Detect, Enhance or Defer: Online Schema Enhancement for Task-Oriented Dialogue

Anonymous ACL submission

Abstract

Task-oriented dialogue (TOD) systems are deployed against fixed schemas, but real users often exceed them by asking about concepts the schema cannot represent, or requesting values outside enumerated spaces. We introduce *schema enhancement*: given a dialogue turn and a deployed schema, decide whether the schema is inadequate and, if so, emit a minimal typed edit; otherwise emit DEFER. We release a 20-cell human-reviewed benchmark drawn from four TOD corpora and seven typed edit kinds (24,340 test samples), and a detect-then-enhance pipeline: a small schema-issue detector gates a larger enhancement LM that uses dual-memory retrieval and a memory-score safety gate. Across four enhancement LMs from 32B to frontier scale, retrieval plus the gate raise Semantic@5 recall by 14–22 absolute points over a matched direct-LLM baseline; the gate alone drives false concrete edits to zero on the closed-source models we tested while keeping 94% of the dual-memory recall gain. Code and data are at https://anonymous.4open.science/r/live-tod-C0DC.

1 Introduction

Task-oriented dialogue (TOD) systems mediate users and downstream services through a typed *schema*: intents, slots, and value spaces that turn free-form language into structured API calls (Rastogi et al., 2020; Ye et al., 2022). An adequate schema lets even a simple state tracker succeed (Lee et al., 2021; Zhao et al., 2022; Li et al., 2024); if the schema is wrong, no amount of fluency or tracking accuracy fixes it. Schemas, however, are static; user needs are not. Customers ask about concepts the schema does not encode, request values outside enumerated spaces, or express preferences at finer granularity than supported slots. This sensitivity is well documented for current trackers (Lee et al., 2022; Richardson et al., 2024; Safa and Şahin, 2025). Figure 1 shows one such turn: the schema, not the tracker, is the source of the failure.

Existing TOD methods treat the schema as given. *Dialogue state tracking* (DST) conditions on a fixed schema and fills it (Cao and Zhang, 2021; Zhao et al., 2022; Finch and Choi, 2024; Xu et al., 2025); the most a tracker can do with the turn in Figure 1 is populate `daily_limit` and drop the rest. *Slot schema induction* (Yu et al., 2022; Finch et al., 2024, 2025) mines new slots offline from dialogue corpora, but its output is a flat additive inventory: it cannot replace, narrow, split, or merge existing schema elements, it has no per-turn binding to dialogue evidence, and it has no abstention. Neither paradigm decides when the schema should change, or how to tell that the change is safe.

We frame *schema enhancement* as a structured per-turn task. Given a dialogue turn d and the deployed schema S, a schema-enhancement system predicts whether S is inadequate and, if so, emits a minimal typed edit (LEAFADD, LEAFREPLACE, VALUEEXTEND, TYPENARROW, GRANULARITYSPLIT, or GRANULARITYMERGE) over S; otherwise it emits DEFER. DEFER is a first-class output, not a fallback, so abstention is itself an evaluable prediction. We build (i) a benchmark of 20 schema-domain cells drawn from four TOD corpora (MultiWOZ 2.4 (Ye et al., 2022), SGD (Rastogi et al., 2020), Taskmaster (Byrne et al., 2019), and STARv2 (Mosig et al., 2020)) with seven typed edit kinds and four families of negatives, totaling 24,340 human-reviewed test samples; and (ii) a two-stage *detect-then-enhance* system: a small fine-tuned schema-issue detector that jointly performs DST and gap detection, followed by a larger enhancement LM that operates under a typed grammar with dual-memory retrieval (Lewis et al., 2020) and a memory-score safety gate. The gate is a hard rule on the memory score μ: a concrete edit is admitted only if its $(kind, target)$ pair has support in the in-cell memory bank.

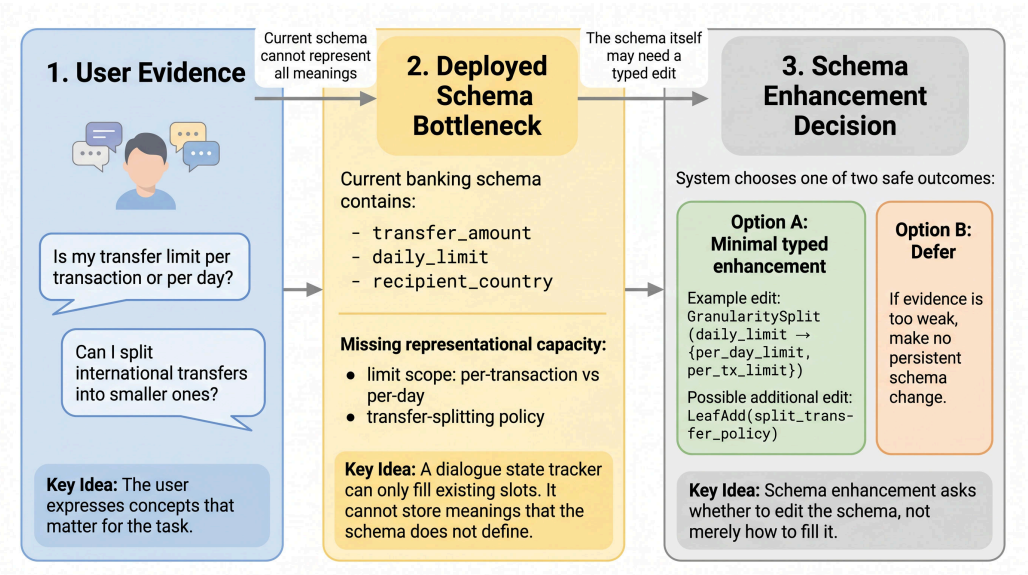

Figure 1: When a user turn exceeds the deployed schema, schema enhancement emits a minimal typed edit, or DEFER.

Contributions. We introduce **(i) the schema-enhancement task** with an explicit DEFER action and dual recall/safety evaluation; **(ii) a benchmark** of 20 schema-domain cells from four TOD corpora (24,340 human-reviewed test samples, 7 edit kinds, 4 negative families) with audit-trail provenance; and **(iii) a detect-then-enhance pipeline** pairing a small schema-issue detector (joint with DST) with a larger enhancement LM that uses dual-memory retrieval and a memory-score safety gate. Across four enhancement LMs from 32B to frontier scale, the combined pipeline raises schema-edit recall over a matched direct-LLM baseline; the gate alone drives detector-triggered false concrete edits to zero on the closed-source models we tested and reduces them substantially on open weights.

2 Schema Enhancement as a Task

Schema enhancement is a structured prediction problem with a typed output space and an explicit DEFER action, scored jointly for recall and safety. Every proposed change is typed and applied by a deterministic validator; DEFER is always available as the alternative.

2.1 Problem formulation

Let S be a deployed service schema (a typed structure of intents, slots, value spaces, and granularities) and d a dialogue turn. An enhancement system observes (d, S) and returns

$$y \in \{\text{DEFER}\} \cup \{(k, t, p) : k \in \mathcal{K}, \; t \in T(S), \; p \in P(k)\},$$

where $\mathcal{K}$ is a fixed set of edit kinds, $T(S)$ the schema-valid targets, and $P(k)$ the kind-specific payload space. DEFER is part of the output space, not a fallback: it asserts that evidence is insufficient to commit a persistent change. A deterministic validator rejects nonexistent targets, malformed payloads, and type-incompatible edits before evaluation.

2.2 Edit grammar

We motivated the seven kinds by auditing dialogue failure modes in the four source corpora (Appendix C), grouped by what must change in the schema artifact rather than surface form: LEAFADD (a missing slot or leaf concept), LEAFREPLACE (an obsolete leaf), VALUEEXTEND (a value outside the enum), TYPENARROW (an overly broad type), GRANULARITYSPLIT (a coarse concept needs splitting), GRANULARITYMERGE (over-fragmented concepts need a parent), and DEFER (no persistent edit). These seven cover every failure our audit found; the grammar extends to other kinds (e.g. for schemas with referential constraints) without changing the safety gate. Per-kind results are in Section 5.

