms	local
Sonnet sampled	65.7%	0.757	3.3	2,696ms	5
Sonnet diverse-forced	**82.6%**	0.301	**5.0**	3,366ms	**1**

Table 2: In-loop speculative prefetching on τ-bench (2-fold cross-eval, K=5, τ=0.3). Arm P: persistence (predict user repeats); C: context-only; A: in-domain MCL; B: zero-shot MCL. Arm A′: coverage-WTA (in-domain adaptation supervised by realized cache hits, §5.4). Ceilings: retail 88.4%, airline 47.3%. *Retail: P < C ≪ A < B (all $p<10^{-4}$; C > P at $p<10^{-10}$). Airline: P ≈ C < A ≈ B. †Persistence maximizes the offline proxy (44.9–59.6%) yet performs at or *below* the predictor-free floor in-loop. ‡A′ beats zero-shot B and no-training retrieval in both domains (pooled p's, §5.4); A′ rows pooled across folds, CIs in Appendix A.16.

Domain	Arm	Turn-level hit rate W=5s	W=15s	W=30s
Retail (460 tasks)	P (persistence)	12.6%	18.7%†	18.7%
	C (context-only)	17.4%	28.4%	28.4%
	A (in-domain, proxy-sup.)	38.9%	52.6%	52.6%
	B (zero-shot)	33.4%	59.4%*	59.4%
	A′ (coverage-WTA)	52.7%	**70.1%**‡	70.1%
Airline (200 tasks)	P (persistence)	3.5%	11.4%	11.4%
	C (context-only)	3.2%	13.9%	13.9%
	A (in-domain, proxy-sup.)	6.7%	20.8%	20.8%
	B (zero-shot)	6.0%	23.7%	23.7%
	A′ (coverage-WTA)	8.2%	**32.2%**‡	32.2%

5.4 End-to-End Speculative Prefetching on τ-bench

We insert PRESTO into the τ-bench execution loop via replay: each turn it speculates (MCL → kNN grounding → Haiku planner → stateless tool prefetch) and checks whether the agent's *actual* tool calls match the cache by exact canonicalized key. Cached tools are stateless and the simulator deterministic, so replay is equivalent to live on cache-hits and tool-wait savings (Appendix A.16). MCL trains in-domain per fold (2-fold cross-eval; best-checkpoint for arms A/B, fixed-final for A′, below; 97.9–99.4% commit from ∼600–1,600 pairs in minutes).

The proxy reversal. A persistence baseline (predict "user repeats themselves") achieves the highest offline commit rate of any predictor (59.6% airline) yet is indistinguishable from no predictor in-loop (11.4% vs. 13.9%, p=0.18) and worse than MCL (p=2.8×10^{-6}). On retail the indictment is sharper: persistence falls *below* the predictor-free floor (18.7% vs. 28.4%, p=2.4×10^{-10}) — a stale utterance actively misleads the planner relative to none. Offline commit rate is a gameable proxy; only in-loop cache-hit rate reflects deployed value.

The entity-revelation boundary. Not all stateless calls are speculatable: a call whose arguments first appear in the utterance being predicted (e.g., "my reservation is UM3OG5") is information-theoretically impossible to speculate. The *oracle ceiling* — the fraction of eligible turns whose call arguments are all pre-revealed in prior history — is 88.4% (retail, 907 eligible) and 47.3% (airline, 317); unrealizable arguments are the dominant miss category in both domains (63–77% of misses on the frozen arms, 78–84% on coverage-WTA), airline's lower ceiling reflecting higher same-turn

entity density. Computable from logs alone, the ceiling bounds the *achievable* hit rate a priori but does not decide whether a better predictor helps below it: on airline, proxy-supervised arm A and zero-shot arm B tie (McNemar p=0.20; ≈2.8pp gap, 95% CI $[-0.9, 6.5]$), both stalled near 24% against the 47% ceiling — a 23pp gap of predictor-addressable headroom, which coverage-WTA (arm A′, below) partly converts, lifting airline to ∼32% (p=4.2×10^{-5} vs. B).

Because τ-bench users are LLM-simulated, we validate the entity-revelation *diagnostic* on non-simulated dialogue: on 10,437 human-authored, human-collected MultiWOZ conversations (Zang et al., 2020) the same ceiling arithmetic produces a discriminating per-domain spread with the same domain-type ordering as τ-bench (the cross-domain *spread* and *ordering* transfer; the *level* does not, and we claim neither; Appendix A.20) — a property of dialogue, not simulated users.

Prediction activates speculation; proxy-supervised adaptation narrows it. Prediction is what turns speculation on: context-only Arm C fires at just 0.13–0.21 prefetches/turn, the planner declining without a hypothesis, while MCL raises coverage to ∼59%/24% (retail/airline B). But adaptation then cuts the wrong way: **zero-shot MCL (Arm B) significantly outperforms in-domain adaptation (Arm A)** on retail (59.4% vs. 52.6% pooled, McNemar p=4.5×10^{-5}; +6.8pp, 95% cluster-bootstrap CI $[3.8, 9.9]$pp excluding zero; direction replicates in both folds). The mechanism is speculation *volume*, not per-prefetch quality: adapted centroids cluster tightly, so dedup leaves 0.74 prefetches/turn against zero-shot's 1.90 — B buys its lead at 2.6× the volume while being *less* efficient per prefetch. This is not a budget artifact: with dedup disabled and the cap at N=5, Arm A saturates at 0.56–0.92 prefetches/turn, unable to reach B's 1.90 even in principle (Appendix A.17).

Is prediction necessary? A baseline sweep with pass/fail bars fixed in advance. We test whether MCL prediction beats non-predictive entity enumeration at matched budget. Three selectors (dialogue-state heuristic, frequency prior, no-training state-to-call retrieval) enumerate feasible calls from already-revealed entities and rank them, swept over per-turn budget. At matched budget the strongest is significantly worse than zero-shot MCL on retail (dialogue-heuristic 53.4% vs. 59.1%, −5.7pp, McNemar p=0.032; robust to task-level clustering, second fold directional; Appendix A.17). A no-training retrieval baseline matches MCL within power (−1.6pp, n.s.); the *zero-shot* predictor's value over retrieval is not raw accuracy but that it is *trainable* on the in-loop objective — coverage-WTA (below) beats this same baseline at over-matched budget, a route a parameterless index lacks. Integer-budget enumeration occupies only one of six MCL operating points; only the gated predictor reaches the low-budget points (Appendix A.17).

Training on the in-loop objective: coverage-WTA. If proxy-supervised adaptation (arm A) *degrades* in-loop, the fix is to supervise the predictor on the in-loop objective directly. *Coverage-WTA* reassigns the winner-take-all target (Eq. 3) from nearest embedding to the hypothesis whose grounded call yields a realized exact-key cache hit — labels the cache self-generates at runtime (*deployment-accruable*, no new supervision), assigned in call space so the K heads cover *distinct* feasible calls (Appendix A.18). This targets arm A's diversity collapse directly (the 0.56–0.92 pf/turn saturation above): coverage-WTA sustains 2.14–2.20 (retail).

Coverage-WTA (arm A′, Table 2) is the first adaptation in our study that helps in-loop, and it helps in *both* domains. It beats zero-shot MCL (arm B) on retail (68.9/71.2% vs. 59.1/59.7%; pooled McNemar p=2.7×10^{-11}) and airline (34.2/30.2% vs. 24.7/22.6%; pooled p=4.2×10^{-5}), both folds direction-consistent, at equal planner budget and *higher* per-prefetch precision than B — so the gain is coverage rather than volume. It also beats no-training state-to-call retrieval even when retrieval is given *more* prefetches/turn than A′: retail +8–10pp (pooled p=1.3×10^{-8}) and airline +4–11pp (pooled p=4.3×10^{-4}; Appendix A.18). That retrieval loses *with more budget* rules out a volume artifact in both domains.

The win is *additive*, not a reversal of the paper's thesis: zero-shot MCL still ties no-training retrieval in-loop (−1.6pp, n.s.), which is what justifies a <1ms, ∼600K-parameter predictor at all; coverage-WTA *adds* a raw-accuracy win on top of that latency/gateability rationale. The entity-revelation ceiling sets how large this win can be per domain, not whether adaptation helps: the same in-loop-supervised adaptation lands at 70.1% pooled on retail (ceiling 88.4%) and 32.2% on airline (ceiling 47.3%) — a +10.7/+8.5pp pooled gain whose endpoint differs with the ceiling, not a domain where adaptation fails. The win is a property of the training *objective*, not the predictor architecture: holding the objective and labels fixed and swapping only the encoder — a wider MLP, and a matched-

capacity attention encoder — reproduces the A′ win in both folds of both domains (all pooled $p<10^{-4}$), with A′'s absolute hit rate stable across backbones (Appendix A.18). Checkpoint selection is fixed-final-epoch (proxy-free); a $\{20, 50, 100\}$-epoch robustness row is in the same appendix.

Coverage-WTA transfers to human-authored dialogue. The same objective, trained on human dialogue, reproduces the in-loop win. On MultiWOZ restaurant Wizard-of-Oz dialogues (real database-grounded queries), we compare zero-shot Arm B against a coverage-WTA model trained on the MultiWOZ train split under deterministic exemplar grounding (no LLM). Since prefetching concerns state *transitions*, the turns exercising the predictor introduce or revise a constraint: there, coverage-WTA lifts exact-key hit rate from 2.1% to 15.3% (7.3×, McNemar exact $p<10^{-13}$, n=476), trading against no-change coverage (96.3% vs. 99.5% where the constraint set is unchanged, needing no prediction). Across all active-constraint turns the arms tie (78.7% vs. 78.3%, 69 vs. 60 discordant, p=0.48, n=2190). The constraint-changing-turn gain replicates on two further domains spanning high- to near-floor speculatability (hotel 19.6 vs. 13.0%, p=0.016; train 2.0 vs. 0.2%, $p=4\times10^{-4}$). The coverage/no-change trade is aggregate-neutral where speculatable updates are plentiful enough to repay it (restaurant, hotel) but net-negative on train, where constraint updates are rarely speculatable (free-text time slots; the lowest revelation-availability point of the three domains): 16 net update wins against a 118-turn no-change coverage cost, −3.7pp on active turns, $p=1.1\times10^{-19}$. Full three-arm results and a proxy-supervised control are in Appendix A.21.

Latency savings. At W=15s, retail Arm B achieves 32.6% latency reduction pooled (∼3.0s/eligible turn); airline saves ∼13%. W=5s is usable in retail (33–39% turn hits, tools 2–5s) but not airline (tools exceed the window). Net benefit is budget-dependent and positive at realistic background-compute cost; net savings cross zero at c^*=0.41/0.46 s of background compute per prefetch issued (retail/airline). Per-arm c^* and no-stale-cost accounting are in Appendix A.17.

Is a frontier LLM the better predictor? A latency-slack regime. The offline proxy ranks a diverse-forced Sonnet predictor far above MCL (82.6% vs. 54.4%, Table 1); run in-loop, feasibility is governed by prediction-time *slack* $= W - \tau_{\text{tool}} - L_{\text{plan}}$ rather than raw accuracy. At W=5s Sonnet completes nothing in time (0% in all 12 cells: L_{pred} p50 3.8s + L_{plan} 0.94s leaves ≤0.3s against ≥3s tools) while MCL (<1ms) serves 32–35%. Where slack is ample a frontier LLM is preferable (the framework is predictor-agnostic): on our measured latencies Sonnet clears both domains' tools only from $W \gtrsim$15s, the slower half of observed reading windows (Appendix A.19). But MCL owns the low-slack regime at zero latency cost and negligible *monetary* cost: the diverse-forced Sonnet predictor costs ∼\$16.5 per 1,000 turns (K=5 API calls each) against a CPU forward pass whose cost rounds to zero (Appendix A.19), so the small model wins wherever five frontier calls per turn is unaffordable — most production settings.

