Scientific Reports

https://doi.org/10.1038/s41598-026-42158-y

Article in Press

Evaluating routing stability and coordination in swarm-based multi-agent task-oriented dialogue systems

Received: 5 January 2026

Accepted: 24 February 2026

Published online: 03 March 2026

Cite this article as: Khan A., Masood F., Iqbal A. *et al.* Evaluating routing stability and coordination in swarm-based multi-agent task-oriented dialogue systems. *Sci Rep* (2026). https://doi.org/10.1038/s41598-026-42158-y

Abuzar Khan, Fahad Masood, Abid Iqbal, Ahmad Junaid, Saad Arif, Mohammed Al-Naeem, Ghassan Husnain & Ali Saeed Alzahrani

Evaluating Routing Stability and Coordination in Swarm-Based Multi-Agent Task-Oriented Dialogue Systems

Abuzar Khan[1], Fahad Masood[1], Abid Iqbal[2,*], Ahmad Junaid[1], Saad Arif[3], Mohammed Al-Naeem[4], Ghassan Husnain[1,*], and Ali Saeed Alzahrani[2]

[1]Department of Computer Science, CECOS University of IT and Emerging Sciences, Pakistan
[2]Department of Computer Engineering, College of Computer Sciences and Information Technology, King Faisal University, Al Ahsa 31982, Saudi Arabia
[3]Department of Mechanical Engineering, College of Engineering, King Faisal University, Al Ahsa 31982, Saudi Arabia
[4]Department of Computer Networks Communications, CCSIT, King Faisal University, Al Ahsa 31982, Saudi Arabia
[*]Corresponding authors: ghassan.husnain@gmail.com; aaiqbal@kfu.edu.sa

ABSTRACT

Conversational systems are becoming a primary interface for services and enterprise automation, and rapid market growth is pushing deployments into safety- and cost-sensitive settings. Reliability remains a bottleneck when interactions span multiple domains: an orchestrator must choose the next specialist, maintain shared dialogue state, and recover from mistakes before they cascade across handoffs. Despite rising interest in swarm-like multi-agent designs, orchestration is rarely evaluated with coordination-centric metrics, making it hard to compare routing policies beyond surface fluency. We present an evaluation-first pipeline for multi-domain task-oriented dialogue on MultiWOZ 2.2 that decouples routing from generation and exposes measurable failure modes. A DeBERTa-based router selects domain specialists, while a FLAN-T5 generator produces structured actions and belief-state updates under a shared memory interface. The protocol tracks delegation correctness, slot-progress coverage, switching and bouncing instability, loop behavior, and recovery after misroutes, and it links early-turn errors to downstream collapse using cascading-error attribution. We further introduce stress tests that simulate reformulation, long-horizon corrections, and tool-latency delays to probe robustness beyond static annotations. Across routing variants, confidence-aware gating yields the strongest stability improvement, achieving routing accuracy of 0.77 while substantially reducing handoff churn, with switching 0.11 and bounce 0.01, relative to a learned baseline with 0.65 accuracy, switching 0.44, and bounce 0.09. At the same time, confidence gating can trade progress for precision when it suppresses belief updates, highlighting an accuracy-progress tension that is important for deployment tuning. Diagnostic summaries identify misrouting and empty-state updates as dominant contributors, while looping is comparatively rare. Finally, applying the same evaluation to SGD shows that coordination challenges persist under schema shift. Overall, the proposed metrics and implementation blueprint provide a reproducible basis for diagnosing coordination failures and selecting orchestration policies for deployment.

1 Introduction

Conversational systems have moved from experimental prototypes to a frontline interface for service delivery, product discovery and enterprise automation.[1] This shift is driven by rapid advances in large language models (LLMs) and by clear commercial pull.[2] Recent market estimates place the global conversational AI market at USD 11.58 billion in 2024, with projections reaching USD 41.39 billion by 2030, while broader generative AI revenues are projected to grow toward USD 1.3 trillion by 2032.[3] In parallel, enterprise leaders are actively testing customer facing deployments, with surveys reporting that 85 percent of customer service leaders will explore or pilot customer facing conversational generative AI in 2025.[4] Despite this momentum, real deployments still struggle with reliability when conversations span multiple skills, tools and domains, such as booking travel while coordinating schedules and resolving payment or policy constraints.[5] These multi domain interactions expose a core systems problem: the model must decide what to do next, which specialist capability should act, what information must be shared and when to stop.[6] As shown in Figure 1, the conversational agent architecture decomposes a dialogue system into modular components responsible for understanding user input, managing dialogue flow, accessing external knowledge and generating responses.

A common approach to building complex assistants is to decompose functionality into a collection of specialist modules coordinated by an orchestrator, often described as a swarm of agents[8]. Each specialist focuses on a narrower domain or

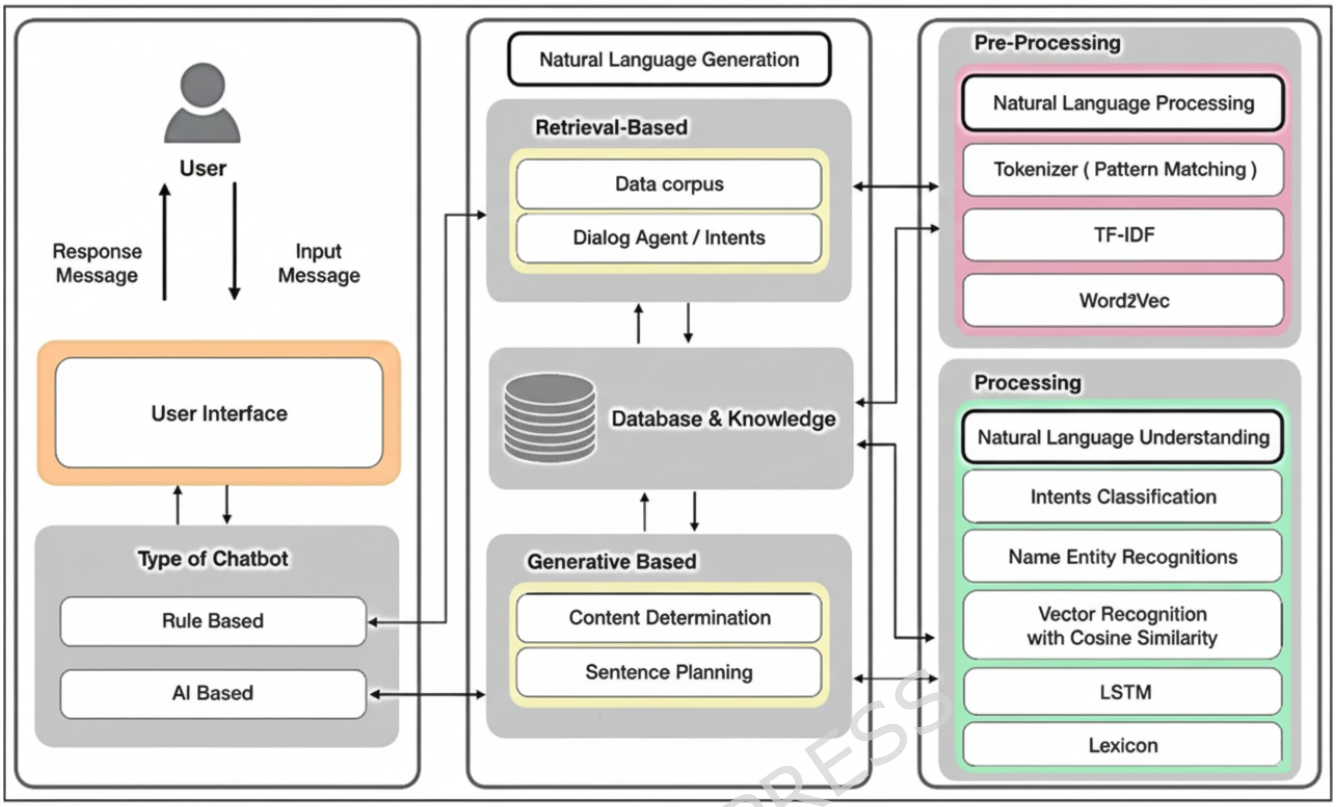


Figure 1. Conversational agent architecture[7].

capability, while the orchestrator is responsible for turn-level routing, shared state management, and aggregation of outcomes[9]. In this work, the term *swarm* refers to this modular multi-agent decomposition, while control is intentionally implemented via a centralized orchestrator to ensure reproducibility and enable controlled measurement of coordination behavior. This design simplifies shared-state consistency and allows routing stability, recovery, and cascading-error effects to be attributed directly to orchestration decisions rather than to emergent peer-to-peer negotiation dynamics. Fully decentralized coordination is a complementary direction, but it introduces additional confounding factors such as distributed state agreement, negotiation overhead, and non-deterministic execution trajectories, which are outside the scope of this evaluation-first study.

Despite growing interest in agentic and multi-agent systems, the scientific foundations of orchestration remain underdeveloped[10]. In particular, there is no widely accepted methodology for evaluating teamwork quality, including whether an orchestrator delegates to the correct specialist, maintains coverage of the user goal, avoids redundant cycles, and recovers after errors[11]. Predictability is also limited, as small routing mistakes can cascade into inconsistent dialogue state, repeated tool invocations, or unstable handoffs that increase latency and operational cost[12]. These challenges are especially pronounced in task-oriented dialogue, where success depends not only on fluent responses but also on satisfying explicit constraints and completing structured subgoals[13]. MultiWOZ 2.2 provides a practical testbed for studying these issues, as it contains goal-driven, multi-domain dialogues with supervision signals that enable systematic evaluation of routing correctness and progress maintenance across domain switches[14].

Unlike model-centric task-oriented dialogue work that primarily improves individual components or architectures, this work targets the orchestration problem through an *evaluation-first* lens.[15] Our objective is to define a reproducible pipeline that quantifies orchestration quality using coordination-centric metrics rather than surface fluency alone.[16] We operationalize key dimensions of teamwork as measurable signals: delegation quality, measured by whether routing selects the appropriate specialist for the current user intent and context; coverage, measured by progress toward satisfying constraints and requested slots across the dialogue; loop rate, measured by repeated routing patterns and stalled state updates; and recovery, measured by the ability to return to the correct domain trajectory after a misroute or missing information event.[16] We implement a modular orchestration baseline in which a router based on microsoft/deberta-v3-base predicts the next domain or agent, while a generation component based on google/flan-t5-base produces structured actions and responses conditioned on shared state.[17] This design isolates routing errors from generation errors and enables controlled experiments on stability as the number of

specialists and handoffs increases.[16]

Our contributions are threefold. First, we propose an evaluation protocol for orchestration on MultiWOZ 2.2 that emphasizes coordination outcomes and supports apples to apples comparison between rule based routing and learned routing. Second, we provide an end to end implementation blueprint suitable for constrained compute settings, enabling rapid iteration on routing policies, memory representations and stopping rules while tracking both quality and resource use. Third, we present an analysis of failure modes that explains when and why orchestration breaks, highlighting conditions that amplify cascading errors, such as ambiguous domain transitions, long context dependencies and sparse user confirmations. The motivation is to move beyond anecdotal demonstrations and toward measurable engineering science for multi-agent systems (MAS) in dialogue, where orchestration choices can be optimized with clear metrics and where claims about reliability and scalability can be validated under standardized conditions. Figure 2 illustrates the evaluation-first orchestration architecture, where a central orchestrator coordinates multiple specialized agents under shared dialogue state while evaluation signals track routing stability, recovery and error propagation.