2.3 Evaluation metrics

A good enhancement system finds the right edits *and* refuses to commit when none is appropriate. We score both objectives at two granularities, parameterized by a top-k cut-off. Call a top-k candidate *concrete* if it is not DEFER and passes the deterministic validator.

Detector-triggered metrics judge the enhancement stage in isolation, on the samples where the schema-issue detector triggers. **Sem@k** (primary; higher is better) averages, over true-positive triggers, the cross-encoder similarity between the gold edit and the best of the top-k candidates. **FC@k** (safety; lower is better) is the fraction of false-positive triggers whose top-k contains a concrete candidate—each such candidate is an unsupported commit.

All-stage metrics use the same numerators but denominators over the full test set, so detector misses and correct deferrals count. **A-Sem@k** divides Sem@k's numerator by *all* gold positives, charging detector misses as zero. **A-FC@k** divides FC@k's numerator by *all* gold-adequate samples, crediting correct deferrals as zero. A-Sem@k is therefore upper-bounded by detector recall and A-FC@k by its false-positive rate, attributing all-stage performance to the detector and to the enhancement stage separately.

A secondary strict **Exact@k** match modulo a deterministic canonicalizer is in Appendix I. Unless noted, we report all metrics at $k = 5$.

3 The Schema Enhancement Benchmark

We want a benchmark where (i) gold edits are deterministic and checkable, (ii) there are also cases where no edit should be made, and (iii) the dialogue surface is natural enough to support deployment generalization. Prior TOD benchmarks satisfy at most two of the three. We get all three by inverting data collection: plan the schema edit first, then use an LM only for surface realization, under deterministic supervision.

3.1 Substrate: dialogue corpora and schema-domain cells

We did not assemble dialogues from scratch. Authoring tens of thousands of TOD turns directly with an LM would (i) drift from real user–service distributions, (ii) lose comparability with the DST and slot-induction work we contrast with, and (iii) force us to design four service ontologies in parallel with the edit grammar. Instead we sample seeds from four existing TOD corpora with vetted typed schemas and human-authored surface: **MultiWOZ 2.4** (Ye et al., 2022), **SGD** (Rastogi et al., 2020), **Taskmaster** (Byrne et al., 2019), and **STARv2** (Mosig et al., 2020)—together covering transactional, informational, and deliberative service types and the same corpora prior DST and induction work uses.

We partition the substrate into 20 *schema-domain cells*—one $(\textit{dataset}, \textit{domain})$ pair each, e.g. `sgd.flights` or `starv2.bank` (Appendix D). Each cell has a single pre-edit schema, so a planned edit validates against *that* schema rather than a corpus-wide super-schema; cells scope dual-memory retrieval banks per service domain to avoid cross-domain leakage; and train/eval/test splits hold cells fixed but split dialogue IDs disjointly, cleanly separating within-cell from cross-cell generalization (Appendix D).

3.2 Edit-first construction

Most TOD data is collected bottom-up (dialogues mined first, labels attached after), which is incompatible with a typed-edit task: the gold edit cannot be recovered unambiguously from a transcript—the same user turn can motivate several distinct edits. We invert the pipeline. Humans plan the typed edit first; an LM realizes a dialogue surface evoking it; deterministic validation verifies admissibility. Humans control *what* the gold edit is; the LM controls only *how* a user phrases a turn requiring it.

Construction (Figure 2): we audit cells for slot/value/turn coverage, then write an edit menu of $(\textit{kind}, \textit{target}, \textit{payload-template})$ tuples covering all seven kinds. We sample split-disjoint seed turns from train/eval/test; an LLM realizes each seed as a natural user turn evoking the planned edit with a candidate evidence span (the LM does not see or write the gold edit). Deterministic validation checks that the evidence span is a dialogue substring, the planned edit applies cleanly to the pre-edit schema, and the post-edit schema is type-consistent. Train and eval samples are released without human review; test samples pass through human review before entering the gold JSONL.

3.3 Four negative families

To diagnose over-commit behavior we include four families of gold-adequate negatives, each a template-generated population over the relevant

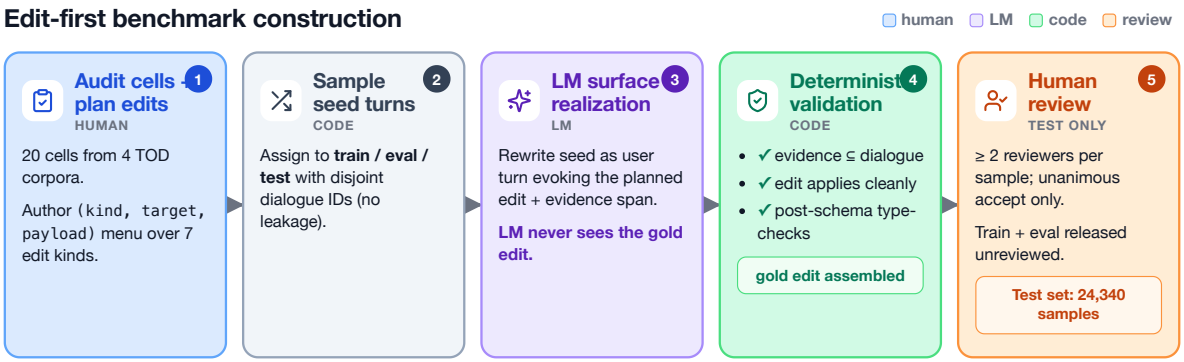

Figure 2: Edit-first benchmark construction. Humans plan typed edits; an LM realizes the surface; code validates and labels; test samples pass a unanimous human-review gate.

cells, covering one over-commit pressure:

- **Paired adequate**: novel dialogue surface for the cell but the schema already represents the user's request—tests whether a system reads surface novelty as schema inadequacy.
- **Cosmetic description tweak**: a slot description is rephrased without any structural gap—tests whether a system mistakes documentation drift for an edit signal.
- **Cosmetic enum extension**: the dialogue mentions an enum value the schema already accepts under a synonym—tests whether a system distinguishes representational gaps from lexical mismatch.
- **Cosmetic irrelevant slot**: the dialogue mentions a schema slot unrelated to the user's actual request—tests whether a system stays focused on the enhancement signal.

Our four-corpus audit produced these four over-commit modes; richer schema hierarchies may add families without changing the rest of the benchmark. Safety on the four is measured by FC@k (Section 2.3).

3.4 Human review and statistics

Test candidates were oversampled $1.5\times$ and routed through $\geq$ 2 independent reviewers per sample; only unanimously-accepted samples entered the released test set. Across the pool, 94.1% were accepted by every reviewer, 4.3% rejected by every reviewer, and 1.6% excluded for disagreement. Train and eval samples are not human-reviewed; they support memory-bank construction and detector fine-tuning, not evaluation. Table 1 reports final sizes.

Item	Count
Schema-domain cells	20
MultiWOZ 2.4	2
SGD	6
Taskmaster	6
STARv2	6
Edit kinds (concrete)	6
Defer action	1
Negative families	4
Train samples	15,841
Eval samples	8,172
Test samples (gold, human-reviewed)	24,340
Gold-positive test samples	12,099
Gold-adequate test samples	12,241
Test acceptance (all reviewers agree)	94.1%
Test rejection (all reviewers agree)	4.3%
Test excluded (reviewer disagreement)	1.6%

Table 1: Benchmark statistics. The test split is balanced between gold-positive samples (a typed schema edit is the correct action) and gold-adequate samples distributed across four negative families. Only the test set is human-reviewed; train and eval samples are released unreviewed.

4 A Detect-Then-Enhance Pipeline

Schema enhancement is more expensive than DST: a proposal must reason over the current schema, retrieved memories, and a typed grammar, at LM scales impractical to invoke on every turn. We split the system into two stages and use the first as a gate: a small fine-tuned detector produces dialogue state and a schema-adequacy signal in one forward pass; a larger enhancement LM is invoked only when the detector flags inadequacy ($\approx$50% of turns in our balanced benchmark; much rarer in real deployments). A pipeline that calls a frontier LLM on every turn is unlikely to be deployed. Figure 3 gives the overview.