5.5 Reliability: Conformal Speculation Gate

Conformal risk control. Oracle-free grounding achieves only ∼5.5% result-usable PDR regardless of planner strength (Appendix A.16), motivating exact-key correctness (§5.4) and a calibrated fire decision. The fixed cosine threshold τ=0.3 is a no-op in-loop: prediction-time scores concentrate in $[0.93, 0.98]$, so it fires on 100% of turns and its precision equals the always-fire baseline (69–70% retail on A′; airline analogous). We replace it with a calibrated threshold $\hat{\tau}_\varepsilon$ carrying a distribution-free guarantee via Learn-then-Test (Angelopoulos et al., 2021; Bates et al., 2021; Angelopoulos et al., 2023) with Clopper-Pearson (CP) bounds: $\Pr[\text{risk}(\hat{\tau}_\varepsilon) \le \varepsilon] \ge 1-\delta$ (marginal rate control; δ=0.05). Risk = $\Pr[\text{fire} \wedge \text{miss}]$. The threshold is calibrated and tested on task-disjoint halves of a held-out fold, both fold-transfer directions, with the task (not the turn) as the exchangeable unit (§5.4). Task-level splitting prevents leakage; the CP certificate pools correlated within-task turns, so the guarantee is marginal over turns, not clustered by task.

On τ-bench retail (A′), at the certified ε=0.10 operating point the learned gate lifts fired precision from 69–70% (always-fire) to 82–87% (Table 3); validity holds across $\varepsilon \in [0.10, 0.30]$. On the *zero-shot* arm the raw cosine converts headroom to precision on retail (+14pp) but is uninformative on airline (AUC 0.45–0.51, scores near 0.95) — which could be mistaken for a structural ceiling. On A′ it is not: raw airline AUC is 0.56–0.64, and a learned entity-availability score reaches 0.72–0.79 (below), lifting precision to 50–59% vs. 29–33% always-fire (n=77/fold, so wide intervals). Gate utility tracks *score quality* × *headroom*, not a fixed domain ceiling.

Table 3: Learned vs. raw gate score on the coverage-WTA arm (A′), both fold-transfer directions, certified ε=0.10 operating point. Ranges span the two directions. All rows LTT-valid at ε=0.10. Airline test n=77/direction, so Fire%/Hits-kept/Prec carry wide intervals.

Domain	Score	AUC	Fire%	Hits kept	Prec@fire
Retail	raw cosine	0.62–0.64	29–36%	—	78–80%
	learned	**0.72–0.75**	**46–53%**	**57–63%**	**82–87%**
Airline	raw cosine	0.56–0.64	5%	—	50%
	learned	**0.72–0.79**	**13–22%**	**23–40%**	**50–59%**

Learned nonconformity score. The learned score recovers informativeness where the raw cosine lacks it, in both domains and (on A′) both fold-transfer directions. Features are grouped: G0 (retrieval confidence), G1 (predictor internals), G2 (dialogue position), G3 (entity availability: does the history already contain the entity types the retrieved utterances reference?). G3 operationalizes the entity-revelation boundary as a per-turn, prediction-time signal.

A logistic regression (10–16 features) is fit on one fold with task-grouped CV feature selection on that fold only; the threshold is calibrated and tested on task-disjoint halves of the other fold as above (Table 3), with Learn-then-Test run as a fixed-sequence test over descending thresholds (family-wise error controlled without further correction).

Entity-availability features (G3) carry weight with domain-appropriate vocabulary: `has_user_id` is top-weighted in both domains, while airline additionally leans on `reservation_id` and zeroes the retail-specific `order_id`/`product_id`. Feature auto-selection varies by fold-transfer direction in *both* domains (G0+G3 in one, the full G0–G3 set in the mirror), reported per-direction.

Two scoping notes. First, the learned gate's advantage differs by domain: on airline the raw cosine is already valid at ε=0.15, so the learned gain is pure utility (up to 5× more hits kept at matched risk), whereas on retail the raw cosine *violates* at ε=0.15 in one direction (test risk 0.162), so the learned score also fixes a validity failure (Appendix A.18). Second, the gate governs *speculation budget*, not served-result correctness; that comes from exact-key match, independent of the gate. With a strong predictor (A′) and entity-availability features, it is a precision instrument in both ceiling-rich (retail) and ceiling-limited (airline) domains.

5.6 Scaling, Ablation, and Robustness

Architecture (Appendix A.13): multi-hypothesis diversity is essential (single-predictor: 0% commit); the framework reaches 53–63% across families, MCL chosen for efficiency (∼600K vs. ∼3M). **Robustness** (Appendix A.9): under 40% adversarial perturbation commit drops 54.4%→44.8% while PDR *rises* to 0.961 (abstains, not mis-commits). **Generalization and scaling** (Appendix A.12): leave-one-domain-out retains 17.8±0.9% (5 seeds), PDR >0.93; quality is log-linear in data (R^2=0.95, unsaturated at 4,428 examples; Appendix A.5).

6 Conclusion

PRESTO's central finding, that offline commit rate *inverts* in-loop, establishes in-loop evaluation as the honest metric; its constructive converse, coverage-WTA, is the first adaptation in our study to win in-loop, in *both* domains. The deployed loop closes on itself: the exact-key cache self-labels every served turn for coverage-WTA training and gate recalibration, never serving stale content (fallback-to-synchronous on any miss).

Limitations and Future Work. τ-bench uses LLM-simulated users with lower entropy than real users, and PRESTOBench latencies are simulated: no corpus offers real users, real tool latency, and replayable trajectories at once. Deployed hit rates will be lower and the conformal guarantee does not transfer under distribution shift, but $\hat{\tau}_\varepsilon$ refits on a few hundred free live-traffic turns (<5 min CPU) and served-turn correctness is shift-independent (exact-key matching). Future work: distillation, online ε-adaptation, and live-API validation. Code and data: `https://anonymous.4open.science/r/PRESTO-5CFC`.

REPRODUCIBILITY STATEMENT

All in-loop results are computed by deterministic replay against the frozen τ-bench and MultiWOZ trajectories: the cache serves on exact canonicalized key match and falls back to synchronous execution otherwise, so a run is reproducible from the released checkpoints, label artifacts, and cached evaluation outputs. MCL training is CPU-only ($<$2 min, no GPU), the predictor and grounding pipeline are deterministic, and every headline contrast is reported with its statistical test, sample size, and (where pooled) both folds. The anonymized code and data release above contains the training data, checkpoints, evaluation harness, and the scripts that regenerate every table, with one exception: the Llama-3.3-70B-generated independence scenarios (395 turns in Table 12) are withheld for license-compliance reasons; the released Sonnet-4.5 and GPT-OSS-120B sets reproduce the same cross-generator check. Benchmark construction, compute budget, and hyperparameters are in Appendices A.4, A.3, and A.5.

AI USE STATEMENT

LLMs appear in this work in two distinct roles. **As studied and instrumented components of the method and experiments** (documented in the body): Claude Sonnet 4.5 and Haiku serve as the grounding planner and as the frontier-predictor baseline; Sonnet 4.5, GPT-OSS-120B, and Llama 3.3 70B generate the benchmark-independence scenarios (Appendix A.8); τ-bench users are LLM-simulated (a limitation we state explicitly); and Qwen2.5-0.5B is a fine-tuned predictor baseline. All such uses are quantified with measured latencies, costs, and sample sizes in the relevant sections. **As a writing aid**, the authors used an LLM to assist with drafting and editing the manuscript prose; all technical content, experimental design, analysis, and claims are the authors' own, and the authors take full responsibility for the paper's content.

REFERENCES

Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *arXiv preprint arXiv:2110.01052*, 2021.

Anastasios N. Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. Conformal risk control. *arXiv preprint arXiv:2208.02814*, 2023.

Stephen Bates, Anastasios Angelopoulos, Lihua Lei, Jitendra Malik, and Michael Jordan. Distribution-free, risk-controlling prediction sets. *Journal of the ACM*, 68(6):1–34, 2021.

Marc Brysbaert. How many words do we read per minute? a review and meta-analysis of reading rate. *Journal of Memory and Language*, 109:104047, 2019.

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.

Zhiliang Chen, Xinyuan Niu, Chuan-Sheng Foo, and Bryan Kian Hsiang Low. Broaden your SCOPE! efficient multi-turn conversation planning for LLMs with semantic space. In *International Conference on Learning Representations (ICLR)*, 2025.

Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

Hanwen Du, Bo Peng, and Xia Ning. SAPIENT: Mastering multi-turn conversational recommendation with strategic planning and monte carlo tree search. In *Proceedings of the 2025 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2025.

Nouha Dziri, Ehsan Kamalloo, Sivan Milton, Osmar Zaiane, Mo Yu, Edoardo Maria Ponti, and Siva Reddy. Faithdial: A faithful benchmark for information-seeking dialogue. *Transactions of the Association for Computational Linguistics*, 10:1473–1490, 2022.

Charles Elkan. The foundations of cost-sensitive learning. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 973–978, 2001.

Faisal Fareed. Cost-aware speculative execution for LLM-agent workflows: An integrated five-dimension method. *arXiv preprint arXiv:2606.07846*, 2026.

Yilin Guan, Qingfeng Lan, Sun Fei, Dujian Ding, Devang Acharya, Chi Wang, William Yang Wang, and Wenyue Hua. Dynamic speculative agent planning. *arXiv preprint arXiv:2509.01920*, 2025.

Abner Guzmán-Rivera, Dhruv Batra, and Pushmeet Kohli. Multiple choice learning: Learning to produce multiple structured outputs. *Advances in Neural Information Processing Systems*, 25, 2012.

Stefan Lee, Senthil Purushwalkam, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. Stochastic multiple choice learning for training diverse deep ensembles. *Advances in Neural Information Processing Systems*, 29, 2016.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. *arXiv preprint arXiv:2211.17192*, 2023.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*, 2025.

Charles E. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, 8(4):283–298, 1978.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

Yifan Sui, Han Zhao, Rui Ma, Zhiyuan He, Hao Wang, Jianxun Li, Kaiqiang Xu, Kai Chen, and Yuqing Yang. Parallelizing tool execution and LLM generation for low-latency agent serving. *arXiv preprint arXiv:2603.18897*, 2026.

Christopher Tegho, Paweł Budzianowski, and Milica Gašić. Uncertainty estimates for efficient neural network-based dialogue policy optimisation. *arXiv preprint arXiv:1711.11486*, 2017.

Luca Anthony Thiede and Pratik Prabhanjan Brahma. Analyzing the variety loss in the context of probabilistic trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9954–9963, 2019.

Carel van Niekerk, Michael Heck, Christian Geishauser, Hsien-Chin Lin, Nurul Lubis, Marco Moresi, and Milica Gašić. Knowing what you know: Calibrating dialogue belief state distributions via ensembles. *arXiv preprint arXiv:2010.02586*, 2020.

Carel van Niekerk, Andrey Malinin, Christian Geishauser, Michael Heck, Hsien-chin Lin, Nurul Lubis, Shutong Feng, and Milica Gašić. Uncertainty measures in neural belief tracking and the effects on dialogue policy performance. *arXiv preprint arXiv:2109.04349*, 2021.

Shirley Wu, Michel Galley, Baolin Peng, Hao Cheng, Gavin Li, Yao Dou, Weixin Cai, James Zou, Jure Leskovec, and Jianfeng Gao. CollabLLM: From passive responders to active collaborators. In *International Conference on Machine Learning (ICML)*, 2025.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Naimeng Ye, Arnav Ahuja, Georgios Liargkovas, Yunan Lu, Kostis Kaffes, and Tianyi Peng. Speculative actions: A lossless framework for faster agentic systems. *arXiv preprint arXiv:2510.04371*, 2025.

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pp. 109–117, 2020.