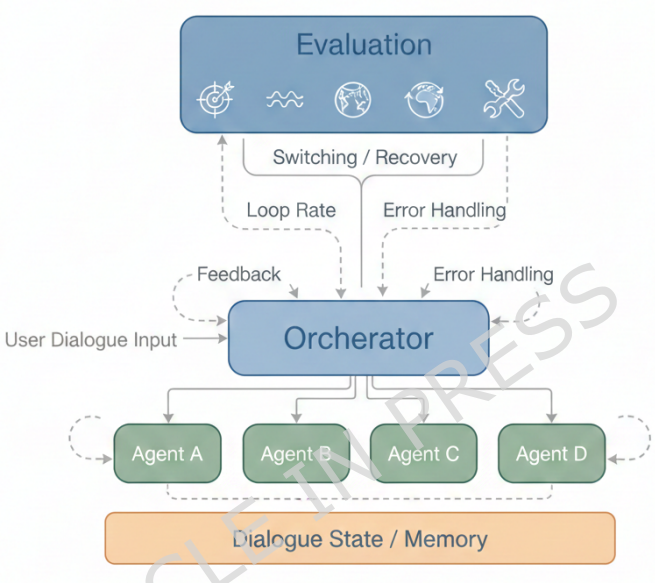


Figure 2. Evaluation-first orchestration framework for multi-agent task-oriented dialogue, illustrating routing, feedback, recovery and coordination between agents under a shared dialogue state.

The remainder of this paper is organized as follows: Section 2 reviews related work, Section 3 details the proposed methodology, Section 4 presents experimental results, Section 5 discusses comparisons and Section 6 concludes with future directions.

2 Literature Review

This section reviews prior work on task-oriented dialogue, dialogue state tracking and multi-agent orchestration and identifies limitations in existing evaluation practices that motivate our proposed framework.

2.1 Task-oriented dialogue and dialogue state tracking

Task-oriented dialogue (TOD) systems are typically built around explicit representations of user goals, constraints and system actions, with dialogue state tracking (DST) acting as the core mechanism that maintains the evolving belief state across turns.[18] MultiWOZ introduced a large-scale multi-domain benchmark that catalyzed research on scalable DST and end-to-end TOD modeling across domains such as hotel, restaurant, taxi and train.[19] Subsequent revisions addressed annotation noise and inconsistencies that significantly affected DST evaluation and model comparisons, notably MultiWOZ 2.1 and MultiWOZ 2.2.[20] These updates enabled more reliable experimentation and encouraged stronger baselines for state tracking.[21]

Within DST, a key trend has been moving from closed-vocabulary classification toward open-vocabulary approaches that reduce dependence on fixed ontologies and improve transfer across domains.[22] TRADE proposed a generative formulation with copy mechanisms to handle unseen slot values and enable cross-domain parameter sharing.[23] Later work explored more efficient incremental state updates, such as SOM-DST, which models dialogue state as an explicit memory and selectively overwrites only the updated parts rather than predicting the full state from scratch each turn.[24] TripPy further developed

value-independent tracking with multiple copy strategies to extract slot values directly from context without enumerating candidate lists.[25] Collectively, these methods highlight that multi-domain TOD performance is tightly coupled to robust state maintenance, especially under long contexts and frequent domain transitions.[26]

2.2 Modular tool use and multi-agent orchestration with LLMs

Recent LLM-centric systems increasingly adopt modular and agentic designs that separate high-level planning and routing from specialized execution.[27] MRKL systems formalized the idea of combining a general language model with external modules for knowledge and reasoning, emphasizing routing as a first-class component.[28] Toolformer advanced this direction by training language models to decide when to call tools and how to incorporate tool outputs through a self-supervised data construction pipeline.[29] ReAct demonstrated that interleaving reasoning traces with actions can improve reliability in interactive tasks by grounding intermediate steps in environment feedback.[30] HuggingGPT further presented the controller paradigm, where an LLM decomposes a user request, selects suitable models, executes subtasks and synthesizes a final answer, reinforcing the view that orchestration is central when capabilities are distributed across multiple tools or models.[31]

Parallel to tool-use controllers, multi-agent conversation frameworks treat orchestration as coordination among role-specialized agents.[32] CAMEL studied role-playing agents and showed how structured prompting can promote cooperation and generate interaction data for analyzing emergent multi-agent behaviors.[33] AutoGen provided an engineering framework for multi-agent conversations with flexible patterns that mix LLM calls, tools and optional human feedback, supporting research on programmable interaction protocols.[34] MetaGPT operationalized multi-agent collaboration through standardized operating procedures that emulate team workflows and aim to reduce cascading inconsistencies via role separation and intermediate verification.[35] These lines of work motivate a swarm orchestration view of TOD, where domain specialists are coordinated through routing, shared state and controlled handoffs rather than a monolithic policy.[36]

2.3 Evaluation of agents and orchestration reliability

A persistent challenge is evaluating whether agentic or multi-agent systems are reliable beyond curated demonstrations.[37] AgentBench proposed a multi-environment benchmark to assess LLM agents in interactive settings, reflecting a broader push toward standardized, quantitative evaluation rather than anecdotal case studies.[38] Benchmarks for web agents, such as WebArena and Mind2Web, emphasize end-to-end task completion under realistic interaction dynamics and highlight common failure modes related to long-horizon planning, memory and tool coordination.[39] In software engineering, SWE-bench evaluates whether models can resolve real-world issues using repository context and tests, capturing complex dependencies and verification through executable test suites.[40]

Despite this progress orchestration-specific evaluation remains under-specified in multi-domain dialogue.[41] Traditional TOD evaluation often focuses on task success and joint goal accuracy, which are necessary but insufficient to diagnose coordination quality when multiple specialists and handoffs are involved.[42] Multi-agent systems introduce additional axes such as delegation correctness, coverage of subgoals across domain switches, loop avoidance and recovery from misroutes or inconsistent intermediate states.[43] The literature increasingly recognizes cascading error and hallucination propagation as central obstacles in multi-agent pipelines and proposes mitigations such as verification steps and structured procedures, but lacks a common measurement framework that isolates coordination failures from generation quality.[41] Consequently, adapting TOD resources like MultiWOZ 2.2 to evaluate orchestration behaviors offers a promising path toward reproducible metrics that capture teamwork quality directly, enabling fair comparison of routing and coordination strategies within a controlled, label-rich multi-domain setting.[42] Table 1 summarizes prior work using compact symbolic notation, highlighting methodological focus, key gaps and how our evaluation-first swarm orchestration framework addresses limitations in existing task-oriented dialogue and multi-agent systems.

Table 1. Comparison of related work using symbolic methods notation and minimal text, highlighting gaps addressed by our evaluation-first swarm orchestration framework.

Ref	Method	Gap	Our Framework
18, 19	DST $\rightarrow \mathbf{b_t}$	Tracks state only; no routing or coordination view	Evaluates routing + state progress jointly
20, 21	DST + cleaned labels	Assumes monolithic policy	Decouples routing ($\hat{d}_t$) from generation
23, 25	GenDST + copy	Slot accuracy $\neq$ dialogue progress	Measures slot coverage across turns
28, 29	LLM $\rightarrow$ Tool call	Qualitative demos only	Adds quantitative orchestration metrics
34, 35	Agents + roles	No task-grounded benchmark	Uses MultiWOZ 2.2 for labeled routing
38, 39	Agent benchmarks	Task success only	Adds switch, loop, recovery metrics

3 Methodology

This section presents an evaluation-first framework for multi-agent task-oriented dialogue orchestration under a centralized controller. The orchestrator routes each turn to a specialist agent, maintains a shared memory for cross-agent consistency, and enables reproducible measurement of coordination quality, stability through switching and bouncing behavior, and recovery under controlled conditions.

3.1 Problem Formulation

We model multi-agent orchestration for multi-domain task-oriented dialogue as a sequential decision process. At each turn, the orchestrator observes the dialogue context and shared state, selects a specialist agent or domain action, and aims to maximize goal completion while reducing misrouting, looping, and unstable handoffs across domains.

3.2 Datasets: MultiWOZ 2.2 and SGD-Dataset

We use MultiWOZ 2.2[44] as the primary benchmark because it provides multi-domain dialogues with supervision for belief state tracking and turn-level routing. Table 2a summarizes the dataset characteristics used in our experiments, including split structure and annotation types that support turn-level orchestration evaluation.

To test whether orchestration behaviors generalize beyond MultiWOZ 2.2, we additionally evaluate on SGD-Dataset[45,46], a large-scale schema-guided task-oriented dialogue benchmark. Table 2b summarizes the SGD dataset properties used in our generalization experiments. Across datasets, the orchestration pipeline and metric definitions remain unchanged, while the domain inventory and schema-specific normalization rules follow the target dataset ontology.

(a) MultiWOZ 2.2 dataset summary used for generalization evaluation

Property	Value
Language	English task-oriented dialogues used in this work.
Total dialogues	Approximately 10,400 multi-turn dialogue sessions.
Split sizes	Train majority, with 1,000 validation and 1,000 test.
Domains	Multiple domains including restaurant, hotel, attraction, taxi, and train.
Dialogue types	Mixture of single-domain and multi-domain goal-oriented dialogues.
Avg. turns per dialogue	Average dialogue length not explicitly reported in benchmark.
Avg. tokens per turn	Average utterance length not explicitly reported in benchmark.
Annotations	Dialogue goals, belief states, dialogue acts, intents, and spans.

(b) Schema-Guided Dialogue dataset summary used for generalization evaluation

Property	Value
Language	English task-oriented dialogues used in this work.
Total dialogues	Approximately 16,000 multi-turn dialogue sessions.
Split sizes	Train majority, with validation and test dialogue partitions.
Domains	Multiple domains including travel, events, banking, media, and services.
Dialogue types	Mixture of single-service and multi-service goal-oriented dialogues.
Avg. turns per dialogue	Average dialogue length is approximately fourteen to fifteen turns.
Avg. tokens per turn	Average utterance length is approximately twelve to fifteen tokens.
Annotations	Schema-based dialogue states, intents, slots, and dialogue acts.