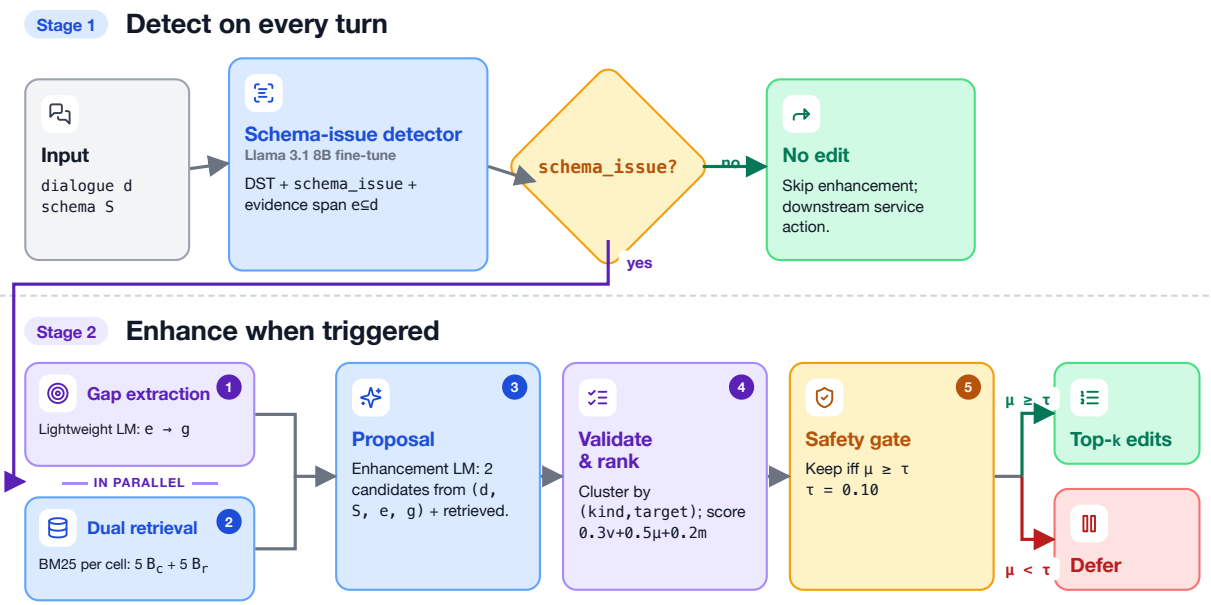

Figure 3: The detect-then-enhance pipeline. **Stage 1**: a small fine-tuned detector triggers on every turn. **Stage 2**: when triggered, CASE retrieves, proposes, validates, and applies a memory-score safety gate.

4.1 Stage 1: Schema-issue detector

The detector is a small instruction-tuned LM fine-tuned to emit, in one output, both the dialogue state and a binary schema-adequacy prediction with an evidence span $e \subseteq d$ pointing at the substring motivating the inadequacy claim. It does not output edit kind, target, or payload—those go to the enhancement stage. The binary output preserves the deployment contract of a normal DST tracker; the evidence span grounds inadequacy in user language and anchors downstream typed enumeration; and joint training adds the schema-issue head without a separate model or DST regression (Section 5.2, Appendix F).

4.2 Stage 2: Contrastive Abductive Schema Enhancement (CASE)

When the detector flags an issue, the enhancement stage receives (d, S, e) but no gold information. We treat inference as constrained abductive search: given a turn the current schema cannot represent, infer the minimal typed edit that makes it representable. The hypothesis space is the grammar of Section 2.1; *contrastive* refers to adequate-memory examples that narrow it by showing where the schema is already adequate.

Dual memory. The *contrast bank* $\mathcal{B}_c$ contains (d, S) pairs where the detector predicts adequacy and the schema is unchanged (*when should user behavior not change the schema?*). The *enhancement bank* $\mathcal{B}_r$ contains $(d, S, \text{enhancement pattern})$ triples from gold-positive training samples (*how have real gaps been enhanced before?*). Both use per-cell BM25. The two play different roles: contrast suppresses over-eager edits on novel-sounding but representable dialogue, while enhancement memory supplies canonical patterns and accounts for most of the Semantic@k gain.

Scoring. For each candidate cluster we compute proposal vote count v, enhancement-memory sup-

Algorithm: CASE per input (d, S, e)
Input: d, S, evidence span $e \subseteq d$; contrast bank $\mathcal{B}_c$; enhancement memory $\mathcal{B}_r$; safety threshold τ. **Output:** ranked top-k candidates, each a typed edit or DEFER.
1. **Gap extraction.** A lightweight LM call rewrites e into a self-contained gap phrase $g \subseteq d$ naming what the schema fails to represent. 2. **Retrieval.** BM25 against the current cell's banks: 5 adequate contrasts from $\mathcal{B}_c$ (suppression signal) and 5 prior enhancement cases from $\mathcal{B}_r$ (canonical-pattern signal). 3. **Proposal.** Draw 2 candidate edits from the enhancement LM given (d, S, e, g) plus retrieved contrasts and cases. 4. **Validation, clustering, ranking.** Drop grammar-invalid candidates; group survivors sharing (*kind*, *target*) into clusters; score by $0.3\,v + 0.5\,\mu + 0.2\,m$. 5. **Safety gate.** Drop concrete clusters with $\mu < \tau$. If none survive, return DEFER at rank 1. Return the top-k.

Table 2: The CASE per-input pipeline (Section 4.2). Retrieval count $(5 + 5)$, proposal sample count (2), scoring weights, and threshold $\tau = 0.10$ are calibrated on the development set (Appendix G).

port μ (per-kind weighted average of target match, payload Jaccard, and evidence overlap against the retrieved enhancement cases), and minimality m (penalty proportional to edit footprint). The composite ranking score is $0.3\,v + 0.5\,\mu + 0.2\,m$, with weights tuned on the development set (Appendix G).

The memory-score safety gate. When the detector fires on a gold-adequate sample, a confident LM with retrieval tends to write a plausible-sounding edit instead of abstaining. The gate turns that failure mode into an explicit DEFER: admit a concrete candidate x only if $\mu(x) \geq \tau$, otherwise return DEFER at rank 1.

The gate itself is a deterministic function of μ. $\mu(x)$ is zero whenever the cell's memory contains no precedent sharing x's edit kind and target, and the dev-set support distribution has an empty interval at $(0, 0.10)$ (Appendix J), so any τ inside the gap separates supported from unsupported candidates the same way on every input. Its empirical safety, though, is bounded by the proposer. On the closed-source LMs we tested, detector-triggered FC@k falls to 0.000. On the open-weight models it stops at 0.121–0.210: weaker proposers more often hallucinate edits whose $(\textit{kind}, \textit{target})$ pair coincides with a real in-cell precedent, and the gate cannot tell those apart from valid candidates. With that caveat in mind, the gate is a strong backstop on closed models and a partial one on open weights; closing the open-weight residual would need a stronger proposer or a learned verifier on top of the gate. The cost is that a true positive with no in-cell precedent is also deferred: we trade about 5% of dual-memory Semantic@k for the zero-FC behavior on closed models (Section 5).

5 Experiments

We investigate: **(Q1)** does the schema-issue detector detect inadequacy reliably enough to serve as a gate without hurting DST? **(Q2)** does retrieval-augmented enhancement beat a matched direct-LLM baseline? **(Q3)** does the safety gate eliminate unsupported edits without losing the retrieval gain?

5.1 Setup

Methods compared. All five methods run on identical detector-triggered samples with a matched proposal budget ($N = 2$ draws, $T = 0.6$, strict JSON): **Defer-only** always outputs DEFER; **Direct LLM + validation** prompts the enhancement LM with (d, S, e) plus grammar validation (no retrieval, gap extraction, or scoring); **CASE without retrieval** is CASE minus dual memory; **Dual-memory CASE** adds retrieval and scoring; **Safety-gated dual-memory CASE** adds the gate at τ=0.10 (headline).