A Appendix

A.1 Full Proof of Proposition 1

Definition 1 (ε_{rel}-reliability)**.** *A commit gate with threshold τ^* is ε_{rel}-reliable if $P(\textit{stale} \mid \textit{committed at } \tau^*) \leq \varepsilon_{\text{rel}}$. The subscript distinguishes this gate-reliability parameter from the conformal risk budget ε of §5.5.*

Proof. We compare the threshold policy to the synchronous (no-speculation) baseline, which pays τ_{tool} on every turn. Condition on the policy's action at each turn: *(i) commit, hypothesis correct*: the policy pays 0, the baseline pays τ_{tool}; regret contribution negative. *(ii) commit, hypothesis incorrect*: the policy pays C_{stale}, the baseline pays τ_{tool}; regret contribution $C_{\text{stale}} - \tau_{\text{tool}} \leq C_{\text{stale}}$. *(iii) discard*: both pay τ_{tool}; regret contribution 0. Missed commits (a correct hypothesis scored below τ^*) fall under case (iii) and contribute zero regret against this baseline. By ε_{rel}-reliability, $P(\text{incorrect} \mid \text{commit}) \leq \varepsilon_{\text{rel}}$, so

$$\mathbb{E}[\text{Regret}] \leq p_{\text{commit}} \cdot \varepsilon_{\text{rel}} \cdot (C_{\text{stale}} - \tau_{\text{tool}}) \;\leq\; p_{\text{commit}} \cdot \varepsilon_{\text{rel}} \cdot C_{\text{stale}}. \tag{4}$$

□

Tightness. The bound is tight in the $\tau_{\text{tool}} \to 0$ limit: there exist distributions where it is approached with equality (all committed turns at exactly $\tau^* + \delta$, fraction ε_{rel} actually stale); for $\tau_{\text{tool}} > 0$ the realized regret is strictly smaller by the recovered tool latency on stale commits.

Non-Stationary Extension. If reliability varies over turns, the bound applies per-turn: $\sum_t \mathbb{E}[\text{Regret}_t] \leq C_{\text{stale}} \cdot \sum_t p_{\text{commit},t} \cdot \varepsilon_{\text{rel},t}$.

Comparison to ECE. Cosine similarity is a scoring function, not a posterior probability. With 95.8% continuation base rate, standard ECE is uninformative. Our ε_{rel}-reliability conditions on the commit event itself, directly measuring operational risk.

Scope of Instantiation. We deliberately give no numeric instantiation of Eq. 2. The natural plug-in for ε_{rel} (one minus pivot-continuation commit precision) inherits the base-rate artifact analyzed in §5: with a 95.8% continuation base rate that precision is achievable by trivial predictors, so the resulting regret number would be uninformative. In the exact-key deployment mode (§5.4) the bound is trivially satisfied ($\varepsilon_{\text{rel}} \to 0$). In predict-and-serve mode, both C_{stale} (re-query overhead plus user frustration) and ε_{rel} must be calibrated per deployment domain before the bound is used quantitatively. In the exact-key deployment mode the certified risk is $\Pr[\text{fire} \wedge \text{no valid cache hit}] \leq \varepsilon$, bounding completely-unproductive fired turns at $\varepsilon\, C_{\text{waste}}$ per turn; it does not control waste from unused hypotheses on turns that do hit. In the *predict-and-serve* regime, by contrast, $p_{\text{commit}} \cdot \varepsilon_{\text{rel}}$ bounds the joint risk $\Pr[\text{commit} \wedge \text{stale}]$, so an LTT certificate at joint budget ε yields $\mathbb{E}[\text{Regret}] \leq \varepsilon\, C_{\text{stale}}$.

Two decisions, four events. The deployed pipeline separates four events that the formulation and evaluation reference: (i) the *fire decision*, made before u_{t+1} arrives, from prediction-time features only (§5.5); (ii) the *cache-hit event*, determined after u_{t+1} by exact canonicalized key match between the agent's actual call and a prefetched one (§5.4); (iii) a *freshness condition* on the cached result (temporally stable reads); and (iv) *fallback*: on any miss the actual call executes synchronously, so a failed speculation costs wasted background compute, not served stale content. The cosine commit score (Algorithm 1) is the *offline* predict-and-serve apparatus used to evaluate predictors; it is not the deployed correctness mechanism.

A.2 Commit Decision Pseudocode

Algorithms 1 and 2 give the offline predict-and-serve and deployed exact-key pipelines, respectively.

A.3 MCL Training Details

Multi-Target Data Generation. Training data is generated from 9 domain-specific conversational templates. Each template defines 12 opener contexts, 12 deep-chain contexts, and 4 plausible

Algorithm 1 Semantic Commit Decision (predict-and-serve regime; used for offline predictor evaluation)

Require: MCL model f_θ, threshold τ^*, encoder ϕ
After turn t (background, hidden behind reading time):
  $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K \leftarrow f_\theta(\phi(s_t))$
  For each k: prefetch tools using $\boldsymbol{\mu}_k$ as intent
When turn $t{+}1$ **arrives**:
  $k^* \leftarrow \arg\max_k \cos(\boldsymbol{\mu}_k,\, \phi(u_{t+1}))$
if $\cos(\boldsymbol{\mu}_{k^*},\, \phi(u_{t+1})) \geq \tau^*$ **then**
  Commit prefetched result for k^* {cost = 0}
else
  Discard all, execute synchronously {cost = τ_{tool}}
end if

Algorithm 2 Deployed Exact-Key Pipeline

Require: MCL model f_θ, encoder ϕ, learned gate g, budget ε
After turn t (background, hidden behind reading time):
  $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K \leftarrow f_\theta(\phi(s_t))$; retrieve nearest utterances (kNN) {<1ms; no access to u_{t+1}}
  compute gate score $g(x_t)$ from features G0–G3 {G0/G1/G3 read the MCL + retrieval outputs above}
if $g(x_t) \geq \hat{\tau}_\varepsilon$ {fire} **then**
    ground, deduplicate, prefetch stateless calls for $\boldsymbol{\mu}_{1:K}$; cache by canonicalized key
end if
When turn $t{+}1$ **arrives**:
  agent plans its actual tool calls independently
if a fresh exact-key cache entry matches an actual call **then**
  reuse cached result {identical to fresh execution for stable reads}
else
  execute the actual call synchronously {miss cost = wasted background compute}
end if

alternative next-utterances per context. Additional contexts via within-domain cross-pollination yield 1,404 unique contexts $\times$ 3–4 targets = 4,428 training examples.

Embedding and Training. All utterances encoded with `all-mpnet-base-v2` (768-dim, L2-normalized). Adam (lr $= 10^{-3}$), batch size 32, 100 epochs, ReduceLROnPlateau (patience 10). Gradient clipping at norm 1.0. Training time: $<$2 minutes on CPU.

Winner-Take-All Dynamics. Final model win rates: Centroid 0: 25.8%, Centroid 1: 14.4%, Centroid 2: 11.0%, Centroid 3: 16.7%, Centroid 4: 32.1%. All 5 active. Asymmetry reflects conversational distribution: Centroid 4 captures common patterns, Centroid 2 handles rare transitions.

Why MDN Failed. We initially attempted a Mixture Density Network with Gaussian components and learnable σ. Despite entropy/repulsion regularization, the MDN consistently exhibited mode collapse. The learnable σ provides an "escape hatch" that trivializes the NLL loss without learning diverse modes. MCL eliminates this by removing σ entirely: the loss is a simple minimum distance with no probabilistic shortcut.

Compute Budget. Total compute is summarized below; all MCL experiments require no GPU.

A.4 PRESTOBENCH DETAILS

Scenario Statistics. PRESTOBench contains 300 scenarios with the following distribution:

- 9 domains: CRM (32), software engineering (34), travel (30), customer support (33), research (32), healthcare (35), legal (33), finance (35), e-commerce (36)

Component	Time	Hardware
MCL training (4,428 examples, 100 epochs)	<2 min	Single CPU
PRESTOBench evaluation (910 turns, $K = 5$)	<5 min	Single CPU
LLM baseline (910 turns $\times$ 5 calls)	$\sim$45 min	API (Claude Sonnet 4.5)

Table 4: Total compute budget. LLM baseline cost: $\sim$\$15 API credits (4,550 calls at $\sim$60 output tokens each). All MCL experiments require no GPU and reproduce from a clean environment.

- Turn distribution: 1210 total user turns, 910 eligible for speculation (excluding first turns of each scenario)
- Conversation lengths (user turns per scenario): 1 turn (61 scenarios), 2–3 turns (50), 4–6 turns (155), 7–12 turns (31), 13–15 turns (3)
- Pivot turns: 38 annotated topic pivots (4.2% of eligible turns)
- Tool latency range: 3–35s per call (domain-dependent)

Domain	Eligible Turns	Commit%	PDR	Avg CPA†
CRM	102	60.8%	0.952	0.345
Software eng.	102	38.2%	0.949	0.277
Travel	89	43.8%	0.949	0.280
Customer support	91	59.3%	0.957	0.366
Research	111	28.8%	0.938	0.265
Healthcare	101	59.4%	0.960	0.354
Legal	90	55.6%	0.960	0.326
Finance	103	55.3%	0.947	0.344
E-commerce	121	84.3%	0.961	0.423

Table 5: Per-domain performance ($K = 5$, 910 turns) varies with inherent domain predictability. Structured domains (e-commerce, CRM) achieve higher commit rates than exploratory domains (research, software engineering). †CPA = Continuation Prediction Accuracy (mean best-of-K cosine).

Per-Domain Results ($K = 5$, $\tau = 0.3$).

Tool Simulation. Tools are simulated with deterministic responses and configurable latency. This enables reproducible evaluation without external API dependencies. Each domain has 3–6 tools with latencies reflecting realistic ranges:

- Retrieval tools (search, lookup): 3–8s
- Database queries (inventory, orders, records): 5–15s
- LLM-based synthesis (summarization, recommendation): 15–35s
- External API calls (flights, pricing, regulatory): 10–25s

Annotation Schema. Each user turn is annotated with:

- `is_pivot` (bool): whether this turn represents a topic shift from the preceding conversational thread
- `stale_if_prefetched` (bool): whether prefetched content based on the prior turn's prediction would be incorrect for this turn
- `required_tools` (list): which tools should be called to properly answer this turn
- `predictable` (bool): whether this turn is inferable from prior context by a reasonable predictor
- `ambiguity_level` (float, 0–1): degree of vagueness in the user's query

Labels for the 300 hand-crafted primary scenarios were author-assigned during construction; labels for the 332 generated independence scenarios were assigned by the generating model from the annotation

schema. Neither used separate annotators. Because the pivot base rate is low (4.2%), the pivot-based PDR is used only as a descriptive sandbox diagnostic — the load-bearing precision claims rest on the grounded tool-match analysis (§5.5), which does not use these labels, and on exact-key cache correctness in deployment, which is label-independent by construction. PDR (Prediction-Decision Rate, i.e., commit precision) is operationalized via this schema: a commit is marked *correct* if and only if the turn's `stale_if_prefetched` annotation is `False`. A false commit occurs when the system commits on a turn where `stale_if_prefetched = True`. **Note:** this pivot-based PDR reflects the 4.2% pivot base rate and does not measure actual tool-match precision (which requires grounding; see §5.5). In deployment, correctness is ensured by exact-key cache matching, not by the cosine gate.

Example Scenario (CRM, 5 turns).

```
Turn 1: "help me prepare for this meeting"
  -> Tools: [account_overview, recent_interactions]
  -> predictable: True, is_pivot: False
Turn 2: "what about the competitive situation"
  -> Tools: [competitive_analysis]
  -> predictable: True, is_pivot: False
Turn 3: "go deeper on the AI angle specifically"
  -> Tools: [competitive_analysis(filter=AI)]
  -> predictable: True, is_pivot: False
Turn 4: "actually, what's the support situation?"
  -> Tools: [support_cases]
  -> predictable: False, is_pivot: True
Turn 5: "which ones are high severity?"
  -> Tools: [support_cases(filter=severity)]
  -> predictable: True, is_pivot: False
```

At Turn 4, the user pivots from competitive analysis to support. PRESTO correctly discards speculation (the pivot utterance produces cosine $< \tau^*$ with all centroids). At Turn 5, the continuation is predictable and PRESTO commits successfully.

A.5 Additional Experiments

Cost-Efficiency of K. Table 6 gives net benefit per turn (seconds, latency-weighted, τ_{tool}=15s) under varying background compute cost c_{compute}. As compute cost rises the optimum shifts from K=5 toward smaller K (at c_{compute}=1s/hypothesis K=2 and K=5 tie at the top); at $c_{\text{compute}} \geq$ 2s/hypothesis, all $K > 1$ are net negative and $K = 1$ is preferred.