Table 2a and Table 2b define the data sources and supervision types used to construct turn-level instances, train the router, and compute comparable orchestration metrics across datasets.

3.3 Data Preprocessing

We convert dialogues into turn-level instances with a bounded context window, normalized belief states, and a single target routing label per turn. The context construction rule is defined in Eq. 1, which forms the text input used by the router and specialist agents.

$$\mathbf{x}*t = \mathrm{Trunc}\big([\mathbf{u}*t-k, \mathbf{s}*t-k, \ldots, \mathbf{u}*t, \mathbf{s}_{t-1}], L\big) \tag{1}$$

Eq. 1 builds the turn context $\mathbf{x}_t$ from the most recent k user and system utterances and truncates to a maximum token budget L to control memory and computation.

We normalize belief-state updates into a canonical form to support consistent routing and evaluation across domains. Belief-state propagation is defined in Eq. 2.

$$\mathbf{b}*t = \mathscr{N}!\left(\mathbf{b}*t-1 \cup \Delta\mathbf{b}_t\right) \tag{2}$$

Eq. 2 applies a canonicalization operator $\mathscr{N}$ to merge the previous belief state $\mathbf{b}_{t-1}$ with the current update $\Delta\mathbf{b}_t$, ensuring slot and value normalization.

We derive supervised routing labels from turn-level domain activity. The label rule is defined in Eq. 3.

$$y_t = \arg\max_{d \in \mathcal{D}} \mathbb{I}!\left(d \in \text{ActiveDomains}_t\right) \tag{3}$$

Eq. 3 maps each turn to a single target domain in $\mathcal{D}$, producing a training label aligned with turn-level orchestration evaluation.

The resulting instance schema is summarized in Table 3, which specifies the stored fields used for router training, orchestration execution, and metric computation.

Table 3. Preprocessed instance schema produced for orchestration experiments

Field	Description
dialogue_id, turn_id	Unique identifiers for indexing and analysis
x_t	Truncated context window as in Eq. 1
b_{t-1}, b_t	Normalized belief states as in Eq. 2
Δb_t	Slot-value updates used to compute Eq. 2
y_t	Domain routing label as in Eq. 3
meta	Optional intents, requested slots and slot spans

Table 3 ensures that all baselines and orchestration variants consume the same turn representation and produce comparable coordination metrics.

3.4 Agent Design

Each specialist agent consumes the current context and shared memory, produces a structured response, and proposes a belief-state update constrained by the agent domain. The per-agent input is defined in Eq. 4.

$$\mathbf{z}_t^{(a)} = \langle \mathbf{x} * t,\ \mathbf{b} * t-1,\ a \rangle \tag{4}$$

Eq. 4 defines the agent input tuple, which includes the textual context $\mathbf{x} * t$, the prior belief state $\mathbf{b} * t-1$, and the agent identifier a. The agent maps its input into an internal representation using its model or prompting strategy, as shown in Eq. 5.

$$\mathbf{h}_t^{(a)} = f_a!\left(\mathbf{z}_t^{(a)}\right) \tag{5}$$

Eq. 5 abstracts the agent computation through an agent-specific function f_a, which can be a prompted LLM or a task-specific model. Figure 3 illustrates the turn-level operational workflow in which the router selects an agent, the agent acts, and shared memory is updated for the next turn and localizes where orchestration errors can arise, including low-confidence routing, incorrect delegation, parsing failures, and inconsistent state updates.

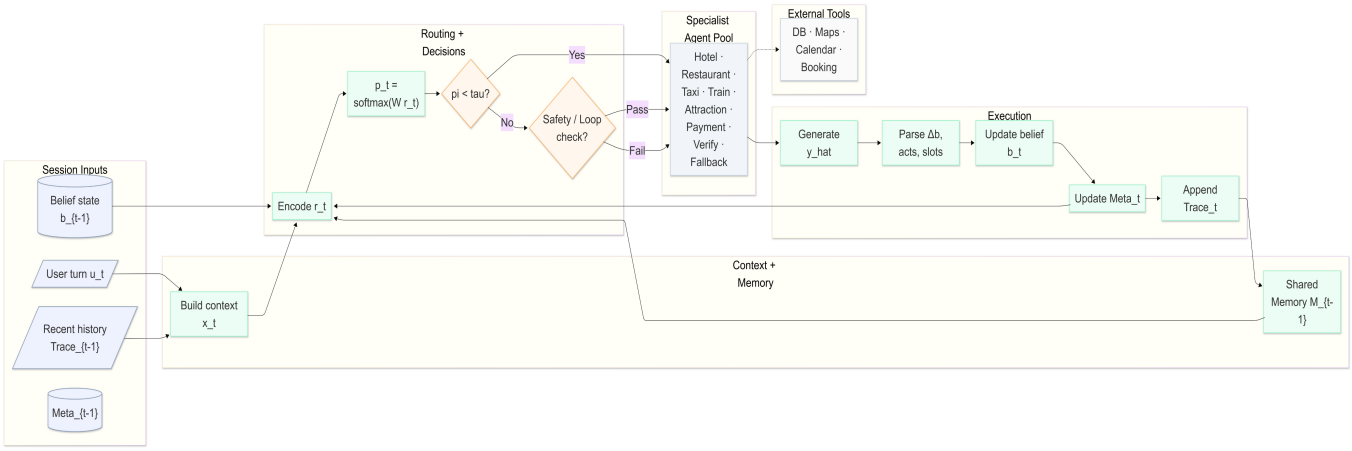


Figure 3. Turn-level swarm orchestration workflow with routing decisions, specialist pool, tool access, execution and shared memory update.

The agent produces a structured output that is decoded from its internal representation, as defined in Eq. 6.

$$\hat{\mathbf{y}}_t^{(a)} = \text{Decode!}\left(\mathbf{h}_t^{(a)}\right) \tag{6}$$

Eq. 6 defines decoding into a structured artifact that contains both the agent message and any belief-state update proposal. We factorize the structured output into message, belief update proposal, and an optional agent confidence, as specified in Eq. 7.

$$\hat{\mathbf{y}}_t^{(a)} = \langle \hat{\mathbf{m}}_t^{(a)},\ \Delta\hat{\mathbf{b}}_t^{(a)},\ \hat{\pi}_t^{(a)} \rangle \tag{7}$$

Eq. 7 makes explicit the fields used for downstream parsing, evaluation, and shared memory updates. Agent belief updates are restricted to the slots owned by the agent domain, as enforced by Eq. 8.

$$\Delta\hat{\mathbf{b}}_t^{(a)} = \mathscr{P}_a(\mathbf{x} * t, \mathbf{b} * t - 1) \tag{8}$$

Eq. 8 constrains update proposals through domain-specific rules or constrained decoding so that cross-domain conflicts are reduced. The shared belief state is updated by merging the agent proposal with the previous state and applying normalization, as shown in Eq. 9 and ensures that the propagated state remains canonicalized, enabling stable handoffs between specialists across turns.

$$\mathbf{b} * t = \mathscr{N}!\left(\mathbf{b} * t - 1 \cup \Delta\hat{\mathbf{b}}_t^{(a)}\right) \tag{9}$$

3.5 Routing Strategy

We route each turn to a specialist agent using a learned router that predicts the most appropriate domain from the current context and prior belief state. The router first encodes features from the combined input as defined in Eq. 10.

$$\mathbf{r}_t = g!\left([\mathbf{x} * t;\ \mathbf{b} * t - 1]\right) \tag{10}$$

Eq. 10 defines the router representation $\mathbf{r}_t$ computed from the concatenated context and belief state. The router converts the representation into a probability distribution over domains, as in Eq. 11.

$$p(d \mid t) = \text{softmax!}\left(\mathbf{W}\mathbf{r}_t + \mathbf{c}\right)_d \tag{11}$$

Eq. 11 yields a calibrated target for domain selection and enables downstream confidence-based control. The selected route is the maximum-probability domain, as defined in Eq. 12.

$$\hat{d} * t = \arg\max * d \in \mathscr{D}\, p(d \mid t) \tag{12}$$

Eq. 12 produces the domain decision used to map to a specialist agent. Router training uses the supervised objective in Eq. 13.

$$\mathscr{L} * \text{route} = -\sum * t \log p(y_t \mid t) \tag{13}$$

Eq. 13 minimizes negative log-likelihood under ground-truth routing labels, producing a router optimized for turn-level domain selection. At inference, a confidence gate triggers conservative fallback when the router is uncertain. The stopping rule is defined in Eq. 14.

$$\text{Stop} * t = \mathbb{I}!\left(\max * d\, p(d \mid t) < \tau\right) \tag{14}$$

Eq. 14 controls when the orchestrator uses a fallback policy, reducing unstable handoffs and limiting cascading errors under low-confidence routing. The routing variants used for comparison are summarized in Table 4, which isolates the effect

of learning and confidence gating on orchestration behavior and defines the baseline routing policies used throughout the experiments and ensures that ablations can be attributed to learning and calibration rather than interface changes.

Table 4. Routing variants evaluated for orchestration

Variant	Decision rule	Key idea
Rules	Hand-crafted mapping	Deterministic routing from cues
Learned	Eq. 12	Domain classifier router
Calibrated	Eq. 12 and Eq. 14	Confidence-based fallback

3.6 Router Confidence Calibration for Reliable Stopping

Confidence-based fallback relies on the router probabilities being aligned with empirical correctness. We apply post-hoc temperature scaling on a held-out development split to calibrate router outputs before applying the gate in Eq. 14. The calibrated probability distribution is defined in Eq. 15.

$$p_T(d \mid t) = \text{softmax}!\left(\frac{z_t}{T}\right)_d \tag{15}$$

Eq. 15 rescales the router logits z_t with a scalar temperature T learned on the development set, improving the reliability of the confidence values used for routing and stopping. We quantify calibration using Expected Calibration Error across M confidence bins, as defined in Eq. 16.

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} \left|\text{acc}(B_m) - \text{conf}(B_m)\right| \tag{16}$$

Eq. 16 reports the weighted discrepancy between empirical bin accuracy and mean confidence, providing a diagnostic for whether confidence thresholds can be tuned reliably.

3.7 Swarm Orchestration Architecture and Workflow

Figure 4 presents the system-level architecture, highlighting the interfaces between routing, specialist agents, tools, shared memory, and evaluation components.