Detector backbone	F_1	Prec.	Recall	FPR	Evid. rate
Llama 3.2 3B	0.834	0.855	0.813	0.136	1.00
Qwen3 4B	0.888	0.875	0.901	0.127	1.00
Qwen3 8B	0.896	0.886	0.906	0.116	1.00
Llama 3.1 8B	**0.926**	**0.928**	**0.924**	**0.071**	**1.00**

Table 3: Schema-issue detector detection results on the full 24,340-sample test set across four backbones. Evid. rate is the fraction of triggered predictions that include a valid evidence span. We use the Llama 3.1 8B detector as the fixed trigger surface for the enhancement experiments.

Enhancement LMs. Four LMs span size and openness: **Gemini 3 Flash** (closed, primary analysis), **Qwen3-32B** (open), **GPT-OSS-120B** (open), and **GPT-5.1** (closed frontier). The schema-issue detector (Llama 3.1 8B fine-tune) is held constant so the comparison isolates the enhancement LM.

Metrics. We report the four metrics of Section 2.3 at $k = 5$: Semantic@5 / FC@5 (detector-triggered) and A-Sem@5 / A-FC@5 (all-stage). Semantic@5 is the headline since canonical enhancements admit surface forms strict matching misses; Exact@5 is in Appendix I. We choose $k = 5$ to match a realistic human-review budget; the top-k curve is flat beyond $k = 5$ (Appendix I).

Data. The trigger surface is the Llama 3.1 8B detector; memory banks contain 8,091 contrast and 7,848 enhancement samples from the training split, both per-cell BM25 indexed.

5.2 Schema-issue detection (Q1)

The strongest backbone in Table 3 (Llama 3.1 8B) reaches $F_1 = 0.926$ with balanced precision and recall and emits an evidence span on every triggered prediction; joint training does not degrade DST (Appendix F), so the detector can serve as a Stage 1 gate without a separate DST tracker. It produces 886 false-positive triggers (≈7% FPR) and misses 926 of 12,099 gold positives (recall 0.924). The 886 false positives are the samples on which we evaluate detector-triggered FC@5; A-Sem@5 and A-FC@5 use the full-test denominators.

Method	Detector-triggered		All-stage	
	Sem@5 ↑	FC@5 ↓	A-Sem@5 ↑	A-FC@5 ↓
Defer-only	0.032	0.000	0.030	0.000
Direct LLM + validation	0.574	0.826	0.530	0.060
CASE without retrieval	0.615	0.812	0.568	0.059
Dual-memory CASE (no gate)	0.841	0.906	0.777	0.066
Safety-gated dual-memory CASE	**0.794**	**0.000**	**0.733**	**0.000**

Table 4: Enhancement results with Gemini 3 Flash as the enhancement LM. Detector-triggered metrics condition on the schema-issue detector firing (11,173 triggered positives for Semantic@5; 886 false-positive triggers for FC@5). All-stage metrics additionally charge missed positives against A-Sem@5 (12,099 gold-positive denominator) and use the full 12,241 gold-adequate denominator for A-FC@5. The safety gate eliminates concrete edits on the false-positive-trigger set, driving all-stage A-FC@5 to 0.

5.3 Enhancement quality and safety (Q2, Q3)

Table 4 reports the headline with Gemini 3 Flash. Direct LLM scores 0.574 Semantic@5 with FC@5 = 0.826, so it commits an edit on most false-positive triggers. Dual memory pulls both numbers up: Semantic@5 → 0.841 (+0.267) and FC@5 → 0.906, since retrieval makes the model more confident on true positives and on hallucinations alike. The gate keeps most of the recall and zeroes the safety cost: Semantic@5 = 0.794 and FC@5 = **0.000**, a +0.220 / −0.826 swing versus matched direct LLM. At the all-stage level the gated method reaches A-Sem@5 = 0.733 (88% of the 0.924 detector-recall ceiling) and A-FC@5 = 0.000, against 0.530/0.060 for direct LLM. We manually inspected 50 false-positive-trigger samples; on this closed model the gate demotes every hallucinated concrete candidate to DEFER. Two illustrative samples are in Appendix K.

5.4 Comparison with slot-induction baselines

We compare against three slot-induction baselines (Table 5): **GenDSI** (Finch et al., 2024), its streaming-refinement successor **Streaming-SSI** (Finch et al., 2025), and an **LLM-prompted inducer** using GPT-5.1 in offline-corpus mode—fed each cell's training dialogues plus the pre-edit schema and asked to propose a slot inventory. Each method is run per cell on the training split; at test time, induced slots are mapped to LEAFADD proposals via BM25 between induced slot description and user turn (Appendix E).

Induction has no detector and no DEFER, so the fair comparison is on the whole test set, not the detector-triggered subset. A-Sem@5 uses all 12,099 gold positives and A-FC@5 all 12,241 gold-adequate samples; detector-triggered columns are

Method	Detector-triggered		All-stage / whole-test	
	Sem@5 ↑	FC@5 ↓	A-Sem@5 ↑	A-FC@5 ↓
Slot-induction baselines (whole test; no detector gating; no DEFER)				
GenDSI (Finch et al., 2024)	—	—	0.318	0.824
Streaming-SSI (Finch et al., 2025)	—	—	0.235	0.752
LLM-prompted inducer (GPT-5.1)	—	—	0.512	0.414
Schema-enhancement methods (Gemini 3 Flash enhancement LM; from Table 4)				
Defer-only	0.032	0.000	0.030	0.000
Direct LLM + validation	0.574	0.826	0.530	0.060
CASE without retrieval	0.615	0.812	0.568	0.059
Dual-memory CASE (no gate)	0.841	0.906	0.777	0.066
Safety-gated dual-memory CASE	**0.794**	**0.000**	**0.733**	**0.000**

Table 5: Three slot-induction baselines compared with our schema-enhancement methods. Induction has no two-stage pipeline—no detector and no DEFER—so it is evaluated on the *entire* test set (12,099 gold positives, 12,241 gold-adequate samples). Detector-triggered metrics presuppose a detector stage that induction does not have and are marked "—". The best induction baseline (LLM-prompted GPT-5.1) reaches A-Sem@5 = 0.512—still 22 absolute points below safety-gated CASE (0.733)—while still committing concrete edits on 41.4% of gold-adequate samples (vs. 0.000 for our headline method). GenDSI and Streaming-SSI commit on 82%/75% of gold-adequate samples. Induction is run per cell on the training split; induced slots are mapped to LEAFADD proposals via BM25 against the user turn (Appendix E).

marked "—". The LLM-prompted inducer (GPT-5.1) is the strongest baseline at A-Sem@5 = 0.512, still 22 points short of safety-gated CASE (0.733); GenDSI and Streaming-SSI reach 0.318 and 0.235. Induction commits concrete edits on 41–82% of gold-adequate samples, versus 0.000 for ours.

Three structural failure modes of induction. (1) Over-proposal across topics. Induction turns *any* salient noun phrase into a slot regardless of task relevance, producing off-topic LEAFADD proposals on adequate samples and driving A-FC@5. **(2) Slot duplication.** Induction has no view of the existing schema and re-invents slots under different names (e.g., `user_destination` vs. `flight.destination`); many duplicates pass the validator as concrete commits. **(3) Append-only output.** Induction cannot express LEAFREPLACE, TYPENARROW, or GRANULARITY{SPLIT,MERGE}. On the 74% of gold positives whose canonical kind is non-additive, this caps semantic credit regardless of method quality and accounts for the 0.512 → 0.733 gap. None of the three are fixable by a better induction method.