Table 6: Net benefit per turn (s) vs. K under varying background compute cost.

c_{compute} (s/hyp)	K=1	K=2	K=3	K=4	K=5
0.0 (free)	2.14	3.41	4.03	5.33	**6.41**
0.5	1.64	2.41	2.53	3.33	**3.91**
1.0	1.14	**1.41**	1.03	1.33	**1.41**
2.0	**0.14**	−0.59	−1.97	−2.67	−3.59

OOD Transfer. Zero-shot on τ-bench, MCL and Qwen converge (retail tie p=0.32; Qwen wins airline $p<10^{-7}$, at ∼1,150ms vs. <1ms); both lag persistence's offline rate, and cheap domain adaptation (§5.4) restores 97.9–99.4% commit (Table 7). Zero-shot MCL commits only 23.0–26.2% OOD because effective K collapses from 5 to ∼2.2–2.5; in-domain adaptation restores effective K to 4.7, lifting commit rate to ∼98% but at the tight-cluster cost that caps its in-loop coverage (§5.4).

Threshold Sensitivity. The cost-ratio quantity $\tau_{\text{cost}} = 1 - \tau_{\text{tool}}/C_{\text{stale}}$ is determined by deployment costs. With $\tau_{\text{tool}} = 15$s and $C_{\text{stale}} = 20$s: $\tau_{\text{cost}} = 1 - 15/20 = 0.25$. In practice, C_{stale} is imprecise (it depends on user patience, re-query cost, and context), so we set the operating threshold $\tau^* = 0.3$, slightly conservative within the derived range. The Pareto trade-off is smooth across 0.25–0.35

Table 7: OOD transfer on τ-bench (zero-shot, no retraining, τ=0.3). Persistence = "predict user repeats themselves" (zero parameters).

System	Retail (n=2,136)	Airline (n=1,290)	Latency
Persistence	44.9%†	59.6%†	<1ms
MCL K=5	26.2%	23.0%	<1ms
Qwen-0.5B K=5	27.3%	31.9%**	∼1,150ms

** McNemar $p < 10^{-7}$; retail non-significant (p=0.32).
† Persistence dominates the proxy but is worst in-loop (11.4% airline, §5.4).

Table 8: Commit rate and pivot-PDR at various thresholds ($K = 3$, Gate 2 only). Pivot-PDR remains >0.93 (reflects 4.2% pivot base rate; see §5.5 for tool-match PDR).

τ	Commit Rate	PDR	Interpretation
0.20	65.8%	0.953	Aggressive
0.25	48.6%	0.955	Moderate
0.30	**34.2%**	**0.952**	**Operating point**
0.35	23.5%	0.953	Conservative
0.40	15.1%	0.949	Very conservative
0.50	6.5%	0.932	Near-zero speculation

(commit rate varies from 48.6% to 23.5% with precision consistently above 95%), confirming the system is robust to the exact C_{stale} estimate.

Data Scaling. Retraining on nested subsets of the multi-target training data (442 / 1,107 / 2,214 / 4,428 examples, i.e. 10–100%) gives K=5 commit rates of 34.1% / 40.8% / 44.2% / 54.4%, well fit by $\text{commit} = a \ln N + b$ (R^2=0.95; K=3: 14.8% → 34.2%, R^2=0.96), with no saturation at the full data size. K=1 saturates by 2,214 examples (18.2% at both larger sizes), so the marginal value of data grows with K: more centroids leave more modes for new examples to claim.

Gate Ablation. We ablate the commit gate design by testing three configurations at $K = 3$ (prediction-only, trajectory-only, dual gate). The trajectory stability gate (cosine between consecutive user turns) removes 136 commits to prevent only 9 false commits, a 14:1 ratio of true commits killed per false commit caught. PDR changes by only +0.014 with the dual-gate. We therefore use the prediction-only gate as the default: the MCL's cosine similarity already captures whether the predicted future matches the actual query, making a separate trajectory check redundant.

Latency Scaling. The net benefit of speculation scales linearly with tool latency:

$$\text{Net savings/turn} = p_{\text{commit}} \cdot \text{PDR} \cdot \tau_{\text{tool}} - p_{\text{commit}} \cdot (1-\text{PDR}) \cdot C_{\text{stale}} \tag{5}$$

The false-commit term charges the full C_{stale} rather than the excess $C_{\text{stale}} - \tau_{\text{tool}}$ of Appendix A.1; savings under this model are therefore conservative relative to that accounting.

Table 9: Theoretical net savings under the pivot-PDR model (upper bound; real tool-match precision is lower, see §5.5). Break-even: $\tau_{\text{tool}} > 0.9$s.

τ_{tool}	Net (K=1)	Net (K=3)	Net (K=5) / Speedup
5s	0.7s	1.3s	2.1s / 1.72×
15s	2.4s	4.6s	7.3s / 1.95×
30s	5.0s	9.4s	15.1s / 2.01×
60s	10.1s	19.2s	30.7s / 2.05×

At $\tau_{\text{tool}} = 60$s (complex API chains), PRESTO ($K = 5$) saves ∼31s per turn, a 2.05× effective speedup purely from speculation. The regret cost from false commits (zero by construction in exact-key mode) is negligible relative to gains regardless of tool latency. PRESTO's value proposition strengthens in exactly the settings where latency matters most.

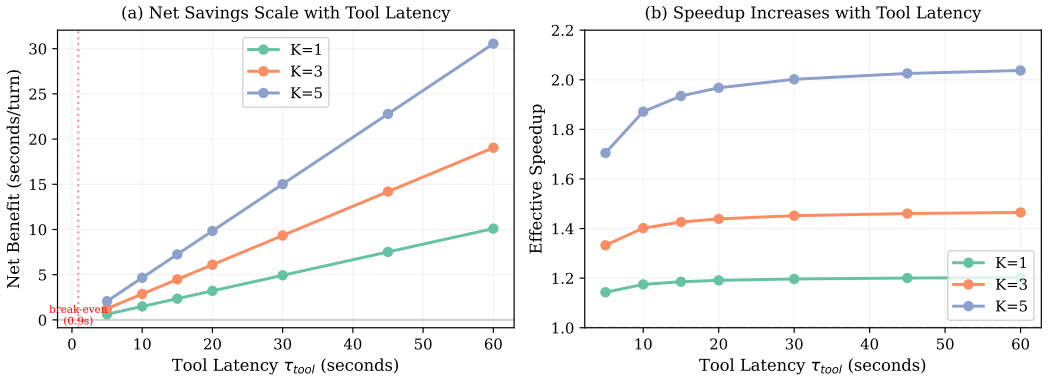


Figure 2: (a) Net savings per turn scale linearly with tool latency. (b) Effective speedup asymptotes to $\sim 2\times$ at $K = 5$ as $\tau_{\text{tool}} \to \infty$. Break-even at $\sim$0.9s (easily satisfied).

Operating Regime. PRESTO provides the largest benefit in settings with: (i) tool latencies > 5s, (ii) predictable multi-turn trajectories of 4+ turns, and (iii) multimodal but structured conversation patterns. On single-turn benchmarks where speculation cannot fire (FaithDial (Dziri et al., 2022)), PRESTO adds no overhead (FaithCritic 0.474 vs. 0.467 baseline). In single-tool domains where a strong LLM solves the task natively (20 Questions), speculation adds unnecessary complexity. These results characterize PRESTO's operating envelope: it is designed for latency-bound, multi-tool, multi-turn settings — not as a general dialogue improvement.

A.6 Analysis Details

What Governs Commit Success? To understand the factors determining commit rate, we fit a logistic model predicting per-turn commit outcomes from: turn depth, best Gate 2 score, conversational velocity (cosine between consecutive utterances), and scenario length. Commit success is determined almost entirely by a single variable: the Gate 2 prediction quality ($\beta_{\text{standardized}} = 8.6$, all other features $|\beta| < 0.2$). This supports that the gate operates as designed: the threshold effectively partitions turns by prediction quality.

Score-Stratified Analysis. A score-stratified analysis shows: *within Gate 2 score bins, turn depth has zero residual association with commit rate*. Within any Gate 2 bin, early turns and late turns commit at identical rates. The observed turn-depth decline (60% at depth 0–1 vs. 36% at depth 5+, at $K = 5$) is entirely accounted for by declining prediction quality at deeper turns (mean Gate 2 drops from 0.35 to 0.28).

Table 10: Gate 2 score accounts for the turn-depth effect. Within each Gate 2 bin, commit rate is identical regardless of turn depth.

Gate 2 Score	Early (depth 0–2)	Late (depth 4+)	Difference
$[0.2, 0.3)$	0%	0%	0
$[0.3, 0.4)$	100%	100%	0
$[0.4, 0.6)$	100%	100%	0

This result has two implications: (1) the commit gate is operationally effective, the threshold $\tau^* = 0.3$ partitioning turns by prediction quality; (2) the "turn-depth problem" is not a fundamental limitation of speculation but a training-data coverage problem. Improving the MCL's predictions at deeper turns (via domain-specific training data) would directly increase commit rate, as the gate will faithfully commit any prediction that crosses the threshold.

Domain Variation: A Descriptive Decomposition. Commit rate varies across domains (e-commerce: 84%, research: 29%). The Pearson correlation between domain-level mean Gate 2 score and commit rate is $r = 0.972$: all 9 domains fall on a single curve relating prediction quality

to speculation success. Structured transactional domains (e-commerce: mean Gate 2 $= 0.42$) have inherently more predictable progressions than open-ended exploratory domains (research: 0.27).

A linear model predicting Gate 2 score from conversation properties ($R^2 = 0.05$) identifies backward velocity (topic continuity, $\beta = 0.08$) and the `predictable` annotation ($\beta = 0.02$) as positive predictors. This is consistent with the descriptive chain: *conversation properties → prediction quality (Gate 2) → commit success → latency savings*. All apparent associations of depth, domain, and scenario structure are accounted for by a single variable: how well the MCL's centroids match the actual next utterance.

Why Multi-Hypothesis Helps. Single-hypothesis speculation ($K = 1$) requires the transition model to predict a centroid within cosine ≥ 0.3 of the *exact* next query. With $K = 5$, the system only needs *any one of five* predictions to be close. The MCL's winner-take-all training ensures the five hypotheses cover different regions of embedding space (average pairwise cosine 0.137), rather than being minor variations of the same prediction.

Under independence, $P(\text{any match at } K) = 1 - (1 - p_1)^K$. With $p_1 = 0.182$ (empirical $K = 1$ commit rate): predicted $K = 5$ commit rate is $1 - (1 - 0.182)^5 = 0.633$. Actual is 0.544 — the gap reflects correlation between centroids (they share a context encoder). Nonetheless, the scaling from 0.182 to 0.544 ($3.0\times$) is substantial and practically meaningful.

A.7 Turn-Index Analysis

Commit rate varies with conversational depth:

Turn Depth	N	Commit Rate	Avg Gate 2 Score
Turn 0→1	239	37.2%	0.352
Turn 1→2	219	34.2%	0.357
Turn 2→3	189	30.2%	0.331
Turn 3→4	91	29.7%	0.329
Turn 4→5	45	17.8%	0.284
Turn 5+	127	15.0%	0.276

Table 11: Commit rate by turn depth ($K = 3$, $\tau = 0.3$). Performance is highest at turns 1–3 and decreases for deeper turns.

This pattern reflects the MCL's training distribution: the 4,428 examples are derived from 1–2 turn conversational templates, so early-turn patterns are better covered. Later turns in long scenarios involve domain-specific deep-dives (e.g., turn 8 of a 12-turn medical consultation) that are less predictable from the template-based training data.

Similarly, medium-length scenarios (4–6 turns) achieve the highest commit rate (31.9%), compared to long scenarios (11–15 turns: 8.6%). This is consistent with the turn-depth finding: longer scenarios have proportionally more deep turns where predictions are weaker.

This analysis suggests a clear path for improvement: expanding training data with longer conversational trajectories (7–15 turns, domain-specific) would likely increase deep-turn commit rates. We leave this to future work.