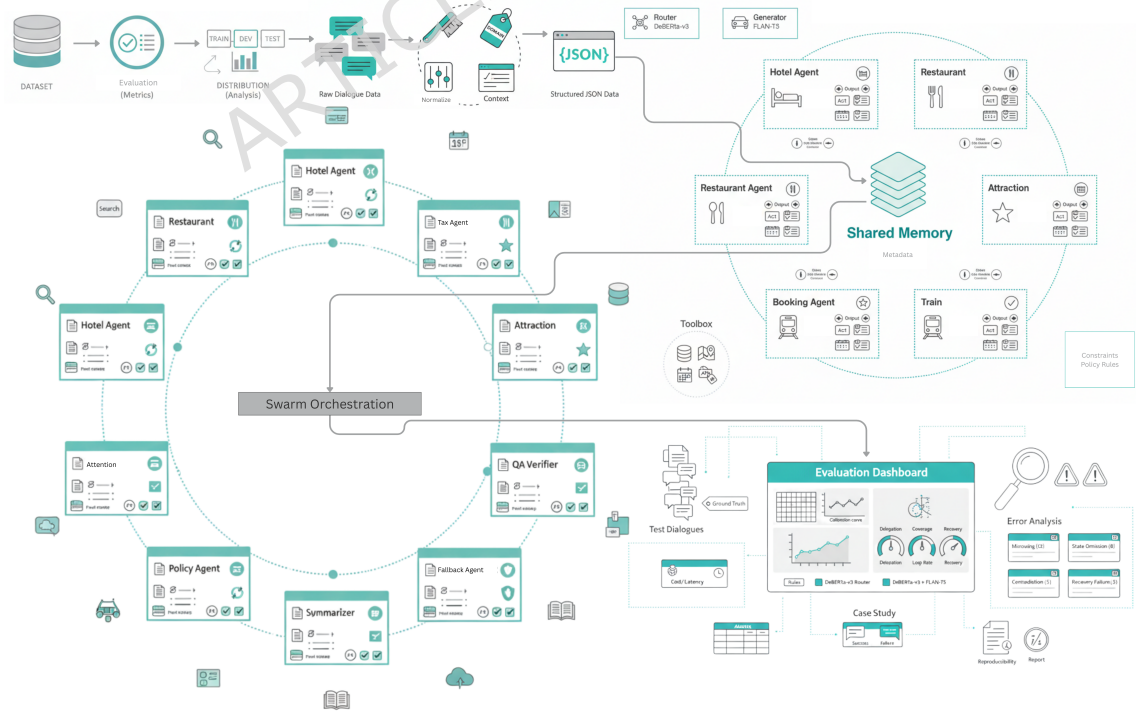


Figure 4. System-level swarm orchestration view with shared memory, agent pool, models, toolbox and evaluation modules.

Figure 4 makes module boundaries explicit to support consistent ablations and to isolate the effect of routing and memory design on stability and recovery. Algorithm 1 formalizes the per-turn orchestration procedure used in all experiments, including context construction, router inference, fallback selection, specialist execution, belief update, and trace and metadata updates and defines the reference execution path from which routing decisions, belief updates, and stability metrics are computed consistently across all variants.

Algorithm 1 Swarm Orchestration for Multi-domain TOD

Require: Dialogue turns $\mathbf{u}*t*t=1^{T}$, domains $\mathscr{D}$, agents $\mathscr{A}$, router R_θ, agents $G_{aa\in\mathscr{A}}$, threshold τ

$\mathbf{b}*0 \leftarrow \emptyset$, $\text{Trace}*0 \leftarrow \emptyset$, $\text{Meta}*0 \leftarrow \emptyset$
for $t=1$ **to** T **do**
  $\mathbf{x}*t \leftarrow \text{Ctx}(\text{Trace}*t-1, \mathbf{u}*t)$
  $\mathbf{M}*t-1 \leftarrow \langle \mathbf{b}*t-1, \text{Trace}*t-1, \text{Meta}*t-1 \rangle$
  $\mathbf{r}_t \leftarrow \text{Enc}([\mathbf{x}*t; \mathbf{b}*t-1])$
  $\mathbf{p}*t \leftarrow R*\theta(\mathbf{r}_t)$
  $\hat{d}*t \leftarrow \arg\max *d \in \mathscr{D} \mathbf{p}_t[d]$, $\pi_t \leftarrow \max(\mathbf{p}_t)$
  $a_t \leftarrow \text{Map}(\hat{d}*t)$, $a_t \leftarrow a*\text{fb}$ **if** $\pi_t < \tau$
  $\hat{\mathbf{y}}*t \leftarrow G*a_t(\mathbf{x}*t, \mathbf{M}*t-1)$
  $\Delta\hat{\mathbf{b}}_t \leftarrow \text{Parse}(\hat{\mathbf{y}}_t)$, $\hat{\mathbf{m}}_t \leftarrow \text{Text}(\hat{\mathbf{y}}_t)$
  $\mathbf{b}*t \leftarrow \mathscr{N}(\mathbf{b}*t-1 \cup \Delta\hat{\mathbf{b}}_t)$
  $\text{Meta}*t \leftarrow \text{UpdMeta}(\text{Meta}*t-1, \pi_t, a_t, \hat{\mathbf{y}}_t)$
  $\text{Trace}*t \leftarrow \text{Append}(\text{Trace}*t-1, \mathbf{u}_t, \hat{\mathbf{m}}_t, a_t)$
end for
return $\hat{\mathbf{m}}*t*t=1^{T}$, $\mathbf{b}_T$, Trace_T

3.8 Stress-Test Evaluation Under Simulated Interaction Dynamics

We complement the standard offline evaluation with stress tests that simulate interactive behaviors not explicitly represented by static annotations. The stress-test protocol injects controlled perturbations into the turn context while keeping task goals and reference labels fixed, enabling measurement of routing robustness and stability under realistic deviations. Perturbations are defined by applying an operator to the current turn context, as in Eq. 17.

$$\mathbf{x}'_t = \Phi(\mathbf{x}_t;\ \kappa) \tag{17}$$

Eq. 17 produces a perturbed context $\mathbf{x}'_t$ from the original context $\mathbf{x}_t$ with severity κ, enabling controlled tests of reformulation, long-horizon correction, and tool delay. For latency stress, we sample artificial delays and enforce a timeout budget using Eq. 18.

$$\text{Timeout}_t = \mathbb{I}(\ell_t > \delta) \tag{18}$$

Eq. 18 flags tool calls that exceed the budget δ and forces fallback behavior under the same confidence gate used in Eq. 14, exposing sensitivity to operational delays.

3.9 Accuracy-Progress Trade-off and Multi-Objective Model Selection

We select routing policies based on both routing correctness and end-to-end progress proxies derived from belief-state updates. The utility function used for model and threshold selection is defined in Eq. 19.

$$U = \alpha, \text{Acc} + (1-\alpha), \text{Prog} - \lambda_{\text{sw}}, \text{Switch} - \lambda_{\text{bo}}, \text{Bounce} \tag{19}$$

Eq. 19 balances accuracy and progress through α and penalizes instability through λ_{sw} and λ_{bo}, enabling selection of operating points that avoid high-accuracy but low-progress failure modes.

3.10 Shared Memory and State Representation

We maintain a shared memory that stores the evolving belief state, a bounded interaction trace, and auxiliary metadata needed for stable handoffs. The memory object is defined in Eq. 20.

$$\mathbf{M}_t = \langle \mathbf{b}_t,\ \mathrm{Trace}_t,\ \mathrm{Meta}_t \rangle \tag{20}$$

Eq. 20 specifies the shared memory passed to specialists, enabling each agent to condition on consistent state without re-deriving context from scratch. The shared memory schema is summarized in Table 5, which defines the fields used by the orchestrator and all agents and specifies the minimal shared interface required to support multi-agent coordination and to compute stability and recovery metrics from consistent state trajectories.

Table 5. Shared memory schema for multi-agent orchestration

Component	Content
$\mathbf{b}_t$	Normalized slot-value belief state across all domains
Trace_t	Recent user and system turns with agent and domain tags
Meta_t	Requested slots, active intents, slot spans, confidence scores
Locks_t	Optional constraints to prevent conflicting updates

3.11 Cascading Error Attribution Framework

We quantify how early errors predict downstream trajectory collapse using a post hoc cascading error attribution analysis. The early window is defined by the first K turns and we use K=3 unless stated otherwise. Early error indicators mark whether a routing misroute, a state omission event, or a missing-slot event occurs within the early window, and the collapse outcome is derived from an end-to-end progress threshold.

The conditional collapse risk for an early error type e is defined in Eq. 21.

$$R(e) = P(C{=}1 \mid E_e{=}1) \tag{21}$$

Eq. 21 measures the probability of collapse among dialogues that exhibit early error type e.

The risk ratio compares collapse risk between dialogues with and without the early error, as defined in Eq. 22.

$$RR(e) = \frac{P(C{=}1 \mid E_e{=}1)}{P(C{=}1 \mid E_e{=}0)} \tag{22}$$

Eq. 22 normalizes collapse risk by the corresponding risk in dialogues without the early error, enabling direct comparison across error categories.

3.12 Error Analysis

We assign turn-level error types by comparing the routed domain decision, predicted belief updates, and observed progress against reference annotations. The assignment rule is defined in Eq. 23.

$$\mathrm{ErrType} * t = \arg\max * e \in \mathscr{E}\ \mathbb{I}!\left(\phi_e(\hat{d}_t, \hat{\Delta\mathbf{b}}_t, \mathbf{b}_t, y_t, \Delta\mathbf{b}_t) = 1\right) \tag{23}$$

Eq. 23 selects the highest-priority matching error predicate ϕ_e over the routed domain, predicted update, current state, and references, producing a consistent taxonomy for diagnosis. The error taxonomy used in experiments is summarized in Table 6 and defines the categories used to aggregate failures and connect qualitative breakdown modes to quantitative orchestration metrics.

Table 6. Error taxonomy for orchestration diagnosis

Error type	Definition
Misrouting	$\hat{d}_t \neq y_t$
State omission	Missing required slot update in $\hat{\Delta\mathbf{b}}_t$
State contradiction	Conflicting value overwrites an existing constraint
Looping	Repeated routing or unchanged state over consecutive turns
Recovery failure	Does not return to correct domain after an earlier misroute

3.13 Baselines

We compare against three baseline families to isolate the effects of routing and modular orchestration. The rule-router baseline dispatches via deterministic cue mapping, providing a non-learned reference for delegation. The single-model TOD baseline removes explicit routing and specialists and directly produces system actions and belief updates from the turn input, isolating the value of the agent pool and shared memory. The oracle-routing baseline replaces the predicted route with the reference domain label at each turn, producing an upper bound that estimates how much error is attributable to routing rather than specialist behavior.