5.5 Cross-model robustness

Table 6 spans the four enhancement LMs. Safety-gated CASE improves Semantic@5 by 14.4–22.0 points in every condition; detector-triggered FC@5

	Direct LLM (triggered)		Safety CASE (triggered)		All-stage (safety)	
Enhancement LM	Sem@5	FC@5	Sem@5	FC@5	A-Sem@5	A-FC@5
Qwen3-32B (open)	0.476	0.712	0.689	0.210	0.636	0.015
GPT-OSS-120B (open)	0.537	0.828	0.732	0.121	0.676	0.009
Gemini 3 Flash (closed)	0.574	0.826	0.794	**0.000**	0.733	**0.000**
GPT-5.1 (closed)	0.737	0.541	**0.881**	**0.000**	**0.814**	**0.000**

Table 6: Cross-model evaluation. *Detector-triggered* columns (Sem@5, FC@5) condition on the schema-issue detector firing; *All-stage* columns (A-Sem@5, A-FC@5) divide by the full 12,099 gold positives and 12,241 gold-adequate samples respectively, so detector misses are charged against recall and detector correctness lowers the FC denominator. A-Sem@5 is bounded above by detector recall (0.924). Schema-issue detector (Llama 3.1 8B fine-tune) held constant.

Gold kind	Direct LLM	Dual-mem	Safety-gated	Δ
Group A: additive (direct LLM near-ceiling)				
LeafAdd	0.972	0.993	0.939	−0.033
ValueExtend	0.949	0.983	0.884	−0.065
Group B: structural / relational				
LeafReplace	0.126	0.739	0.692	+0.566
GranularitySplit	0.610	0.976	0.927	+0.317
GranularityMerge	0.299	0.853	0.810	+0.511
TypeNarrow	0.050	0.332	0.347	+0.297
Defer	0.830	0.925	0.962	+0.132

Table 7: Per-kind Semantic@5 on the full test set (Gemini 3 Flash enhancement LM). The largest gains from dual memory + safety gate are on structural and relational kinds where prior schema-shaped precedent is most informative; on additive kinds the direct LLM is already near-ceiling. Δ is Safety-gated − Direct LLM.

drops by 50.2–82.6 points, reaching 0.000 only on closed models and stopping at 0.121–0.210 on open weights. At the all-stage level A-FC@5 falls to 0.000 on closed and 0.009–0.015 on open-weight because the detector deflects most adequates first; A-Sem@5 spans 0.636–0.814 against the 0.924 recall ceiling.

5.6 Per-kind analysis and failure modes

Table 7 shows two clean groups. *Additive edits* (LeafAdd, ValueExtend) are near-ceiling for direct LLM (0.97, 0.95); the gate trims top-5 diversity for a small cost (−0.033, −0.065). *Structural and relational edits* (LeafReplace, TypeNarrow, GranularitySplit, GranularityMerge) are far from ceiling for direct LLM (0.05–0.61); dual memory recovers canonical patterns (+0.566, +0.297, +0.317, +0.511).

Table 8 decomposes the 4,831 exact-match failures: 48.3% kind-selection, 46.5% payload-canonicalization, 5.2% target-selection. Payload canonicalization is the main bottleneck.

6 Related Work

Schema-guided DST and induction. Schema-Guided Dialogue (Rastogi et al., 2020) made typed

Failure category	Count	% of misses
Kind selection failed	2,334	48.3%
Payload canonicalization failed	2,245	46.5%
Target selection failed	252	5.2%
Total exact-match misses	4,831	100.0%
Exact-match recall	6,342 / 11,173 (56.8%)	

Table 8: Failure decomposition for safety-gated dual-memory CASE on the full test set (Gemini 3 Flash). Payload canonicalization is the principal remaining bottleneck.

service schemas central to multi-domain DST, and follow-on work compares schema encoders, prompts, and slot-name choices (Ye et al., 2022; Lee et al., 2022; Cao and Zhang, 2021; Lee et al., 2021; Zhao et al., 2022). Slot induction learns inventories from dialogue (Yu et al., 2022; Finch et al., 2024); schema-only DST uses synthetic data and distillation (Xu et al., 2025). None evaluates persistent typed enhancement with explicit defer.

LLM DST, ontology change, retrieval-augmented memory. LLM-based DST with function calls, open-vocabulary trackers, and schema augmentation (Li et al., 2024; Safa and Şahin, 2025; Finch and Choi, 2024; Richardson et al., 2024) improves tracking under changing coverage but assumes a given schema. Knowledge-graph change languages (Hegde et al., 2025) define ontology-edit primitives; our grammar grounds each edit in dialogue evidence with executable validation and Defer. Our dual memory builds on RAG (Lewis et al., 2020) and case-based agent memory (Zhou et al., 2025; Ouyang et al., 2026) but feeds retrieval into a safety gate. Continual DST (Feng et al., 2024) and task synthesis (Finch et al., 2025) update parameters or evaluation sets; we ask whether the schema itself should change.

7 Conclusion

Schema enhancement is a per-turn typed task with an explicit Defer. Our 20-cell benchmark and detect-then-enhance pipeline drive detector-triggered FC@k to zero on the closed-source proposers we tested while keeping 94% of the dual-memory recall gain. The 0.12–0.21 residual on open weights is the gate's limit in isolation: it filters the candidates the proposer offers, not the proposer's tendency to anchor on a precedented kind and target. Closing that gap is the next step.

8 Limitations

Synthetic dialogue surface. The benchmark uses LM-realized user turns under deterministic supervision rather than transcripts collected from real users. Test samples pass through unanimous human review (Appendix B), which catches surface artifacts that misalign with the planned edit, but it cannot rule out stylistic patterns in the LM's output that a detector or enhancement LM might pick up on. Generalization to live user populations should be measured directly before deployment.

Coverage. The four source corpora are English-only and cover Western consumer-service domains. The seven-kind grammar was motivated by failures observed in those domains, and we do not claim it is exhaustive for schemas with referential constraints, deeper type hierarchies, multilingual surface, or specialized regulatory contracts (medical, legal, financial). New cells in those regimes may produce failure modes our grammar does not cover.

Deployment risk. Schema enhancement is a privileged action: a committed edit changes the contract a downstream service is held to. A wrong commit can break downstream validation, expose unsafe value ranges, or leave the system in an inconsistent state across replicas. Our system is designed for human-in-the-loop deployment, where the top-k output is reviewed before any persistent commit; the released artifacts should not be treated as a drop-in autonomous schema mutator.

Adversarial inputs. We did not evaluate against users or upstream agents that try to coerce inappropriate edits through prompt injection, evidence-span spoofing, or memory-bank poisoning. A live deployment where end users can influence the dialogue turn would need a separate adversarial audit; our results say nothing about that threat model.

Closed-source dependence. The strongest reported numbers come from closed-source enhancement LMs (Gemini 3 Flash and GPT-5.1), whose behavior may change as vendors update their models. Open-weight backbones provide an independently reproducible reference point in Section 5.5, but the headline figures depend on APIs we do not control.

References

Bill Byrne, Karthik Krishnamoorthi, Chinnadhurai Sankar, Arvind Neelakantan, Ben Goodrich, Daniel Duckworth, Semih Yavuz, Amit Dubey, Kyu-Young Kim, and Andy Cedilnik. 2019. Taskmaster-1: Toward a realistic and diverse dialog dataset. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4516–4525, Hong Kong, China. Association for Computational Linguistics.

Jie Cao and Yi Zhang. 2021. A comparative study on schema-guided dialogue state tracking. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 782–796, Online. Association for Computational Linguistics.

Yujie Feng, Xu Chu, Yongxin Xu, Guangyuan Shi, Bo Liu, and Xiao-Ming Wu. 2024. TaSL: Continual dialog state tracking via task skill localization and consolidation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1266–1279, Bangkok, Thailand. Association for Computational Linguistics.

James D. Finch and Jinho D. Choi. 2024. Diverse and effective synthetic data generation for adaptable zero-shot dialogue state tracking. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12527–12544, Miami, Florida, USA. Association for Computational Linguistics.

James D. Finch, Yasasvi Josyula, and Jinho D. Choi. 2025. Generative induction of dialogue task schemas with streaming refinement and simulated interactions. *arXiv preprint arXiv:2504.18474*.