A.8 Benchmark Independence Validation

The 300 primary scenarios are hand-crafted across 9 domains and six construct categories. To verify that PRESTOBench results are not an artifact of hand-authored scenario design or any single model's conversational style, we additionally generated 332 scenarios using three model families: Claude Sonnet 4.5 (180 scenarios), GPT-OSS-120B (62 scenarios), and Llama 3.3 70B (90 scenarios), each from domain/tool specifications only (the GPT-OSS and Llama runs additionally cycled through author-written one-line situation seeds to force opener diversity), without exposure to the hand-crafted scenarios or each other's outputs.

Generation Procedure. For each model and each of the 9 domains, the LLM received a prompt containing only (1) the domain name, a brief description, and a list of typical task types, (2) the list of available tools with their signatures, and (3) a turn/pivot-count and the annotation schema. The GPT-OSS and Llama runs additionally drew from a per-domain list of author-written situation seeds and instructions to avoid generic openers and vary tone/style. No hand-crafted scenarios, example conversations, or MCL training data were included. Each model generated conversations independently, producing 332 total scenarios with 1,649 eligible turns across three model families. The Sonnet-4.5 and GPT-OSS-120B scenario sets are included in the code and data release; the Llama-3.3-70B set is excluded from distribution for license-compliance reasons, and only its aggregate statistics are reported (Table 12).

Table 12 compares results across all four scenario sources.

Table 12: Commit rates are consistent across the hand-crafted primary set and three additional generator families (generator spread <3pp at $K = 5$; all-source spread <5pp), confirming benchmark design does not flatter the method.

		$K = 1$		$K = 5$	
Source	Turns	Commit	PDR	Commit	PDR
Hand-crafted	910	18.2%	0.946	54.4%	0.956
Claude Sonnet 4.5	1,024	23.4%	0.880	49.7%	0.921
GPT-OSS 120B	230	24.3%	—	52.2%	0.808
Llama 3.3 70B	395	24.1%	—	51.6%	0.770

Commit rates are consistent across all sources: the three LLM generators produce 49.7–52.2% at $K = 5$ (spread <3pp), confirming that the MCL's predictions transfer regardless of scenario-authoring model. Pivot-PDR is lower on the additional-generator scenarios (0.77–0.92 vs. 0.956 on the hand-crafted primary set) because their pivots use more natural, ambiguous language that produces higher cosine similarity with predicted centroids, a harder test for the commit gate's pivot-discrimination ability. Note that all pivot-PDR values should be interpreted against their respective base rates (see §5.5 for the tool-match PDR that measures actual grounding precision). Claude Sonnet 4.5 also serves as the diverse-forced predictor baseline (Table 1); its role as one of the three *independence* generators is the point, not a confound — those 180 scenarios were produced blind to the hand-crafted set expressly to test for authoring-model bias, and the PRESTOBench results (the sandbox commit-rate metric) derive from the hand-crafted primary set, which shares no authoring model with any predictor.

A.9 Robustness Under Adversarial Perturbation

To test whether PRESTO degrades gracefully on noisy, realistic input, we perturb 40% of turns in the 300 hand-crafted PRESTOBench scenarios with five adversarial perturbation types: typos (character swaps/deletions), vague follow-ups ("tell me more"), mid-sentence corrections ("actually no wait —"), ultra-short fragments ("yes", "ok", "next"), and conversational filler ("umm so like...").

Table 13: Stress-test results ($K = 5$, $\tau = 0.3$, 40% perturbation rate). Pivot-PDR remains ≥0.955 across all conditions: the gate degrades gracefully by committing less under noise (commit rate drops) while pivot-level precision holds.

Condition	Commit Rate	Δ vs. Base	PDR
Baseline (clean)	54.4%	—	0.956
Typos	50.9%	−3.5pp	0.961
Corrections	53.2%	−1.2pp	0.957
Filler	53.7%	−0.7pp	0.955
Mixed (all types)	44.8%	−9.6pp	0.961
Vague follow-ups	25.2%	−29.2pp	0.956
Ultra-short	29.7%	−24.7pp	0.978

Interpretation. The commit gate is inherently robust to noisy input. Under mixed adversarial perturbation, commit rate drops 9.6pp (from 54.4% to 44.8%) while PDR *increases* to 0.961, because perturbed turns produce lower cosine scores, the gate abstains more aggressively, and the remaining commits are the highest-confidence predictions. Typos, corrections, and filler have minimal impact (<4pp) because the sentence encoder is robust to surface noise. Vague and ultra-short turns show larger drops (25–29pp) because they genuinely lack the semantic content needed for prediction — but the gate correctly identifies this and discards rather than committing stale content.

This supports a key deployment property: *under adversarial noise, PRESTO abstains more rather than committing incorrectly*. Adversarial or out-of-distribution user behavior reduces speculation coverage while maintaining high precision in observed conditions.

A.10 Failure Analysis

False Commits (15 turns at K=3, 910 eligible). All false commits occur on pivot turns where the best centroid happens to have cosine ≥ 0.3 with the pivot utterance. These are "soft pivots" where the topic changes but the utterance phrasing is superficially similar to a continuation (e.g., "actually, what about the support situation?" has high cosine with "what about the competitive situation?" due to shared "what about X" structure). The false commit rate (1.6%) is low because most pivots produce centroids below threshold.

Missed Commits (599 turns at K=3). Turns where the system discards but could have committed (non-pivot turns below threshold). The primary cause is domain-specific vocabulary: predictions trained on template language ("what are the risks") don't match actual user phrasing ("how dangerous is this option"). The semantic meaning is equivalent but the cosine distance exceeds τ^*. This is a training-data coverage issue, not a fundamental limitation, as shown by the predictor-quality scaling analysis (§5).

A.11 Hyperparameter Sensitivity

Commit Threshold τ^*. The operating threshold is derived from the cost ratio: $\tau_{\text{cost}} = 1 - \tau_{\text{tool}}/C_{\text{stale}}$. Table 8 shows commit rate at various τ values, demonstrating smooth Pareto trade-off. The cost-ratio-derived value ($\tau_{\text{cost}} = 0.25$ for $\tau_{\text{tool}} = 15$s, $C_{\text{stale}} = 20$s) operates in the productive range; we set the operating threshold $\tau^* = 0.3$ (slightly conservative) as the default. Deployments with higher stale costs (e.g., medical or legal domains) should use higher thresholds; low-stakes domains (e.g., travel recommendations) can operate at $\tau = 0.20$ for higher commit rates with acceptable precision (95.1%).

Table 14: Full K ablation on PRESTOBench (300 scenarios, 910 turns, $\tau^* = 0.3$). 95% Wilson CIs in []; K=2/K=4 CIs computed from reported rates. All pairwise K comparisons significant at $p < 10^{-4}$ (McNemar's). MCL K=5 significantly beats persistence (p=0.04, McNemar).

System	Commit Rate [95% CI]	Inference
No speculation	0.0%	—
Persistence (identity)	49.8%	<1ms
PRESTO ($K = 1$)	18.2% [15.9, 20.9]	<1ms
PRESTO ($K = 2$)	28.9% [26.0, 31.9]	<1ms
PRESTO ($K = 3$)	34.2% [31.2, 37.3]	<1ms
PRESTO ($K = 4$)	45.2% [42.0, 48.4]	<1ms
PRESTO ($K = 5$)	**54.4% [51.1, 57.6]**	<1ms

Number of Components K. Table 14 shows the full $K \in \{1, 2, 3, 4, 5\}$ ablation. Returns are diminishing but monotonic: each additional centroid adds ∼5–11% absolute commit rate. The marginal benefit of $K = 4 \to K = 5$ (+9.2%) is smaller than $K = 1 \to K = 2$ (+10.7%), consistent with diminishing returns as the centroids begin to overlap. We recommend $K = 3$–5 depending on available background compute bandwidth. Beyond $K = 5$, we expect minimal further gains given the embedding space geometry (alternatives are spread at cosine ∼0.2; 5 well-placed centroids already cover the main modes).

MCL Hidden Dimension. The model ($\sim$600K parameters at $d_h = 128$) is insensitive to architecture choices:

- $d_h = 64$: marginally worse (Avg CPA 0.268 vs. 0.274)
- $d_h = 128$: default (reported results)
- $d_h = 256$: no improvement (Avg CPA 0.273), $4\times$ parameters

The bottleneck is training data diversity, not model capacity. The 768-dim input space with 4,428 examples is well-served by a compact encoder.

Context Window for State Embedding. The MCL encodes context as a concatenation of the last 3 turns ("$u_{t-2} \mid u_{t-1} \mid u_t$"). We found that:

- Last 1 turn: Avg CPA 0.251 (insufficient context)
- Last 3 turns: Avg CPA 0.274 (reported results)
- Last 5 turns: Avg CPA 0.270 (no benefit; longer context dilutes the embedding)

The sentence encoder (`all-mpnet-base-v2`) has a 384-token context limit; 3 turns typically fits within this window.

A.12 Leave-One-Domain-Out Evaluation

To verify that PRESTOBench results do not merely reflect the MCL memorizing domain-specific templates, we perform a leave-one-domain-out (LODO) evaluation: for each of the 9 domains, we retrain the MCL on training data from the remaining 8 domains (3,936 examples) and evaluate on PRESTOBench scenarios from the held-out domain.

Table 15: Leave-one-domain-out results ($K = 5$, $\tau = 0.3$; LODO commit rate mean $\pm$ std over 5 seeds). LODO retrains the MCL from scratch per fold, so we report seed variation; the pooled average is stable ($17.8 \pm 0.9\%$) while per-domain rates vary more. Full column is the LODO harness's own full-training run (5-seed retrain average $56.2 \pm 1.9\%$); per-domain values differ from Table 5 within retraining variance (aggregate 55.1% vs. 54.4%). PDR remains high on all held-out domains with non-zero commits.

Held-out Domain	Full (%)	LODO (%)	Δ	PDR (LODO)
CRM	64.7	31.0 ± 6.9	-33.7	0.962
Software eng.	48.0	10.2 ± 1.1	-37.8	1.000
Travel	30.3	0.0 ± 0.0	-30.3	—
Customer support	58.2	22.9 ± 5.9	-35.3	0.942
Research	35.1	14.6 ± 3.1	-20.5	0.963
Healthcare	61.4	5.1 ± 1.9	-56.3	1.000
Legal	54.4	29.6 ± 2.2	-24.8	0.955
Finance	60.2	10.3 ± 3.4	-49.9	0.943
E-commerce	77.7	33.2 ± 2.9	-44.5	0.965
Average	**55.1**	$\mathbf{17.8 \pm 0.9}$	$\mathbf{-37.3}$	**0.966**

Interpretation. The average LODO commit rate of $17.8 \pm 0.9\%$ (5 seeds) demonstrates that approximately 32% of in-distribution performance ($17.8/55.1$) transfers as general conversational structure: question-answer patterns, follow-up types, and information-seeking sequences that are domain-agnostic. The remaining 68% requires domain-specific vocabulary and conversational templates.

Crucially, **PDR remains >0.93 on all held-out domains with non-zero commits**. The commit gate abstains appropriately under domain shift: when the MCL is uncertain (OOD predictions produce lower cosine scores), it correctly abstains rather than making false commits. This safety property is essential for deployment: a model deployed on a new domain will simply commit less often, not commit incorrectly.

Domain-Shift Gradient. The results establish a gradient of domain shift:

- Full training (in-distribution): 55.1%
- τ-bench (external benchmark, partially overlapping domains): 23–26%
- LODO (held-out domain, zero semantic overlap): 17.8%

τ-bench outperforms LODO because its domains (retail, airline) share partial vocabulary with training domains (customer support, e-commerce, travel): the MCL benefits from this incidental semantic transfer. LODO is the harder test: a domain held out entirely from training shows minimal overlap with the remaining 8. This suggests that domain-specific MCL training on target conversation logs would recover full in-distribution performance: the predictor-quality scaling story (§5) generalizes to the cross-domain setting.

Per-Domain Variation. Domains with more formal/transactional language (e-commerce: 33.2%, CRM: 31.0%, legal: 29.6%) transfer better because they share structured question-answer patterns with other transactional domains in training. Specialized domains with unique vocabulary (travel: 0%, healthcare: 5.1%, software engineering: 10.2%) show minimal transfer: their conversational patterns involve domain-specific terminology (flight itineraries, medical workups, code review workflows) absent from other domains.