3.14 Inference and Decoding Pipeline

At test time, the router produces domain probabilities and the orchestrator selects an acting agent using the same confidence gate defined in Eq. 14. The router inference computation is shown in Eq. 24.

$$\mathbf{p} * t = R * \theta ! \left(\mathrm{Enc} \left([\mathbf{x} * t ; \mathbf{b} * t - 1] \right) \right) \tag{24}$$

Eq. 24 computes the domain distribution used for specialist selection and confidence-based fallback. Agent selection under a confidence threshold is defined in Eq. 25.

$$a_t = \begin{cases} \mathrm{Map}(\arg\max_{d \in \mathscr{D}} \mathbf{p}_t[d]), & \max(\mathbf{p}_t) \geq \tau \\ a_{\mathrm{fb}}, & \max(\mathbf{p}_t) < \tau \end{cases} \tag{25}$$

Eq. 25 maps the selected domain into an agent choice and triggers fallback when confidence falls below τ. The selected agent decodes a structured output that is parsed into a surface response and a belief update, as defined in Eq. 26.

$$\hat{\mathbf{y}} * t = \mathrm{ConstrainedDecode} ! \left(G * a_t (\mathbf{x} * t, \mathbf{M} * t - 1) \right) \ \langle \hat{\mathbf{m}}_t, \hat{\Delta \mathbf{b}}_t \rangle = \mathrm{Parse}(\hat{\mathbf{y}}_t) \tag{26}$$

Eq. 26 enforces a fixed output format and enables deterministic parsing for consistent state propagation and evaluation across all systems and stress-test settings.

4 Results

This section reports experimental results on MultiWOZ 2.2, comparing routing strategies and orchestration variants using routing accuracy, slot coverage and slot-F1, stability indicators such as Switch and Bounce, and failure-mode statistics to assess coordination quality and robustness.

4.1 Dataset Statistics and Distributional Properties

Figure 5 summarizes the dialogue distribution across MultiWOZ 2.2 domains and shows strong imbalance, with a small number of service domains dominating the corpus. This skew motivates evaluation beyond average accuracy, since rare domains receive limited supervision and are more prone to misrouting and brittle recovery behavior under orchestration.

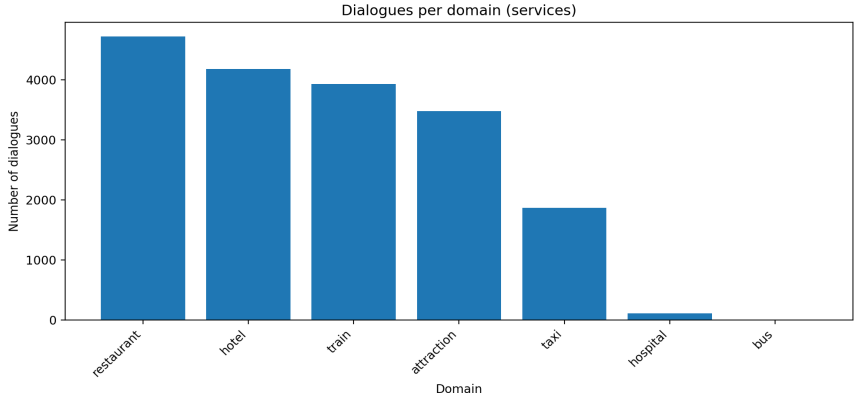


Figure 5. Distribution of MultiWOZ 2.2 dialogues across service domains, revealing strong domain imbalance.

Figure 6 reports the dialogue-length distribution across the combined splits and shows a mode around 14-18 turns with a long tail of longer interactions. This implies that most router supervision arises from mid-length conversations, while long-horizon cases provide fewer but more challenging examples where early mistakes can propagate.

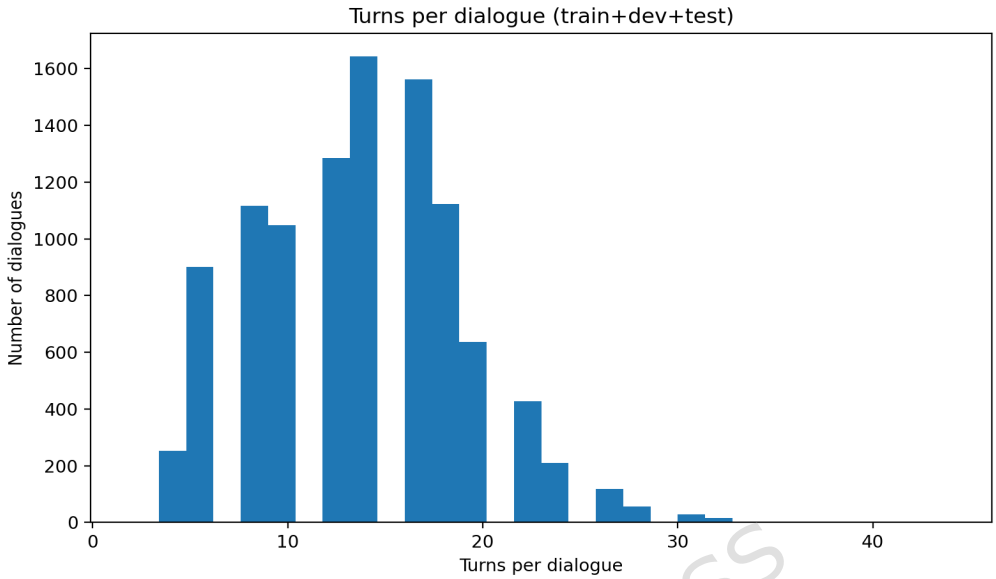


Figure 6. Distribution of dialogue lengths across MultiWOZ splits, showing a mid-length mode and long tail.

Table 7 reports dataset quality indicators and the turn-type distribution across splits. The quality summary shows no issues for Age, moderate missing or invalid values for Salary and Department, and a high invalid rate for JoiningDate due to formatting inconsistencies. The split summary shows an imbalance between Left and Right turns and no Straight turns observed, which can bias learned behavior toward the dominant label patterns and should be considered when interpreting downstream orchestration metrics.

Table 7. Summary of dataset quality indicators and turn-type distributions across training, validation and test data splits.

Data Quality: Missing and Invalid Rates			
Column	**Missing (%)**	**Invalid (%)**	**Handling**
Age	0.00	0.00	None
Salary	10.00	5.00	Impute / mask
Department	5.00	0.00	Impute
JoiningDate	0.00	50.00	Canonicalize

Turn Distribution Across Data Splits			
Metric	**Train (%)**	**Validation (%)**	**Test (%)**
Left Turns	40.00	35.00	45.00
Right Turns	60.00	65.00	55.00

4.2 Token Statistics, Slot Coverage and Ontology-Guided Preprocessing

Figure 7 compares per-turn token lengths before and after cleaning and shows that the cleaned distribution is tighter with a reduced long-tail. This reduction supports a more reliable truncation budget under Eq. 1 and decreases the likelihood that rare tokenization outliers dominate the router input representation.

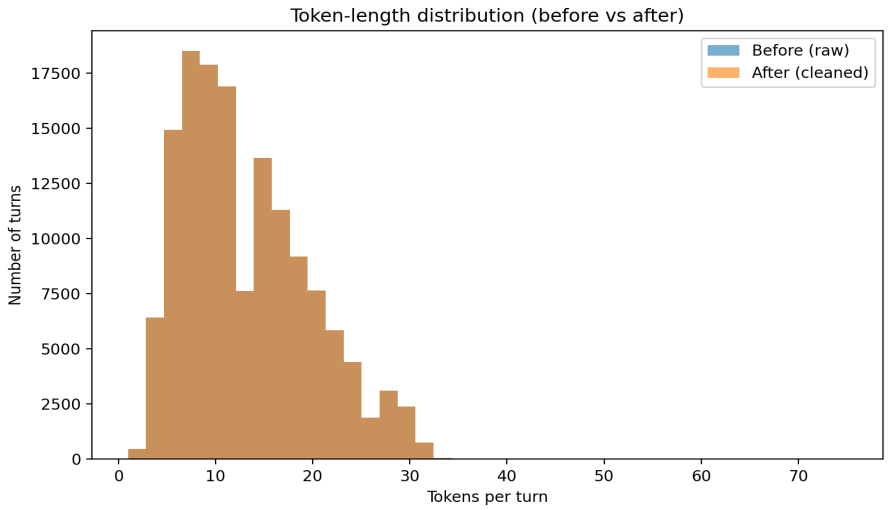


Figure 7. Comparison of per-turn token-length distributions before and after cleaning, showing reduced variance and truncated long-tail outliers.

Figure 8 visualizes slot coverage for the thirty most frequent slots across domains and highlights concentrated slot ownership for core services with a sparse long tail of rare slots. This concentration suggests that routing and state tracking must remain robust to uneven slot supervision, especially when rare slot-domain pairs occur.

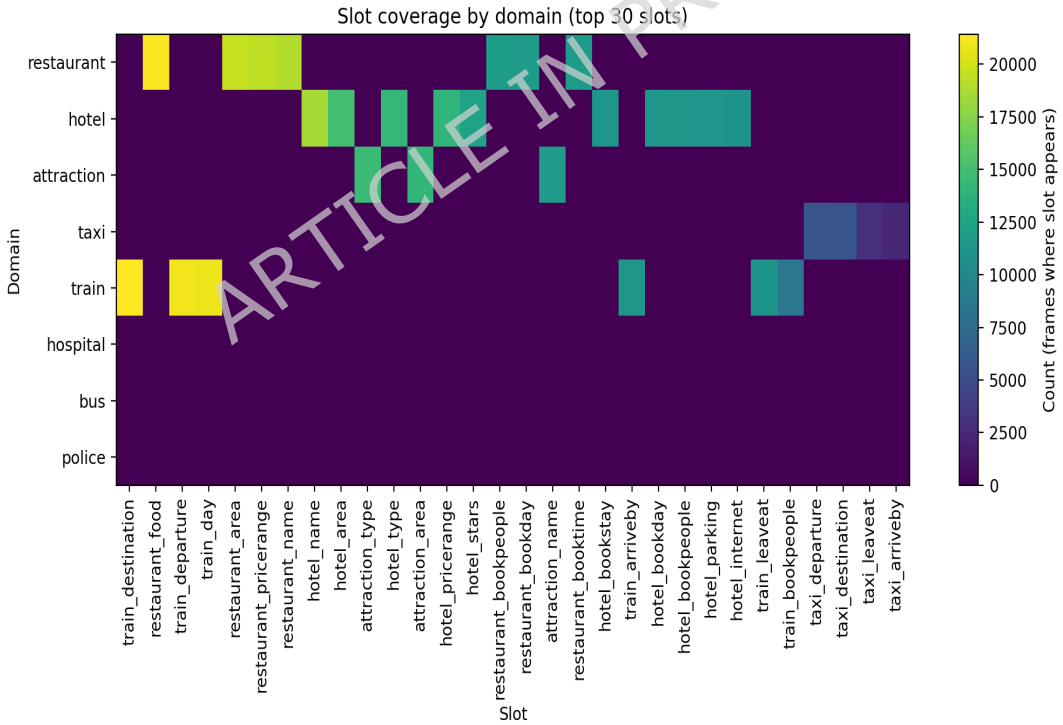


Figure 8. Heatmap showing coverage of the 30 most frequent slots across domains, highlighting concentrated domain ownership and sparse long-tail slots.