James D. Finch, Boxin Zhao, and Jinho D. Choi. 2024. Transforming slot schema induction with generative dialogue state inference. In *Proceedings of the 25th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 317–324, Kyoto, Japan. Association for Computational Linguistics.

Harshad Hegde, Jennifer Vendetti, Damien Goutte-Gattat, J. Harry Caufield, John B. Graybeal, Nomi L. Harris, Naouel Karam, Christian Kindermann, Nicolas Matentzoglu, James A. Overton, Mark A. Musen, and Christopher J. Mungall. 2025. A change language for ontologies and knowledge graphs. *Database*, 2025:baae133.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*. ACM.

Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. 2021. Dialogue state tracking with a language model using schema-driven prompting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4937–4949, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Harrison Lee, Raghav Gupta, Abhinav Rastogi, Yuan Cao, Bin Zhang, and Yonghui Wu. 2022. SGD-X: A benchmark for schema robustness in task-oriented dialogue. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zekun Li, Zhiyu Zoey Chen, Mike Ross, Patrick Huber, Seungwhan Moon, Zhaojiang Lin, Xin Luna Dong, Adithya Sagar, Xifeng Yan, and Paul A. Crook. 2024. Large language models as zero-shot dialogue state tracker through function calling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8688–8704, Bangkok, Thailand. Association for Computational Linguistics.

Johannes E. M. Mosig, Shikib Mehri, and Thomas Kober. 2020. STAR: A schema-guided dialog dataset for transfer learning. *arXiv preprint arXiv:2010.11853*.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. 2026. ReasoningBank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations (ICLR)*. ArXiv:2509.25140.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696.

Christopher Richardson, Roshan Sharma, Neeraj Gaur, Parisa Haghani, Anirudh Sundar, and Bhuvana Ramabhadran. 2024. Schema augmentation for zero-shot domain adaptation in dialogue state tracking. *arXiv preprint arXiv:2411.00150*.

Abdulfattah Safa and Gözde Gül Şahin. 2025. A zero-shot open-vocabulary pipeline for dialogue understanding. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7562–7579, Albuquerque, New Mexico. Association for Computational Linguistics.

Huan Xu, Zequn Li, Wen Tang, and Jian Jun Zhang. 2025. From schema to state: Zero-shot scheme-only dialogue state tracking via diverse synthetic dialogue and step-by-step distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1640–1652, Suzhou, China. Association for Computational Linguistics.

Fanghua Ye, Jarana Manotumruksa, and Emine Yilmaz. 2022. MultiWOZ 2.4: A multi-domain task-oriented dialogue dataset with essential annotation corrections to improve state tracking evaluation. In *Proceedings of SIGDIAL*, pages 351–360, Edinburgh, UK. Association for Computational Linguistics.

Dian Yu, Mingqiu Wang, Yuan Cao, Izhak Shafran, Laurent Shafey, and Hagen Soltau. 2022. Unsupervised slot schema induction for task-oriented dialog. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1174–1193, Seattle, United States. Association for Computational Linguistics.

Jeffrey Zhao, Raghav Gupta, Yuan Cao, Dian Yu, Mingqiu Wang, Harrison Lee, Abhinav Rastogi, Izhak Shafran, and Yonghui Wu. 2022. Description-driven task-oriented dialog modeling. *arXiv preprint arXiv:2201.08904*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. 2024. LlamaFactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 400–410, Bangkok, Thailand. Association for Computational Linguistics.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. 2025. Memento: Fine-tuning LLM agents without fine-tuning LLMs. *arXiv preprint arXiv:2508.16153*.

A LLM Usage

We used a general-purpose LLM assistant for two non-experimental tasks: polishing grammar and English phrasing in the manuscript, and writing routine boilerplate code (data loaders, plotting, test fixtures). All technical claims, citations, and code that produces reported numbers were authored and reviewed by the human authors. LLMs used *inside* the experimental pipeline are described in Section 5 and Appendix G.

B Annotators

Test-sample human review (Section 3.4) was carried out by unpaid volunteer annotators from the authors' lab. All were fluent English speakers familiar

with task-oriented dialogue data; the workload was light enough that we did not recruit external crowd-workers.

Review prompt. Each reviewer was shown the dialogue turn, the pre-edit schema rendered as a slot table, the proposed gold edit, and the evidence span highlighted in the user turn. The instruction was:

> *You will see dialogue turns paired with a proposed schema edit and an evidence span. For each sample, decide:*
>
> *(1) Does the evidence span appear verbatim in the user turn?*
>
> *(2) Does the user turn motivate a change to the schema, rather than a state-tracking decision the current schema already supports?*
>
> *(3) Is the proposed edit the smallest typed change that lets the schema represent the user's request, given the edit grammar shown in the reference card?*
>
> *Mark **Accept** only if all three are yes. Mark **Reject** otherwise. If you cannot decide, mark **Unsure**; the sample will be excluded rather than adjudicated.*

A sample entered the released test set only when every reviewer returned **Accept**; this yields the 94.1% / 4.3% / 1.6% accept / reject / excluded split in Section 3.4.

C Edit Grammar, Audit, and Kind Descriptions

Audit motivation for the seven kinds. We sampled ∼50 dialogue turns from each of the four source corpora where the schema appeared inadequate, and grouped the observed failure modes by what *must change* in the schema artifact. The grouping converged to seven kinds: four operate on a single existing or new element (LEAFADD, LEAFREPLACE, VALUEEXTEND, TYPENARROW); two operate on element-count (GRANULARITYSPLIT, GRANULARITYMERGE); and DEFER is the no-commit action. We illustrate one observed failure per kind:

- LEAFADD – SGD `hotels`: "Is there a pet fee?" when no `pet_fee` slot exists.
- LEAFREPLACE – Taskmaster `flight`: a `flight1_detail_other_description` catch-all slot is used to record flight duration; the canonical fix replaces it with a dedicated `duration` slot.
- VALUEEXTEND – MultiWOZ `hotel`: user requests `area="waterfront"` but the enum only lists cardinal areas.
- TYPENARROW – STARv2 `bank`: `transfer_amount` is typed `string`; downstream service needs a numeric type with currency code.
- GRANULARITYSPLIT – SGD `flights`: `stops` conflates layover count and layover duration; the user distinguishes both.
- GRANULARITYMERGE – Taskmaster `flight`: separate `flight1_*` and `flight2_*` groups should be merged into a list-of-flight-segments structure when the user describes multi-leg trips uniformly.
- DEFER – any cell: ambiguous one-off requests where the user's intent is clearer as a query than a schema change.

The seven kinds are not claimed exhaustive of schema-failure space in general; richer hierarchies or referential integrity constraints could motivate additional kinds. They cover every failure observed in our audit and are sufficient for the benchmark's evaluation.

Payload shapes (validator-enforced).

- LEAFADD(t, p): t new schema path; p = (name, description, type, optional value space).
- LEAFREPLACE(t, p): t existing leaf; p = (new name and/or description, optional type change).
- VALUEEXTEND(t, p): t value space; p = (set of new values, optional descriptions).
- TYPENARROW(t, p): t slot or value space; p = narrower type or constrained value space.
- GRANULARITYSPLIT(t, p): t coarse concept; p = (finer children, mapping rule).
- GRANULARITYMERGE(t, p): t set of fine concepts; p = (merged parent, mapping rule).
- DEFER: empty payload.

The deterministic validator checks target existence in S, payload shape conformance, type compatibility, and validity of the post-edit schema.

D Benchmark Detail

The 20 cells. MultiWOZ 2.4 (`hotel`, `train`); SGD (`buses`, `events`, `flights`, `hotels`, `restaurants`, `services`); Taskmaster (`flight`, `hotel`, `movie`, `restaurant`, `ride`, `sports`); STARv2 (`apartment`, `bank`, `hotel`, `party`, `restaurant`, `ride`).