A.13 Architecture Ablation

We ablate the prediction component within the PRESTO framework to isolate the contribution of multi-hypothesis diversity from the specific predictor architecture.

Table 16: Predictor design space ($K = 5$, $\tau = 0.3$, 910 turns). Multi-hypothesis diversity is essential (single-predictor: 0%). The framework achieves 53–63% regardless of predictor architecture, with MCL as the parameter-efficient default.

Method	Commit Rate	Parameters	Inference
Single predictor + noise	0.0%	∼120K	<1ms
kNN retrieval (5-nearest)	53.1%	— (index)	∼5ms
MCL (shared encoder + WTA)	54.4%	∼600K	<1ms
K-independent MLPs	62.6%	∼3M	<1ms

Interpretation. Multi-hypothesis diversity is the critical mechanism: single-predictor perturbation (Gaussian noise around a learned mean) produces zero successful commits, confirming that diversity must be structurally learned, not sampled. Given diverse predictions, the PRESTO framework achieves 53–63% commit rate *regardless of predictor architecture*, demonstrating that the speculation framework (commit gate + parallel prefetching + cost-derived threshold) is the primary contributor, with the specific predictor as a secondary design choice.

K-independent MLPs (5 separate models, no shared encoder) achieve the highest commit rate (62.6%) by eliminating centroid correlation, but at 5× the parameter budget (∼3M vs. ∼600K). MCL achieves 87% of K-independent performance with a single unified model, the best measured accuracy–cost trade-off under the deployment constraint of <1ms inference with minimal memory overhead. kNN retrieval (53.1%) approaches MCL without any learning, suggesting that at the current training scale (4,428 examples), memorization captures most available signal; the scaling law analysis predicts this gap widens with additional data.

Centroid Specialization. Winner-take-all training produces centroids that specialize primarily by domain: Centroid 1 captures 56% healthcare examples, Centroid 3 captures 47% travel, Centroid 4 captures research (26%) and software engineering (24%). This reflects that next-utterance distributions differ more across domains than across conversational moves within a domain. The domain specialization is functional: it allows each centroid to capture domain-specific continuation patterns, which is why removing a domain from training (LODO, Appendix A.12) disproportionately affects commit rate.

A.14 READING-TIME WINDOW GROUNDING

The speculation window W is bounded below by the user's inter-turn gap, whose floor is reading time. We measure the length of the assistant text preceding each user turn across all n=4,596 such turns in the τ-bench trajectories (retail + airline) and convert to silent-reading time at the meta-analytic non-fiction rate of 238 wpm and a slow-reader bound of 175 wpm (Brysbaert, 2019). Turns are short (median 45 words); the resulting windows (Table 17) place our injected grid $W \in \{5, 15, 30\}$s so that it brackets the empirical distribution: W=5s is a hostile sub-median window (requiring $\sim$540 wpm), W=15s sits at the median-to-slow reader, and W=30s covers the 90th percentile. Reading time is a strict floor on the inter-turn gap—comprehension and reply composition are additive—so deployable windows are at least this large.

Table 17: Length of assistant text preceding each user turn (τ-bench, n=4,596) and implied silent reading time at 238/175 wpm (Brysbaert, 2019).

	words	@238 wpm	@175 wpm
median	45	11s	15s
90th pct.	91	23s	31s
99th pct.	155	39s	53s
max	268	68s	92s

A.15 MEASURED LATENCY

To ground latency claims in measured wall-clock time, we profile each component of the PRESTO decision path on a single CPU core (AWS SageMaker, ml.m5.xlarge).

Table 18: Measured component latencies (20 sample turns, 5 repeats). PRESTO's decision overhead is 0.3% of a representative tool call.

Component	Mean	P95
Context encoding (Sentence-BERT)	9.1ms	9.2ms
MCL forward pass ($K = 5$)	0.32ms	0.33ms
Cosine gate decision	0.04ms	0.04ms
Full pipeline (encode + MCL + gate)	**9.6ms**	**9.7ms**
Real tool call (Claude Sonnet via Bedrock)	3,130ms	3,330ms

The MCL forward pass completes in 0.32ms, confirming the $<$1ms claim. Context encoding (9.1ms) dominates the full pipeline but occurs in the background speculation thread — not on the critical commit-decision path when the user's query arrives. At that point, only the cosine gate fires (0.04ms). The full PRESTO overhead (9.6ms) represents 0.3% of a representative LLM-based tool call (3.1s), confirming that speculation adds negligible latency to the system.

A.16 EXTENDED τ-BENCH ANALYSIS

The main body (§5.4) reports the interventional in-loop cache results (2-fold matrix, 4 arms, McNemar contrasts). This appendix provides supplementary context from the pre-interventional observational analysis.

Table 2 cluster-bootstrap CIs. 95% cluster-bootstrap CIs (by task, W=15s) — retail: P $[16.0, 21.5]$, C $[25.3, 31.5]$, A $[49.5, 55.6]$, B $[56.2, 62.8]$; airline: P $[8.0, 14.9]$, C $[10.2, 17.9]$, A $[16.6, 25.2]$, B $[19.2, 28.4]$ (all in %).

Stateless Tool Classification. Of τ-bench tool calls, 55% (retail) and 48% (airline) of turns invoke exclusively stateless tools whose output is time-invariant. The in-loop cache (§5.4) restricts speculation to these stateless tools; in the deterministic replay environment, exact-key hits return the same output as synchronous execution (deployment additionally requires a freshness policy, e.g. TTL or versioned reads).

Entity-Revelation Boundary (per-tool). Oracle ceilings (harness-aligned): retail 88.4% turn-level (907 eligible turns), airline 47.3% (317 eligible). The dominant failing args: retail — order_id (696 same-turn), zip (368), name (289); airline — reservation_id (226), date (94), user_id (87). See §5.4 for the full ceiling analysis and miss decomposition.

Grounding Gap (per-grounder detail). Oracle-free grounding (§5.5): the committed centroid maps to tool calls via kNN retrieval (G1) or a planner LLM (G2/G3), scored against the next turn's annotated `required_tools`. Tool-set *coverage* is 21.8% (G1) / 24.8% (G2, Haiku) / 18.0% (G3, Sonnet) on committed turns. Feeding the planner the *oracle* next utterance (G4) raises coverage to 54.5% (Haiku) / 47.9% (Sonnet), a +29.7pp gap from prediction error, invariant across planner classes. *Result-usable* PDR (coverage and all args grounded in context) is $\sim$5.5% for both planner classes because planner args are grounded only 50–65% of the time regardless of utterance or planner strength. Haiku-to-Sonnet improves arg groundedness (+15pp) but reduces coverage ($-$7pp) as the stronger model conservatively declines to speculate; usable PDR is unchanged.

Calibration Diagnostic. A calibration analysis using PRESTOBench's `stale_if_prefetched` labels shows that the stale rate ($\sim$4%) is approximately constant across cosine bins, indicating that rare pivot events are not cosine-dependent. This motivates the conformal gate (§5.5): since raw cosine is uninformative for conditional risk, we replace it with a learned score over entity-availability features that is informative (Table 3).

A.17 Baseline Sweep with Bars Fixed in Advance (Is Prediction Necessary?)

To test whether MCL prediction is necessary rather than dumb entity enumeration, we fixed the comparison design — selectors, budget-matching rule, and pass/fail bars — before looking at any result: a matched-budget comparison of three non-MCL selectors against the frozen Arm A/B in-loop results, on both τ-bench domains. Each selector enumerates the feasible tool calls whose arguments are already revealed in conversation history and ranks them; none trains, none uses the planner. Selectors: **dialogue-state heuristic** (rank by a next-tool transition prior estimated from the opposite fold), **frequency prior** (rank by opposite-fold next-call frequency), and **no-training state-to-call retrieval** (retrieve nearest historical states, use their next-call identities to rank). Budget is swept via a per-turn top-N cap ($N \in \{1..5\}$); the cell whose achieved prefetches/turn matches a frozen Arm A/B budget within $\pm 5\%$ is graded by per-turn McNemar (exact-binomial if $b+c<25$, else χ^2 with continuity correction) plus a task-cluster bootstrap CI. All labels (in-band/descriptive) were pinned from achieved budgets *before* hit rates were unblinded.

Table 19: Retail selector budget curve (turn-level hit rate at $W=15$s, per fold, per top-N). Frozen MCL: Arm A (0.74 pf/turn, 52.6%), Arm B (1.90, 59.4%). The graded cell is dialogue-heuristic at $N=2$ (1.81 pf/turn, in-band vs Arm B): 53.4% vs 59.1%, -5.7pp, McNemar $p=0.032$, 95% CI $[-10.2, -1.0]$pp. Selectors reach Arm B's rate only above Arm B's budget (efficiency, not an unreachable ceiling); the sub-1.0 region is empty (integer-N floor, no abstention). Parenthetical second header row: realized prefetches/turn at each N.

Selector	Fold	$N=1$ (1.0)	$N=2$ (1.8)	$N=3$ (2.4)	$N=4$ (3.0)	$N=5$ (3.7)
dialogue-heuristic	f1	47%	**53%**	57%	60%	64%
	f2	48%	52%	57%	60%	63%
frequency prior	f1	49%	56%	60%	65%	66%
	f2	50%	56%	61%	65%	67%
state-to-call NN	f1	48%	58%	61%	63%	66%
	f2	50%	57%	61%	65%	68%

Reading (pinned). On retail, non-predictive enumeration loses to zero-shot MCL at matched budget (graded above: dialogue-heuristic $N=2$, -5.7pp, McNemar $p=0.032$). The loss is robust to task-level clustering: the task-cluster bootstrap 95% CI excludes zero (Table 19 caption) and a task-clustered sign-flip permutation test gives $p=0.024$; the second fold replicates directionally ($p=0.003$) but is marginally out-of-band on achieved budget and so is reported descriptively, not graded. The per-prefetch efficiency gap underlying Arm B's volume win is 0.55 hits/prefetch for

adapted Arm A (0.74 pf/turn) versus 0.39 for zero-shot Arm B (1.90 pf/turn): B is less efficient per prefetch but reaches higher coverage on volume. The no-training retrieval baseline ties within power (-1.6pp, p=0.60), so what is shown necessary is *conditioning on dialogue state*, not training, which supports the paper's latency/gate framing rather than undermining it. On airline, MCL's operating points (Arm A 0.94–1.01, Arm B 1.23–1.28 pf/turn) fall in the gaps of the selector's integer-N budget ladder (0.75, 1.4, 2.1, ...), so no cell is in-band and the airline leg is descriptive-only; the selector curves track the same efficiency shape, milder as expected in the ceiling-limited domain. Across both domains, enumeration occupied exactly one of six MCL operating points: a gated predictor sets its budget continuously and reaches the low-budget region that integer enumeration cannot.

Volume-matched Arm A. To rule out the converse confound — that zero-shot Arm B wins only by prefetching more — we attempted to raise adapted Arm A to Arm B's 1.90 prefetches/turn by disabling centroid dedup entirely and sweeping the per-turn cap $N \in \{3, 4, 5\}$. Arm A does not reach it: budget and hit rate both saturate well below Arm B's operating point — the "adaptation over-narrowed hypothesis support" outcome we had named as the expected result in advance. Per fold at N=5: fold 1 reaches 0.557 pf/turn / 49.1% (from 0.533/48.9% at N=3), fold 2 reaches 0.920 pf/turn / 55.9% (from 0.861/54.2%); the marginal gain from N=4 to N=5 is $+0.005$/$+0.020$ pf/turn. Because no cell approaches 1.90, the arm is descriptive-only (it grades nothing). At each fold's ceiling, descriptively vs. that fold's Arm B: fold 1 -10.0pp (49.1% vs. 59.1%, McNemar p=3.5×10^{-5}, 95% CI $[-14.1, -5.8]$pp), fold 2 -3.8pp (55.9% vs. 59.7%, p=0.12, CI $[-8.3, +0.7]$pp). The "cannot be volume-matched" conclusion is fold-uniform; whether Arm A still trails at its ceiling is not. This closes the budget-artifact reading of the in-loop A<B gap (§5.4) — Arm A's coverage ceiling is structural, not a budget choice — but does not by itself establish equal per-hypothesis quality.