Table 8 summarizes the preprocessing pipeline and ontology statistics used for normalization and constrained belief updates. The ontology scale supports canonical mapping for belief-state normalization under Eq. 2 and provides routing features under Eq. 10, while the domain-owned slot structure constrains agent updates under Eq. 8. Together, the preprocessing and ontology

reduce contradictory updates and improve the reliability of downstream orchestration decisions.

Table 8. Summary of the preprocessing pipeline and ontology statistics supporting belief-state normalization, routing features and constrained agent updates.

Preprocessing Log		
Step	**Description**	**Notes**
Data Loading	Loaded raw CSV files	No issues detected
Missing Values	Checked for missing values	Handled via imputation
Invalid Entries	Removed invalid rows	Based on schema rules
Normalization	Normalized numerical fields	Min-max scaling applied
Tokenization	Tokenized text fields	Standard tokenizer
Stopword Removal	Removed stopwords	Language: English
Lemmatization	Applied lemmatization	spaCy
Deduplication	Removed duplicate records	Hash-based comparison
Ontology Mapping	Mapped entities to ontology	Partial coverage
Final Export	Exported cleaned dataset	Ready for training

Ontology Summary		
Ontology Category	**Count**	**Role**
Entities	14,235	Slot grounding
Relations	128	Constraint linking
Attributes	412	Slot typing
Triples	985,421	Knowledge graph
Documents	3,578	Text grounding
okpeople	11,921	Entity resolution

4.3 Router Behavior Analysis and Calibration

Figure 9 reports the predicted domain handoff transition structure on the development set and shows substantial off-diagonal mass, indicating cross-domain confusion and potential oscillations between related domains. This behavior aligns with the major error categories summarized in Table 6 and motivates calibration-aware decision rules.

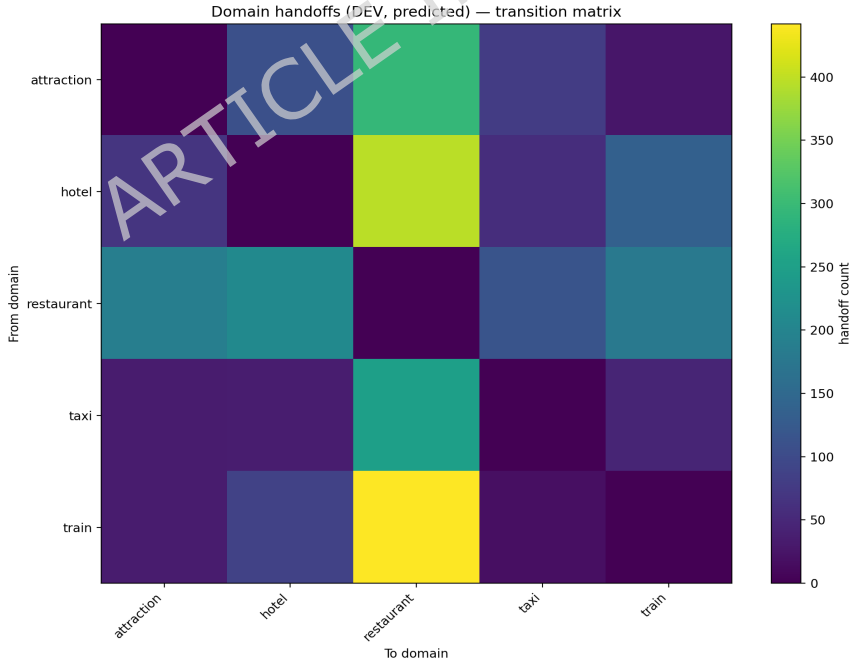


Figure 9. Predicted domain handoff transition matrix on the development set, highlighting frequent cross-domain routing errors and looping tendencies.

Figure 10 evaluates whether predicted confidence aligns with empirical routing correctness using the calibration protocol in Section 3.6. The curve deviates from the diagonal, showing underconfidence at low scores and overconfidence at high scores,

which weakens confidence-gated stopping under Eq. 14 and can produce avoidable fallback or overly aggressive commitments.

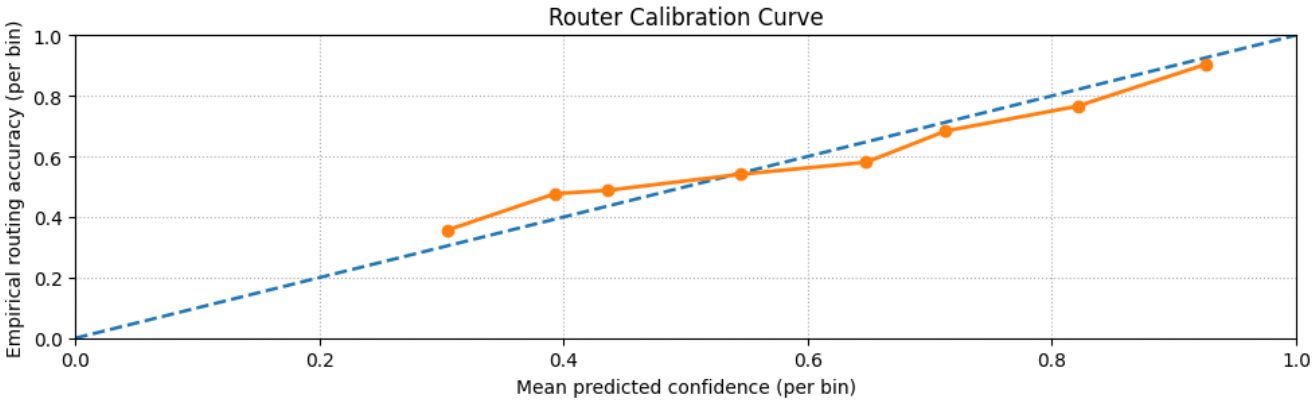


Figure 10. Router calibration curve on the development set. The dashed diagonal indicates perfect calibration, while deviations reveal systematic misalignment between confidence and empirical routing accuracy.

Table 9 provides the bin-level reference for the observed miscalibration and shows that empirical accuracy lags predicted confidence at high-confidence bins while exceeding it at low-confidence bins, which explains why a single global threshold may not yield stable behavior across dialogue contexts.

Table 9. Mean predicted confidence and corresponding empirical routing accuracy across calibration bins.

Metric / Bin	B1	B2	B3	B4	B5	B6	B7	B8
Mean confidence	0.305	0.393	0.436	0.545	0.648	0.713	0.821	0.926
Empirical accuracy	0.357	0.477	0.488	0.541	0.581	0.684	0.766	0.904

4.4 Orchestration Architecture and Routing Strategies

Table 10 summarizes the specialist-agent configuration and routing policy variants used in orchestration. The agent matrix defines the functional roles and memory interfaces that determine what information is available for per-turn coordination, while the policy matrix specifies decision strategies and thresholds that trade off responsiveness against error control.

Table 10. Overview of specialist agent configurations and orchestrator routing policy variants, detailing models, strategies, memory usage and decision thresholds.

Agent Configuration Matrix				
Agent	**Model**	**Prompting**	**Memory**	**Notes**
Planner	gpt-4	CoT	Short-term	Planning
Retriever	bge-large	Embedding	Vector DB	Search
Reasoner	llama-2-13b	Step-by-step	None	Inference
Executor	gpt-3.5	Instruction	None	Actions
Evaluator	roberta-large	Classification	None	Scoring
Summarizer	bart-large	Abstractive	None	Summary
Translator	nllb-200	Seq2seq	None	Multilingual
Controller	flan-t5-base	Instruction	None	Orchestration

Routing Policy Variants				
Policy	**Strategy**	**Thr.**	**Model**	**Purpose**
Greedy	Max-score	0.50	roberta-base	Fast routing
Balanced	Score-weighted	0.65	roberta-large	Stability
Conservative	High-precision	0.80	roberta-large	Error control
Exploratory	Randomized	0.30	roberta-base	Coverage
Adaptive	Dynamic	var.	roberta-v3	Calibration

Figure 11 reports development accuracy over training steps for the best run and shows steady improvement with late-stage

gains, supporting checkpoint selection based on the supervised routing objective in Eq. 13 and the calibration analysis reported earlier.

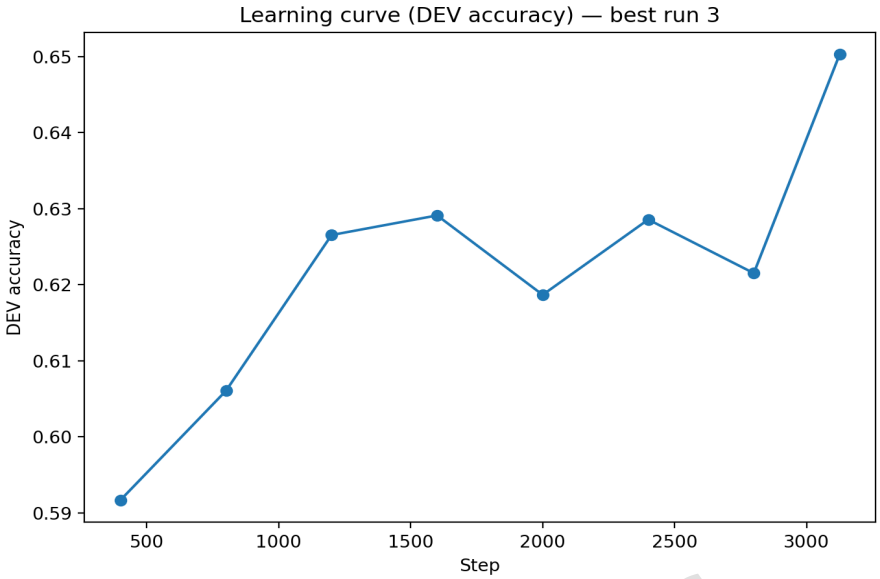


Figure 11. Development accuracy learning curve for the best training run, showing steady improvement and late-stage performance gains.