Source corpus choice. The four corpora were chosen because (a) each provides typed schemas, not free-text intents; (b) together they cover transactional (banking, flights), informational (restaurants, services), and deliberative (apartment hunting, party planning) service types; (c) they have non-overlapping dialogue populations, so cells can be assembled without dialogue-level leakage. We exclude STAR-1 (subsumed by STARv2) and MediTOD (different downstream contract).

Cell organization motivation. A cell is one $(\textit{dataset}, \textit{domain})$ pair, e.g. `sgd.flights` or `starv2.bank`. Cells separate within-cell and cross-cell generalization and scope memory banks per service.

Negative family construction. **Paired adequate** – for each positive sample, a paired sample with similar dialogue surface but schema already adequate; **cosmetic description tweak** – trivial slot-description changes; **cosmetic enum extension** – an additional enum value mentioned in dialogue that the schema already accepts under a synonym; **cosmetic irrelevant slot** – schema slot mentioned in dialogue but unrelated to the user's request.

E Slot-Induction Baseline Protocol

Conceptual taxonomy. Before describing the operational protocol, Table 9 summarizes the seven axes along which schema enhancement differs from slot schema induction (Finch et al., 2024, 2025). The two most consequential are *output expressiveness* (induction produces only additive slot inventories; enhancement produces typed edits in six concrete kinds plus DEFER) and *safety semantics* (induction has no over-proposal penalty; enhancement pays for unsupported commits with FC@5).

To compare against slot-induction methods on a single common surface, we run each induction baseline per-cell on the training-split dialogues (matching the per-cell scoping of our memory banks) and adapt their outputs to our typed-edit grammar.

Per-cell induction. For each of the 20 cells we feed the cell's training-split dialogue turns to the induction method and obtain an induced slot inventory $\mathcal{I}_{\text{cell}}$, where each element is (induced slot name, induced slot description, optional value set). Methods that operate across the whole corpus are restricted to the cell's data so they cannot exploit cross-domain text overlap that our enhancement memory banks are prevented from using.

New-slot extraction. For each cell we compute $\mathcal{N}_{\text{cell}} = \mathcal{I}_{\text{cell}} \setminus \text{slots}(S)$ where S is the cell's pre-edit schema. $\mathcal{N}_{\text{cell}}$ is the set of *newly induced* slots that would correspond to LEAFADD edits.

Mapping induced slots to edit candidates (whole test). Induction has no detector stage and no DEFER, so the fair evaluation runs it on the *whole* test set, not the detector-triggered subset. For each test sample (d, S) in the cell, we score each $n \in \mathcal{N}_{\text{cell}}$ by BM25 similarity between n's induced description and the user turn of d. The top-five $\{n_1, \ldots, n_5\}$ by score become LEAFADD candidates $(\text{LEAFADD}, n_i.\text{name}, n_i.\text{payload})$ for the sample.

Mandatory output, modulo validator rejection. Induction baselines have no DEFER; on every test sample the mapping procedure tries to emit LEAFADD candidates. The deterministic typed-grammar validator can still reject some candidates—most commonly when the induced slot duplicates an existing schema slot (different name, same concept) and fails type or path checks, or when the BM25 mapping produces no induced slot scoring above zero. As a result, induction's empirical A-FC@5 is below 1.0 but remains very high (0.414–0.824 in our experiments, Table 5) because most induced slots pass the validator and become concrete commits on adequate samples.

Methods compared. **GenDSI** (Finch et al., 2024): generative slot schema induction; we use the released SIGDIAL 2024 implementation. **Streaming-SSI** (Finch et al., 2025): the streaming-refinement successor to GenDSI; we use the released code from the same lab and process each cell's training dialogues sequentially. **LLM-prompted inducer (GPT-5.1)**: we prompt GPT-5.1 with the cell's training dialogues plus the pre-edit schema and ask it to propose the slot inventory it would add to make the schema cover the observed user needs. The prompt is offline-corpus-shaped (the model sees all training dialogues for the cell

Axis	Slot schema induction (Finch et al., 2024, 2025)	Schema enhancement (this work)
Operating mode	Offline batch over a dialogue corpus	Per-turn online, gated by the schema-issue detector
Output unit	Flat slot inventory: (slot name, type, value set)	Typed edit (k, t, p) over an existing schema, or DEFER
Edit kinds covered	LEAFADD only (implicitly: every induced slot is an addition)	LEAFADD, LEAFREPLACE, VALUEEXTEND, TYPENARROW, GRANULARITYSPLIT, GRANULARITYMERGE
DEFER action	Not modeled; cannot abstain on a turn	First-class output; gold action on ≈5% of positives plus all triggered negatives
Evidence binding	Aggregate corpus statistics (turn-level evidence is averaged out)	Per-turn evidence span emitted by the schema-issue detector and used by the proposer
Safety semantics	None; over-induction is silently absorbed into the inventory	FC@5 on triggered negatives; structural memory-score safety gate
Empirical A-Sem@5 / A-FC@5 on our test set	$\leq$ 0.512 / $\geq$ 0.414 across three induction baselines (Table 5)	0.733 / 0.000 realized (Gemini 3 Flash, safety-gated CASE; Table 4)

Table 9: Schema induction and schema enhancement are different tasks. Induction expresses only additive slot proposals (no LEAFREPLACE, TYPENARROW, GRANULARITY{SPLIT,MERGE}) and has no DEFER, so it cannot match non-additive gold edits and cannot abstain on adequate turns. Empirically the three induction baselines we test reach at most 0.512 A-Sem@5 and at least 0.414 A-FC@5 (Section 5.4); our headline method reaches 0.733 / 0.000.

at once, not one turn at a time) so that the comparison preserves the induction–enhancement task distinction. The prompt template is released with the artifacts. This is not a literature method but quantifies how much of the gap closes by simply scaling the LM under the offline-corpus framing—it directly addresses the "would a bigger LM in induction mode solve schema enhancement" question.

Evaluation. For induction baselines we report only the pipeline (whole-test) metrics: A-Sem@5 on all 12,099 gold positives and A-FC@5 on all 12,241 gold-adequate samples. Induction earns Semantic@5 credit only on samples where its LEAFADD proposal is semantically equivalent to the gold edit; on positives whose gold kind is not LEAFADD it can earn at best partial credit via the same-target eligibility rule (Appendix H). Detector-triggered metrics presuppose a detector stage that induction does not have, so they are marked "—" in Table 5. Semantic@5 of the top-five LEAFADD candidates uses the same cross-encoder evaluator and piecewise scoring as for our schema-enhancement methods, applied to the (candidate, gold) edit-card pair.

F DST Impact of Joint Training

We compare two Llama 3.1 8B detector variants on a held-out DST evaluation slice: (i) DST-only fine-tuning and (ii) joint DST + schema-issue fine-tuning. The joint schema-issue detector matches DST-only within 0.5 joint-goal-accuracy points and 0.3 average-slot-F1 points, with no statistically significant difference. The schema-issue head is effectively-free to add at the deployment-time detector contract.

G Implementation Details

Detector fine-tuning. Four schema-issue detector sizes are fine-tuned on the benchmark training split: Llama 3.2 3B, Qwen3 4B, Qwen3 8B, Llama 3.1 8B. The training objective produces both dialogue state and the schema_issue head jointly (LoRA fine-tuning; full configs released).

Enhancement LM sampling. All four enhancement LMs (Gemini 3 Flash, Qwen3-32B, GPT-OSS-120B, GPT-5.1) are queried in strict JSON mode at temperature 0.6 with N=2 self-consistency draws per sample. Malformed or ungrounded outputs are retained in the audit trace as proposal failures rather than silently discarded.

Retrieval banks. The contrast bank contains 8,091 (d, S) pairs from training samples where the schema-issue detector predicts adequacy and the schema is unchanged. The enhancement memory bank contains 7,848 $(d, S,$ enhancement pattern$)$ triples from gold-positive training samples. Both banks are indexed per cell with BM25; we retrieve k=5 per channel.