Net-benefit accounting (τ-bench, measured). Net benefit per turn = (latency saved)/turn $-$ (pf/turn) $\cdot$ c_{compute}, using measured saved-latency and achieved budget over a single denominator (all turns), at W=15s. There is no stale-content term: exact-key matching makes a miss a wasted prefetch, not a wrong served result. Break-even c^* (seconds saved per prefetch issued, the exact zero-crossing): retail Arm A c^*=0.82 (net-positive to 0.82s/prefetch of background compute), Arm B c^*=0.41, A′ (coverage-WTA) 0.41; airline Arm A 0.38, Arm B 0.35, A′ 0.46. The split is low-budget-vs-rest, not retail-vs-airline: Arm B's 1.90 pf/turn buys $+6.8$pp accuracy but only $+0.18$s/turn more saved latency at 2.6× the volume, so its per-prefetch yield halves relative to Arm A. Coverage-WTA matches Arm B's per-prefetch efficiency on retail (c^*=0.41 at 2.17 pf/turn) and is the most cost-efficient airline arm (c^*=0.46 at 1.39 pf/turn) — its accuracy win carries no per-prefetch cost penalty. At any realistic background-compute cost (<0.4s/prefetch for a local forward pass), all arms are net-positive; the accounting simply shows the efficient operating point is the cost-robust one.

A.18 Coverage-WTA: In-Loop-Supervised Adaptation

Method. Coverage-WTA modifies the winner-take-all assignment (Eq. 3) only: instead of the nearest-embedding centroid, the winning head for a training pair is the one whose grounded, canonicalized tool call yields a realized exact-key cache hit against the actual next-turn calls. Hit labels are self-generated by the exact-key cache at runtime (no new supervision), computed on the training fold only (leakage boundary: fold 1 labels from tasks [230:460], fold 2 from [0:230]). Winners are assigned in *call space* — distinct canonical calls distributed across the K=5 heads — so coverage comes across pairs rather than relying on within-pair multi-hit multiplicity (thin: $\sim$10% of labeled pairs). Pairs with no hit-yielding label ($\approx$82% — the same-turn-entity ceiling dominates) are masked from the loss, not substituted. On retail, label generation yields 310/287 hit-yielding pairs (fold 1/2) over 118/104 distinct calls, versus the $\sim$8 effective calls the proxy-supervised arm collapses to.

In-loop result (retail, W=15s). Arm A′ reaches 68.9/71.2% turn-level hits, vs. zero-shot B 59.1/59.7% (pooled McNemar p=2.7×10^{-11}) and no-training retrieval at over-matched budget (+9.7/+14.1% more prefetches/turn) 60.7/61.4% (+8.2/+9.8pp, pooled p=1.3×10^{-8}). Not a volume artifact: A′ runs at equal planner budget and *higher* per-prefetch precision than B (0.41/0.38 vs. 0.39/0.38) — so the gain is coverage, and retrieval loses even when given more prefetches/turn, not a byproduct of A′ prefetching more. The retrieval win is significant in *both* retail folds. On airline, A′ beats over-matched retrieval by +4–11pp (pooled p=4.3×10^{-4}; fold 1 p=5×10^{-4}, fold 2 directional but per-fold n.s. at n=100 — the same underpowered-single-fold airline pattern the

backbone swaps carry). Measured against the predictor-addressable gap (ceiling − B), coverage-WTA closes a near-identical fraction in both domains: retail $(70.1-59.4)/(88.4-59.4)=36.9\%$ and airline $(32.2-23.7)/(47.3-23.7)=36.0\%$. We note this cross-domain regularity as an observed post-hoc pattern over two domains, not a tested law.

Gate ε-grid and validity (both fold-transfer directions). Table 3 reports the certified $\varepsilon=0.10$ operating point, where all four rows (both scores, both domains) are LTT-valid. Away from it, the full sweep records the certified-level exceptions honestly, all traceable to finite-sample test risk rather than a broken procedure. On retail: the *raw* score violates at $\varepsilon=0.15$ direction 1 (test risk 0.162) and at $\varepsilon=0.30$ direction 1 (0.306); the *learned* score violates only at the tightest $\varepsilon=0.05$ direction 1 (0.051, mirror direction valid at 0.042). On airline (calib $n\approx 81$/direction), at $\varepsilon=0.05$ LTT *abstains* for both scores in direction 2 (no threshold achieves the Clopper-Pearson UCB at that n) — the certificate declining rather than over-promising; direction 1 certifies and holds. The raw score also violates at $\varepsilon=0.40$ direction 1 (0.429); the learned score is valid at every *certified* level in both airline directions. The single-split exceedances at tight ε and small n are expected fluctuation and are reported as-is, not harmonized. Feature auto-selection differs by direction (G0+G3 vs. the full G0–G3 set) and is reported per-direction; test folds are $n=77$ (airline), so fired-precision point estimates carry wide intervals.

Epoch-robustness (fixed-final selection). A′ is selected at a fixed final epoch chosen in advance (proxy-free; offline commit rate is *not* used for selection, as it is the metric the proxy reversal indicts). Offline commit peaks early then declines (retail fold 1: 72.7% at epoch 20 → 64.0% at epoch 100; airline peaks by epoch 10–20 in both folds), so we report the in-loop hit rate at epochs $\{20, 50, 100\}$ as a robustness row. Retail pooled: 66.9/70.1/70.1% (fold 1 $66.7/68.5/68.9$; fold 2 $67.2/71.6/71.2$); airline pooled: 30.9/33.1/32.2% (fold 1 $31.6/34.8/34.2$; fold 2 $30.2/31.4/30.2$). Every checkpoint in both domains beats zero-shot B (59.4%/23.7% pooled): the in-loop win is not a checkpoint artifact. On retail, epoch 20 —where the offline proxy peaks—is the in-loop *worst* of the three in both folds, the proxy reversal recurring at checkpoint granularity. All per-epoch contrasts are descriptive (no per-epoch test); epoch 100 is the pinned primary throughout (on airline, epoch 50 is descriptively $+0.9$pp higher, reported as-is—selection stays fixed-final).

Backbone-robustness: is the win the objective or the architecture? Coverage-WTA is a change to the training *objective* (winner assignment by realized cache hits), not the predictor architecture. To test whether the in-loop win is a property of the objective rather than the specific $768\to128\to128$ MLP, we hold the objective, labels (identical call-space `win_idx`), folds, $K=5$, and fixed-final selection fixed, and swap only the encoder — retraining both A′ and zero-shot B and re-running the full in-loop harness (W=15s) for each. Two swaps: a *capacity* variant (MLP hidden width $128\to256$, 1,315,328 params) and an *architecture-class* variant (a matched-capacity self-attention encoder over the last $N=3$ per-turn embeddings, 676,608 params, 4 heads; same information as the MLP's joined context, different aggregation). In every case A′ beats B in *both* folds of *both* domains, pooled McNemar significant (Table 20): capacity retail $p=7.6\times10^{-32}$, airline $p=4.1\times10^{-5}$; attention retail $p=2.3\times10^{-37}$, airline $p=1.8\times10^{-6}$ (attention airline fold 1 directional but per-fold n.s., $p=0.24$ at $n=158$ — the same underpowered-single-fold pattern the airline retrieval cell carries, reported as-is). The adapted predictor's *absolute* level is backbone-stable: airline is within 0.5σ under the capacity swap (30.9%) and reproduces the banked 32.2% exactly under the attention swap (102 hits), and retail is within 2σ under both (70.1% banked vs. 71.1%/72.8% capacity/attention) — we claim stability, not improvement. Zero-shot B lands measurably below its banked level under both swaps (retail 54.1/54.2% vs. banked 59.4%, $z\approx-3.2$; airline within noise, -1.6pp, $z\approx-0.7$), so the *margin* widens on retail purely because B drops, not because A′ rises — the like-for-like margin comparison is airline ($+8.5$ banked vs. $+8.8$/$+10.1$pp). Notably, the two independent non-banked B retrainings land within 0.1pp of each other in both domains (54.1/54.2 retail, 22.1/22.1 airline) while A′ holds its level under the fixed-final rule — consistent with B's proxy-selected checkpoint carrying more run-to-run variance, a small point in favor of proxy-free selection. The win is thus a property of the objective, robust across a wider-MLP capacity change and an MLP→attention architecture-class change.

Table 20: Backbone-robustness of the coverage-WTA win (in-loop turn-hit %, W=15s, pooled over folds; McNemar A′ vs. B). The objective and labels are held fixed; only the encoder changes. A′ beats B in both folds of every backbone; airline A′ reproduces its banked absolute level exactly under the attention swap and is within 0.5σ under the capacity swap.

Domain	Backbone (params)	B	A′	McNemar p
Retail	MLP 128 (627K, banked)	59.4	70.1	2.7×10^{-11}
	MLP 256 (1.32M)	54.1	71.1	7.6×10^{-32}
	Attention (677K)	54.2	72.8	2.3×10^{-37}
Airline	MLP 128 (627K, banked)	23.7	32.2	4.2×10^{-5}
	MLP 256 (1.32M)	22.1	30.9	4.1×10^{-5}
	Attention (677K)	22.1	32.2	1.8×10^{-6}

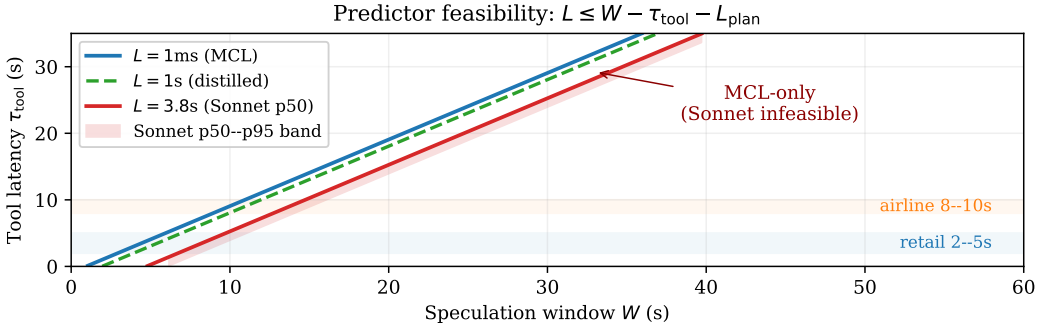


Figure 3: Predictor feasibility over window W and tool latency τ_{tool} (feasible below each boundary; L_{plan} p50 0.94s, predictor p50 with p95 band). Contours: L=1ms (MCL), L=1s (distilled), L=3.8s (Sonnet p50). The τ-bench domains (retail 2–5s, airline 8–10s) sit at the light-tool end, so the MCL-only region grows with heavier tools (PRESTOBench 3–35s); MCL is feasible almost anywhere a prefetch can complete, slower predictors only at high slack (large W, light tools).

A.19 Sonnet-as-Predictor In-Loop: Full Budget Sweep

Within our two τ-bench domains tools are light (2–10s); Sonnet (L_{pred}+L_{plan}≈4.7s p50) needs $W \gtrsim$12.7–14.7s to clear the 8–10s airline tools, so it is infeasible on airline at the 238-wpm median (11s window) and marginal at the slow-reader median (15s: 0.3–2.3s slack), clearing both domains only from $W \gtrsim$15s upward. The W=5s cells demonstrate the low-slack regime's existence and mechanism, whose practical prevalence comes from the tool-latency axis (heavy or chained tools), not from fast readers. Figure 3 maps the feasibility region over (W, τ_{tool}) for each predictor-latency contour.

Full 12-cell sweep behind Table 21 (Sonnet-diverse-forced as the in-loop predictor; both domains, both folds, per-turn cap N swept; W=15s turn-hit rate and achieved prefetches/turn). No cell reaches Arm B's budget band (retail [1.805, 1.995]; airline [1.17, 1.34]), so by the rule we fixed in advance the arm is descriptive-only (no matched-budget grade fires); W=5s is 0% in every cell: predictor latency (L_{pred} p50 3.8s) plus planning (L_{plan} p50 0.94s) consumes effectively the entire window ($\le$0.3s of slack against tools of $\ge$3s), so no prefetch completes in time. Where slack is ample (e.g. W=15s), Sonnet reaches 64–70% on retail across cells — at lower budget than Arm B's 59.1–59.7% — confirming the high-slack regime.