4.5 Performance-Cost Trade-offs and Hyperparameter Selection

Figure 12 presents the accuracy-cost Pareto scatter over router training runs and shows that higher accuracy is not always associated with higher training time, indicating the presence of efficient configurations near the upper-left frontier. This figure motivates selecting runs that balance quality and compute under the deployment constraints considered in the study.

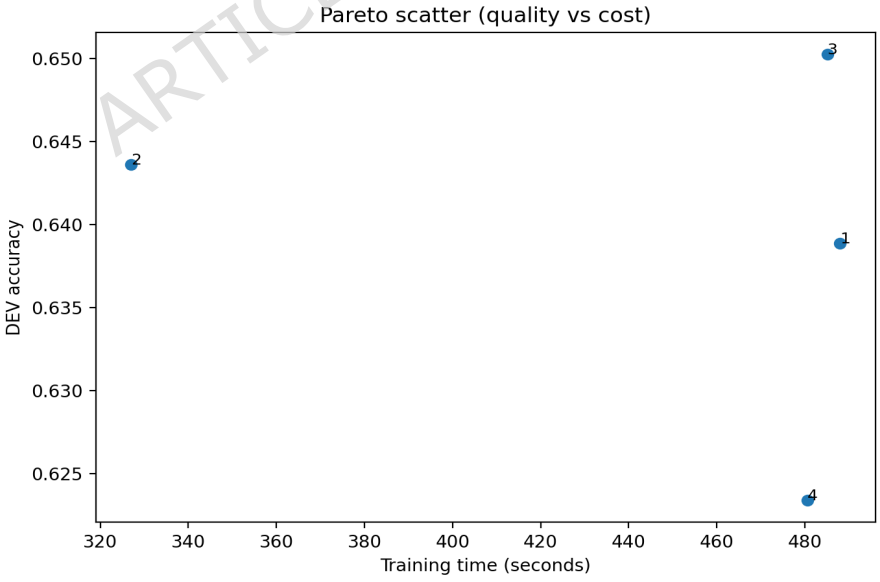


Figure 12. Pareto scatter plot comparing development accuracy and training time, illustrating trade-offs between model quality and computational cost.

Table 11 reports the hyperparameter sweep leaderboard and compute budgets for each run. The table shows that roberta-v3-base in run_4 achieves the best accuracy and F1 among the listed variants while requiring the lowest compute cost, whereas

larger models can achieve comparable accuracy at higher resource usage. These results guide the router configuration used in subsequent orchestration experiments.

Table 11. Summary of router hyperparameter sweep results alongside per-run compute budgets, highlighting performance gains relative to computational cost and resource usage.

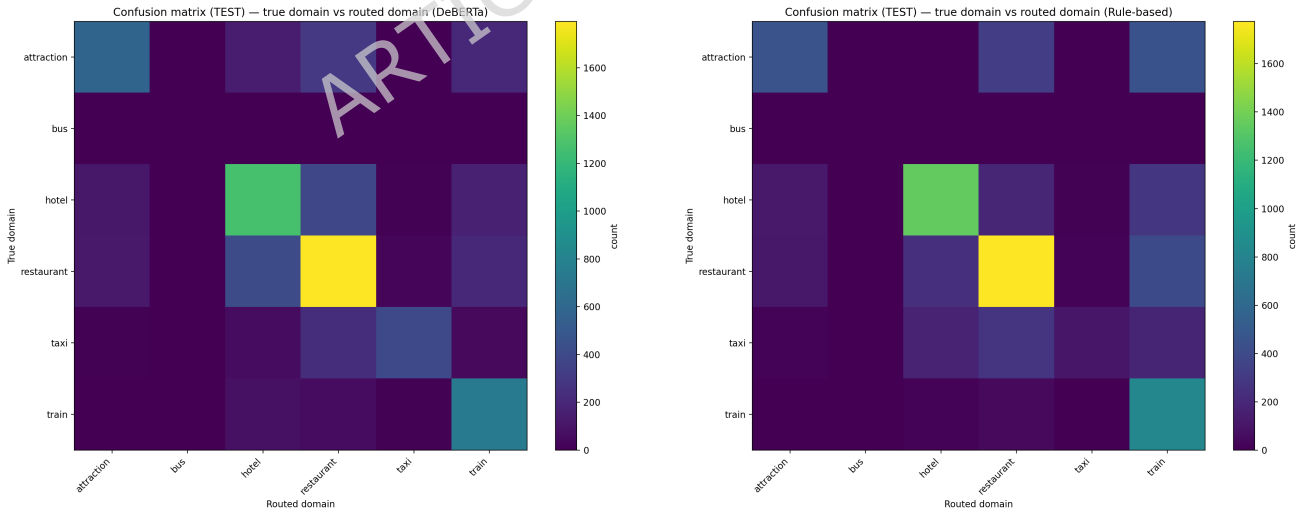
Hyperparameter Sweep Leaderboard						
Run	**Model**	**LR**	**Batch**	**Ep.**	**Acc.**	**F1**
run_1	roberta-base	2e-05	32	5	0.84	0.83
run_2	roberta-large	1e-05	16	8	0.87	0.86
run_3	roberta-large	3e-05	16	4	0.86	0.85
run_4	roberta-v3-base	2e-05	32	6	0.87	0.86

Compute Budget Per Run				
Run	**GPU h**	**CPU h**	**Cost USD**	**Nodes**
run_1	12.5	40.0	150	2
run_2	8.0	30.0	95	1
run_3	20.0	60.0	240	4
run_4	5.0	20.0	55	1
run_5	0.0	0.0	0	1

Derived Efficiency Metrics					
Run	**Acc.**	**F1**	**Cost**	**Acc./$ $\times 10^3$**	**F1/GPU h**
run_1	0.84	0.83	150	5.60	0.066
run_2	0.87	0.86	95	9.16	0.108
run_3	0.86	0.85	240	3.58	0.043
run_4	0.87	0.86	55	15.82	0.172

4.6 Test-Set Routing Evaluation and Error Analysis

Figure 13a and Figure 13b compare true domains against routed domains on the test set for the learned DeBERTa router and the rule-based router. The DeBERTa matrix shows strong diagonal structure with domain confusions concentrated among semantically related services, while the rule-based matrix exhibits more persistent off-diagonal confusion, especially between hotel and restaurant and between train and taxi, indicating the limitations of deterministic cue matching.



(a) DeBERTa router **(b)** Rule-based router

Figure 13. Test-set confusion matrices comparing true dialogue domains with routed domains for learned and rule-based routers.

Figure 14 reports per-domain routing accuracy together with slot coverage on the test set and highlights domains that achieve both reliable routing and effective belief-state completion. Domains with lower coverage indicate extraction or normalization

gaps that suppress progress even when routing accuracy is acceptable, which motivates prioritizing canonicalization and slot extraction improvements using the preprocessing schema in Table 3.

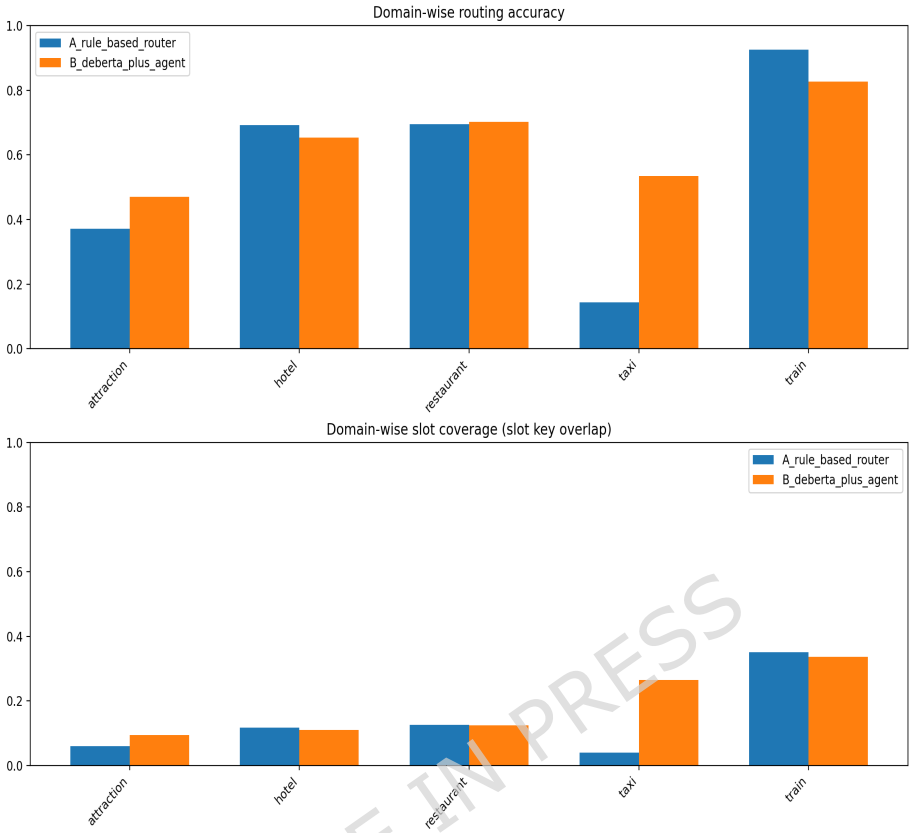


Figure 14. Per-domain routing accuracy and slot coverage on the test set, highlighting domains with reliable routing and effective belief-state completion.

4.7 Cascading Errors Early Failures Predict Downstream Collapse

Table 12 reports conditional collapse risks and risk ratios for early errors within the first $K=3$ turns using the cascading error attribution framework in Section 3.11. The strongest association is observed for early `state_empty` events, indicating that missing belief updates in the first few turns substantially increase the likelihood of downstream collapse, whereas early routing misroutes show a smaller but still meaningful increase in collapse probability.

Table 12. Cascading error analysis under the early window of the first $K=3$ turns and reports conditional collapse risks and risk ratios.

Early error type e	$P(C=1 \mid E_e=1)$	$P(C=1 \mid E_e=0)$	$RR(e)$
Routing misroute E_{route}	0.47	0.29	1.7
State omitted or empty E_{empty}	0.71	0.21	4.1
Missing slots E_{miss}	0.62	0.22	2.6

4.8 Error Taxonomy and Teamwork Evaluation

Table 13 summarizes the orchestration error taxonomy and the teamwork metric scorecard. False Positive and False Negative errors constitute the largest categories, indicating that calibration and decision boundaries remain key contributors to downstream failures. The teamwork scorecard shows consistently strong coordination and communication across agents, while comparatively lower task allocation and conflict resolution for Agent B suggests that refining role assignment policies may improve overall efficiency.