Scoring weights. The composite candidate score is $0.3\,v{+}0.5\,\mu{+}0.2\,m$, where v is the proposal vote

count, μ is the per-kind weighted average of target match, payload Jaccard, and evidence overlap against the retrieved enhancement cases, and m is a minimality penalty. The weights were selected by a small grid search on the development set: we swept each weight in $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$ subject to $w_v + w_\mu + w_m = 1$ and picked the triple that maximized development-set Semantic@k while keeping development-set FC@k at zero under the safety gate. The selected $(0.3, 0.5, 0.2)$ is then applied unchanged on the test set.

Enhancement LM prompt skeleton. Each enhancement call sends a JSON payload with five sections: (a) the truncated dialogue context, (b) the current schema S rendered as a list of (slot, type, description, value space) entries, (c) the detector-emitted evidence span e, (d) the retrieved adequate-memory contrasts and positive enhancement cases (rendered as lightweight $(d, S, \text{enhancement pattern})$ summaries), and (e) the typed-edit grammar with payload schema for each kind. The LM is asked to emit a JSON object with fields `kind`, `target`, `payload`, `evidence_span`, `gap_phrase`, `minimality_reason`, `confidence`. Strict JSON parsing is enforced; malformed responses count as proposal failures and are recorded in the audit trace.

H Metric Details and Validation

Semantic@5 definition. For each gold-positive triggered sample, the evaluator renders each valid top-five candidate and the gold edit as symmetric natural-language edit cards. Candidate–gold pairs are eligible when they share edit kind, or when both are slot-touching edits on the same target. Eligible pairs are scored with a cached BGE-RERANKER-V2-M3 cross-encoder under a sigmoid head, and probabilities are mapped to credit by a piecewise linear function: $< \theta_{\text{low}}$ scores 0, $> \theta_{\text{high}}$ scores 1, and $[\theta_{\text{low}}, \theta_{\text{high}}]$ is linearly rescaled.

Threshold calibration via human pilot. The thresholds $(\theta_{\text{low}}, \theta_{\text{high}}) = (0.30, 0.95)$ are not arbitrary. We built a 200-pair human-validation pilot: two annotators labeled 200 (candidate, gold) edit-card pairs spanning all seven edit kinds and all 20 cells as either "equivalent" or "not equivalent" ($\kappa = 0.78$). We selected θ_{high} as the lowest cross-encoder probability at which both annotators consistently labeled the pair equivalent (≥ 0.95), and θ_{low} as the highest probability below which both annotators labeled it non-equivalent (≤ 0.30). The piecewise band $[0.30, 0.95]$ captures the disagreement zone where one annotator's call differs from the cross-encoder's confidence by a small margin, and we award partial credit proportionally. Under these thresholds, the cross-encoder agrees with the majority human label on 94% of the 200 pairs; we report Semantic@5 as our primary metric on this basis.

Exact@5 definition. Exact@5 counts a candidate as a hit iff its $(kind, target, payload)$ triple matches the gold edit, modulo a deterministic payload-canonicalizer (synonym resolution for slot names, value-set permutation, type-preserving renames). We report it as a strict surface-equivalence sanity check; the qualitative ordering of methods under Exact@5 is the same as under Semantic@5, but absolute values are lower because canonical paraphrases of the same edit receive zero credit.

FC@5. Computed deterministically against the typed grammar: a candidate counts as concrete if it (i) is not DEFER, (ii) passes the validator, and (iii) appears in the top five ranked candidates. FC@5 is the fraction of gold-negative triggered samples with at least one such candidate.

I Top-k Curve and Exact@5 Comparison

Safety-gated dual-memory CASE with Gemini 3 Flash on the full test set:

k	Semantic@k	Exact@k
1	0.764	0.696
3	0.787	0.777
5	0.794	0.790
10	0.794	0.792
20	0.794	0.792

The curve is flat beyond $k = 5$ for all methods evaluated, and Exact@5 tracks Semantic@5 closely (within 0.4 absolute points), confirming that the cross-encoder is not awarding spurious credit on top of the strict payload-canonicalizer.

J Safety-Gate Threshold Sensitivity

Trimodal support distribution on the development set. On the development split, the enhancement-memory support distribution over concrete candidates is *trimodal*: about 26% of candidates score exactly 0 (no same-kind precedent in the cell), essentially no candidates fall in the interval $(0, 0.10)$ (a hard support gap), and the

remaining candidates form a continuous positive cluster from roughly 0.30 upward, peaking around 0.70–0.85. The $\mu(x) = 0$ spike empirically corresponds to unsupported concrete edits on triggered negatives; any $\tau \in (0, 0.10]$ catches exactly the zero-support candidates without affecting the supported cluster, so we calibrate $\tau = 0.10$ on the development set and apply it unchanged to the test set.

Test-set sensitivity sweep. We swept $\tau \in \{0, 0.05, 0.10, 0.30, 0.50\}$ on the test set with Gemini 3 Flash as the enhancement LM:

τ	Semantic@5	FC@5
0.00 (off)	0.841	0.906
0.05	0.794	0.000
0.10	**0.794**	**0.000**
0.30	0.790	0.000
0.50	0.778	0.000

The test-set support distribution also has the hard gap on $(0, 0.10)$ that was used to calibrate τ, so any $\tau \in (0, 0.10]$ yields identical outputs. Above 0.30 the threshold begins cutting supported candidates.

K Qualitative Examples

Table 10 grounds the recall–safety tradeoff in two test samples: a true positive where direct LLM proposes LEAFADD but LEAFREPLACE is canonical, and a false-positive-trigger sample (gold-adequate, paired-adequate family) where the direct LLM hallucinates a baggage slot and the safety gate correctly demotes the candidate to DEFER. We plan to expand this section in future releases.

L Open-Source Software

Fine-tuning. All schema-issue detector fine-tuning runs (Llama 3.1 8B, Llama 3.2 3B, Qwen3 4B, Qwen3 8B) use **LlamaFactory** (Zheng et al., 2024) with LoRA adapters. We rely on its trainer, dataset loaders, and chat-template handling; the per-backbone training YAMLs are released alongside the benchmark.

Open-weight inference. Inference for the open-weight enhancement LMs (Qwen3-32B and GPT-OSS-120B) and the open-weight detector backbones uses **vLLM** (Kwon et al., 2023) with structured-output decoding constrained by the typed edit grammar. Closed-source enhancement LMs (Gemini 3 Flash, GPT-5.1) are queried through their respective APIs in strict-JSON mode.

Example A: structural enhancement (`taskmaster.flight`)	
User	"I'd like a flight to Houston on March 20th that **takes about 8 hours**."
Schema	has a vague catch-all `other_description` slot
Gold	LEAFREPLACE `other_description` → dedicated `duration` (string) slot
Direct LLM	LEAFADD `flight1_detail_duration` (adds new slot, leaves vague slot in place)
Safety CASE	LEAFREPLACE `other_description` → `duration` (memory support 0.83)
Example B: safe deferral on a false-positive trigger (`taskmaster.flight`, paired-adequate)	
User	"I would like first class with only one layover, and **I'll need to check a bag**."
Schema	already supports baggage and class via existing slots
Gold	DEFER (schema is adequate for this turn)
Direct LLM	LEAFADD `checked_bag` (hallucinated commit)
Safety CASE	DEFER (LLM proposal had memory support 0.0; gate demoted, Defer promoted)

Table 10: Two test samples showing the recall–safety tradeoff. **A:** direct LLM identifies the right schema pressure but proposes the wrong structural action, leaving the vague catch-all in place; CASE retrieves a same-cell precedent and proposes the canonical LEAFREPLACE. **B:** a false-positive trigger (schema is actually adequate, but the schema-issue detector wrongly fires). Direct LLM hallucinates a baggage slot; the safety gate finds no same-kind precedent and demotes the candidate to DEFER—the gate catches the schema-issue detector's error.

Other dependencies. BM25 retrieval over the contrast and enhancement banks uses `rank-bm25`. The Semantic@5 cross-encoder is BGE-RERANKER-V2-M3 loaded via `sentence-transformers`. The deterministic validator, CASE pipeline, and evaluation harness are implemented in Python; all versions are pinned in the released `uv.lock`.