Deployment cost. The latency moat has a monetary twin. A diverse-forced Sonnet predictor issues K=5 API calls per turn; our measured baseline (Table 4, 4,550 calls over 910 turns at ∼60 output tokens each) cost ∼$15, i.e. ∼$16.5 per 1,000 turns. The MCL is a 0.32ms CPU forward pass (Table 18); at commodity CPU rates (∼$0.19/instance-hour) its per-turn cost rounds to zero ($<\$10^{-4}$ per 1,000 turns) — a $\sim 10^6\times$ gap dominated by the frontier API price, not the hardware. Figures are illustrative at list prices (AWS Bedrock / EC2, as of 2026-08) and scale linearly with K and traffic;

Table 21: Selected cells (per fold, W=15s; budgets NOT matched to Arm B, so descriptive — no significance test). Cell per fold = closest achieved budget to Arm B's band (max-N in all folds; a budget-only rule, the variable inspectable pre-outcome). W=5s MCL comparator is Arm B.

Domain	Fold	Sonnet (pf/turn, $W15$)	Arm B (pf/turn, $W15$)	$W5$: Son./MCL
Retail	f1	1.29, 66.2%	1.90, 59.1%	0% / 34.5%
	f2	1.36, 69.5%	1.90, 59.7%	0% / 32.2%
Airline	f1	0.99, 29.1%	1.25, 24.7%	0% / —
	f2	1.04, 30.2%	1.25, 22.6%	0% / —

Table 22: Sonnet-as-predictor budget sweep (achieved pf/turn, W=15s turn-hit). Bold = the max-N cell selected for Table 21.

Domain	Fold	N	pf/turn	W=15s hit
Retail	f1	3	1.180	64.4%
	f1	4	1.269	67.8%
	f1	5	**1.288**	**66.2%**
	f2	3	1.224	65.5%
	f2	4	1.295	66.1%
	f2	5	**1.363**	**69.5%**
Airline	f1	2	0.826	28.5%
	f1	3	0.874	27.8%
	f1	4	**0.991**	**29.1%**
	f2	2	0.809	24.5%
	f2	3	0.957	28.3%
	f2	4	**1.043**	**30.2%**

the qualitative point — a per-turn frontier-call budget is prohibitive at production traffic while the small model is effectively free — is price-list-independent. This is the honest form of the moat: MCL wins not only where prediction-time slack is short (heavy or chained tools) but wherever five frontier calls per user turn is uneconomic, which is most production settings.

A.20 Entity-Revelation Boundary Transfer to MultiWOZ

To test whether the entity-revelation-boundary diagnostic (§5.4) is a property of dialogue rather than an artifact of LLM-simulated τ-bench users, we compute the same ceiling arithmetic on MultiWOZ 2.2 (Zang et al., 2020) (10,437 human-authored, human-collected dialogues, all splits pooled). This measures *ceiling arithmetic only* (the fraction of eligible turns whose tool-call arguments are pre-revealed in prior history), not in-loop hit rate, latency, or cost; no model runs. We construct a hypothetical read-only lookup tool per domain from the schema's find-intent slots (excluding booking-side and requestable-output slots), and ask whether the values a lookup would need were revealed before the triggering turn. We report two denominators: *update-turns* (a new argument value is introduced; primary) and *active-turns* (any usable domain arguments; comparable to τ-bench semantics).

Table 23: MultiWOZ per-domain entity-revelation ceiling (all splits, v2.2). same-turn-revealed % $= 100-$ ceiling %, derivable. bus excluded (n<20). Under the update denominator the level is structurally lower (each eligible turn introduces a fresh value); only the cross-domain *spread* is comparable to τ-bench, not the level.

Domain	Update ceiling (n)	Active ceiling (n)
train	7.5% (8,772)	62.6% (23,768)
restaurant	13.3% (7,523)	71.0% (24,697)
attraction	14.7% (4,039)	74.7% (17,329)
hospital	7.8% (103)	71.6% (335)
hotel	20.9% (5,739)	77.0% (22,125)

The active-turn ceiling spans 62.6% (train) to 77.0% (hotel), a 14.4pp spread whose extremes are preserved under the update denominator (train lowest, hotel highest in both; restaurant and hospital swap mid-order). Scheduling-style domains (train) rank lowest, mirroring τ-bench airline; browse-and-select domains (hotel, attraction) rank higher, mirroring retail, consistent with higher same-turn entity density in scheduling queries, though we measure ceilings, not entity density directly. The τ-bench spread (41pp) is larger; we claim the diagnostic's discrimination transfers and the domain ordering is consistent, never a magnitude match. Caveats: values are matched by substring (may slightly overcount revelation); coreferent/implicit values count as not-revealed (conservative, ceiling is a lower bound); boolean, `dontcare`, and length-≤ 2 values are excluded (17.2% of candidate values); the tool schema is hypothetical (MultiWOZ has no tool API).

A.21 Is Coverage Supervision's Advantage Universal? (Proxy-WTA Control)

Coverage-WTA's in-loop win over *proxy*-supervised adaptation (Arm A) on τ-bench (§5.4) could be a general property of training on the deployed objective, or specific to τ-bench's planner-in-loop failure mode. We test it on human dialogue: on each MultiWOZ domain we train Arm A (stock winner-take-all, nearest-embedding winner) on the *identical* label artifact as A′ — only the training-time winner rule differs (Arm A ignores the stored winner index and recomputes nearest-embedding winners) — and run the three-arm in-loop eval (B, A′, A) under deterministic grounding. The two winner rules are statistically independent: the coverage hash-partition target and Arm A's trained nearest-centroid winner coincide on $19.4\%/21.2\%/20.1\%$ of labeled training pairs (restaurant/hotel/train), matching the independence expectation given each rule's marginal head distribution ($19.4\%/20.9\%/20.0\%$). The A-vs-A′ null is therefore a dissociation between genuinely different supervision signals, not two equivalent objectives.

	A′-vs-B update	A-vs-A′ update	A-vs-B update	Active A′-vs-B
Restaurant	**15.3** vs 2.1%, p=1.2×10^{-14}	p=0.34 (null)	61/8, p=3×10^{-11}	tie, p=0.48
Hotel	**19.6** vs 13.0%, p=0.016	p=0.16 (null)	51/40, p=0.29 (null)	wash, p=0.50
Train	**2.0** vs 0.2%, p=4×10^{-4}	p=0.42 (null)	13/2, p=0.007	**B**>**A**′, p=1.1×10^{-19}

Table 24: Three-arm MultiWOZ transfer. The constraint-changing-turn contrast (A′-vs-B update, fixed before grading as the transfer criterion) is significant in every domain; coverage supervision buys no significant margin over proxy supervision (A-vs-A′ update null, 3/3).

Two structural facts hold across all three domains, and one contrast varies. (i) The constraint-changing-turn contrast — coverage-WTA over zero-shot — is significant everywhere, even at train's near-floor headroom. (ii) Yet A′ does *not* significantly beat A on update turns in any domain (McNemar null, 3/3). Arm A itself numerically exceeds zero-shot B on update turns everywhere, but significantly only on restaurant (61/8, p=3×10^{-11}) and train (13/2, p=0.007), not hotel (51/40, p=0.29) — on hotel only coverage-supervised A′ clears zero-shot. Trained prefetching thus helps under either supervision where the effect is significant, but coverage supervision buys no reliable margin *over* proxy supervision.

The mechanism is visible in the winner-head distribution (argmax cosine of the K centroids to the realized target). Coverage supervision does what it is designed to do — it is the only arm that maintains update-turn head diversity (proxy concentrates 58–71% of winner mass on a single head; coverage stays $\leq 27\%$) — yet this diversity confers no hit-rate margin under deterministic grounding (A-vs-A′ null, 3/3). The τ-bench supervision margin is planner-specific: winner-head concentration only costs hits where a dedup step prunes the candidate set (§5.4); since the hit rule counts all K grounded keys, a concentrated winner distribution need not imply a redundant candidate set. Plain WTA also retains the no-change coverage A′ trades away, dominating A′ on no-change turns in every domain (restaurant 54/0, hotel 35/0, train 118/0); on the active denominator this makes A significantly exceed A′ on restaurant and train, though not hotel (p=0.10). (Zero-shot B, by contrast, places 70–72% of active and $\sim$90% of update winner mass on one head identically across all three domains — a fixed zero-shot geometry consistent with its weak update-turn rates.) The proxy reversal is thus a property of the planner-in-loop failure mode, not a universal training effect — a scoping of the §5.4 claim, consistent with its stated mechanism.

Every turn that A′ hits and zero-shot misses is constraint-changing: the $69/60$ active discordance decomposes into $69/6$ on update turns and $0/54$ on no-change turns. This exactness (the 0 in $0/54$: A′ never uniquely wins a no-change turn) is restaurant-specific — it does not hold on train, where the coverage trade is net-negative on the active denominator (above).

Train key-construction disclosure. On train, the time slots `arriveby`/`leaveat` are keyed on raw surface sets (no closed schema vocabulary exists) and the annotation sentinel `unknown` is retained as a literal value ($\times 12$ departure, $\times 11$ destination); both are keyed identically across all arms, so they affect the level uniformly and not the between-arm contrast.

A.22 Human Annotation Study

To ground the benchmark's turn-level structure in human judgment rather than author assertion, we ran a crowd annotation study on Amazon Mechanical Turk. For each user turn after the first, three independent workers answered two questions: *is this a topic pivot?* and *is this turn predictable from the prior context?* We collected 3 ratings each on 419 scenarios (239 of the 300 hand-crafted primary scenarios — the 61 remaining are almost entirely the single-turn vague-intent, clarify-vs-proceed, and multi-tool categories, which have no turn ≥ 2 to annotate — plus 180 of the generated scenarios), 1,257 assignments from 783 distinct workers.

Humans confirm the pivot/predictability structure. Inter-annotator agreement is moderate: Fleiss' κ=0.46 (pivot) and 0.60 (predictable), with 80% raw pairwise agreement on pivots. Two internal checks pass loudly. *Construct validity:* the human majority pivot rate tracks the designed scenario families — 86.9% of topic-pivot scenarios carry a human-marked pivot, versus a 5.6% turn-level pivot rate on the designed pivot-free latency probes — so workers were reading, not clicking. *Premise confirmation:* humans judge a non-pivot turn predictable 99.9% of the time versus 44.7% on pivot turns, an independent human endorsement of the paper's premise that non-pivot continuations are predictable and pivots are exactly where prediction should break.

Human vs. reference labels, both directions. The comparison spans all 419 study scenarios: the hand-crafted primary scenarios carry author-assigned labels and the generated scenarios carry model-assigned labels (both §A.4), so this contrasts crowd judgments against the benchmark's designer and model labels. Against the human majority vote (strict rule, ambiguous double-votes dropped; N=1,818 turns): humans confirm 82% of designated pivots (recall $83/101$), and raw agreement is 80.5%. But Cohen's κ=0.25: the disagreement is one-directional — humans flag additional soft topic-shifts the stricter reference labels do not mark (337 human-only pivots vs. 18 reference-only). These extra flags concentrate in the long, free-form conversations (a 14–31$\%$ turn-level pivot rate across the free-form multi-turn families) and are absent from the tightly-designed pivot and soft-pivot probes (no human-only pivots), consistent with workers applying a broad lay notion of "subject change" to open-ended dialogue rather than the benchmark's narrower prefetch-relevant pivot. We report both numbers: the designated pivots — the only labels the pivot-PDR conditions on — carry meaningful human endorsement, while the *completeness* of the pivot set is not claimed, which is exactly why pivot-PDR is treated as a descriptive diagnostic (§5.5) rather than a load-bearing metric.

Compensation. Workers were paid \$0.75 per assignment (median completion time $\sim$10 minutes); total spend including platform fees was $\sim$\$1,130. The `stale_if_prefetched` label was not part of the study.