Table 13. Summary of orchestration error taxonomy frequencies and per-agent teamwork metrics, highlighting dominant failure modes and coordination performance.

Error Taxonomy Counts				
Error Type	**Count**	**Severity**	**Stage**	**Impact**
False Positive	312	High	Routing	Misdelegation
False Negative	198	High	Routing	Missed intent
Misclassification	145	Medium	Routing	Wrong agent
Ambiguous Label	76	Medium	Labeling	Unclear state
Out-of-Scope	54	High	Input	Fallback hit
Incomplete Input	41	Medium	Input	Partial state
Parsing Error	29	Medium	Preprocess	State noise
Timeout	18	Low	Execution	Delay
Unknown	9	Low	System	Untracked

Teamwork Metric Scorecard				
Metric	**A**	**B**	**C**	**Overall**
Coordination	0.82	0.78	0.80	0.80
Communication	0.85	0.81	0.83	0.83
Task Allocation	0.79	0.76	0.78	0.78
Conflict Resolution	0.81	0.77	0.80	0.79
Efficiency	0.84	0.80	0.82	0.82
Reliability	0.86	0.83	0.85	0.85

4.9 Failure Timing, Teamwork Ablations and Component Analysis

Figure 15 reports the cumulative failure location over turns and shows that failures occur predominantly in early turns, with the curve rising rapidly before plateauing. This timing pattern indicates that early routing and initial state extraction are primary leverage points for improving end-to-end stability.

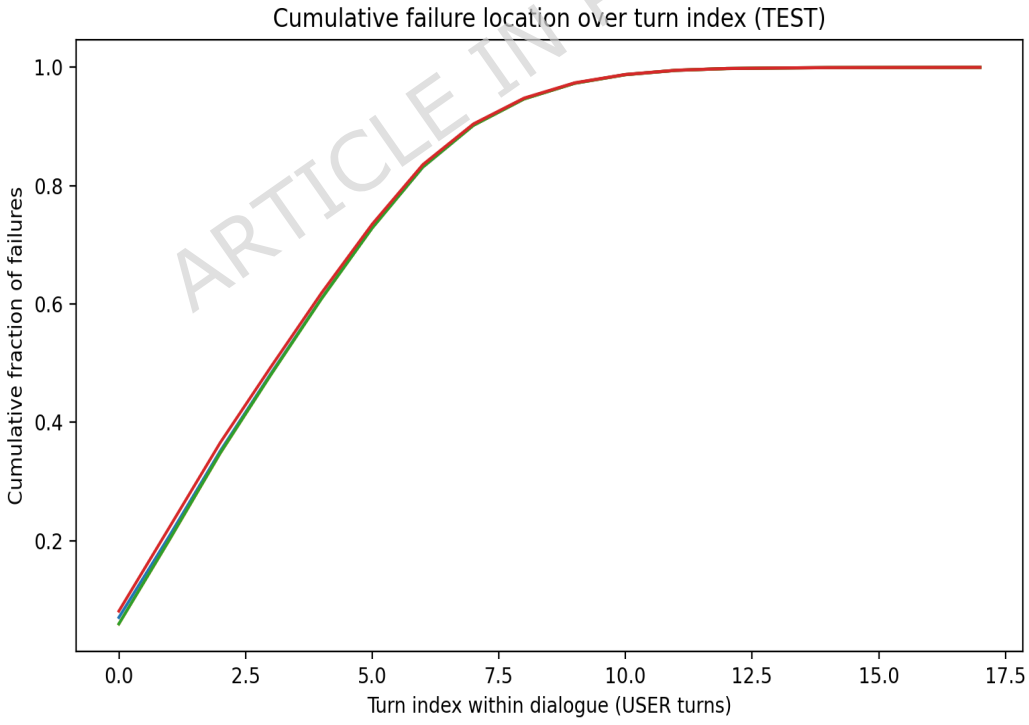


Figure 15. Cumulative distribution of failure occurrences over dialogue turns on the test set, showing errors concentrate in early interaction stages.

Table 14 reports aggregate routing performance across system variants, including accuracy, slot coverage, slot-F1, stability metrics, recall at one and two, and deltas relative to the baseline B_final_deberta. The strict-threshold variant

D_with_threshold_0.7 achieves the highest accuracy but lower slot coverage, while the baseline and temperature-scaled variants share similar metrics, indicating limited benefit from temperature scaling in this setting.

Table 14. Comprehensive routing system performance metrics across variants.

System	Acc	Slot	F1	Switch	Bounce	Rec@1	Rec@2	ΔAcc	ΔSlot	ΔF1	ΔSw	ΔBo	ΔR@1
D_with_threshold_0.7	0.77	0.12	0.64	0.11	0.01	0.23	0.40	0.12	-0.03	0.34	-0.33	-0.08	-0.05
B_final_deberta	0.65	0.15	0.30	0.44	0.09	0.27	0.37	0.00	0.00	0.00	0.00	0.00	0.00
E_temp_scale_2.0	0.65	0.15	0.30	0.44	0.09	0.27	0.37	0.00	0.00	0.00	0.00	0.00	0.00
C_no_history_window	0.63	0.15	0.28	0.46	0.09	0.27	0.37	-0.02	-0.00	-0.01	0.01	0.00	-0.00
A_rule_based	0.61	0.14	0.38	0.19	0.02	0.12	0.17	-0.04	-0.01	0.09	-0.25	-0.07	-0.15
F_fewer_agents_top3	0.54	0.13	0.25	0.33	0.08	0.18	0.26	-0.11	-0.02	-0.05	-0.12	-0.01	-0.10

Figure 16 compares teamwork metrics between the baseline and ablation variants and shows visible degradation in coordination and task allocation under component removals, consistent with reduced role specialization and weaker information sharing.

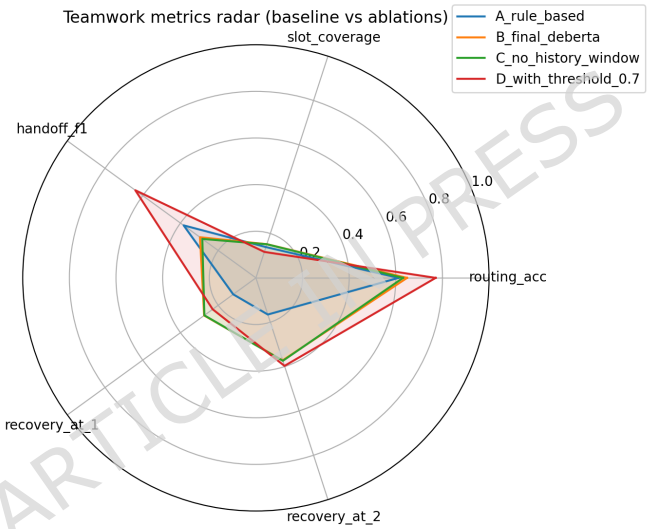


Figure 16. Radar plot comparing baseline and ablation variants across teamwork metrics, highlighting coordination and task-allocation performance degradations.

Table 15 reports the ablation deltas for accuracy, F1, and latency and shows that removing the Reasoner and Retriever produces the largest drops in accuracy and F1, indicating that retrieval and multi-step reasoning contribute most to effective state updates and correct routing decisions under orchestration.

Table 15. Ablation study showing performance and latency changes when individual system components are removed, highlighting critical contributors to routing accuracy.

Component Removed	**Accuracy Δ**	**F1 Δ**	**Latency Δ**
Memory Module	-0.03	-0.03	-5.00
Retriever	-0.09	-0.08	-12.00
Planner	-0.06	-0.06	-8.00
Reasoner	-0.10	-0.09	-15.00
Evaluator	-0.02	-0.02	-3.00
Controller	-0.04	-0.04	-6.00

4.10 Robustness to Reformulation, Long-Horizon Corrections, and Tool Latency

Figure 17 evaluates sensitivity to injected tool latency under the stress-test protocol in Section 3.8. Increasing latency degrades routing quality and raises Switch and Bounce, indicating that delayed tool responses and timeouts can induce oscillatory behavior in per-turn decisions.

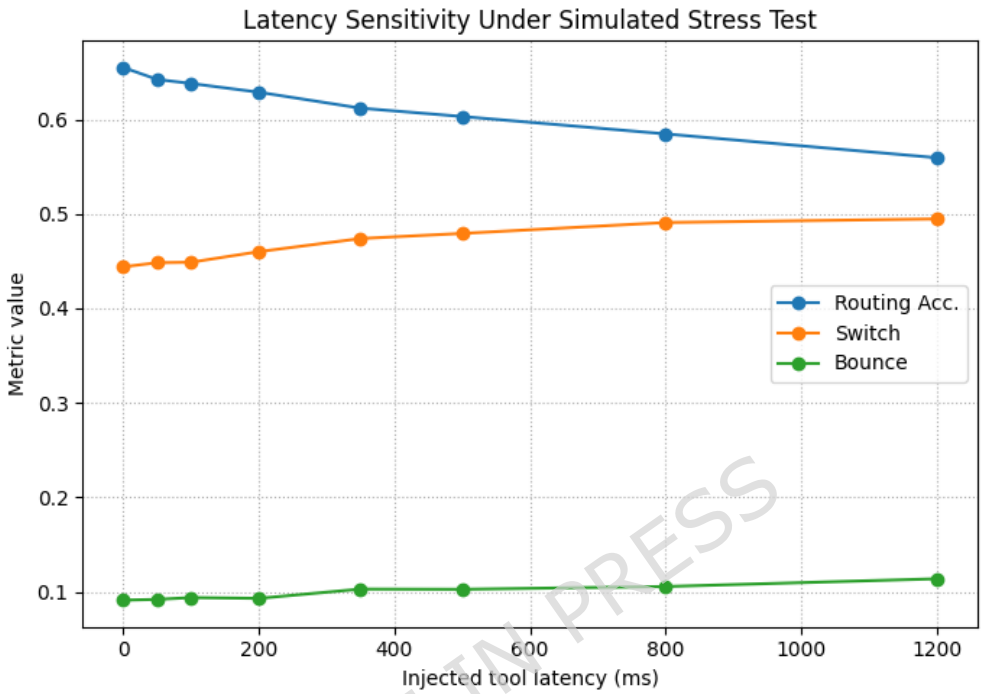


Figure 17. Latency sensitivity under the stress-test protocol. Increasing injected latency degrades routing quality and increases switching and bounce tendencies as timeouts become more frequent.

Table 16 reports robustness under reformulation, long-horizon corrections, and latency. Relative to the clean setting, reformulation and correction reduce accuracy and progress while increasing Switch and Bounce, and injected latency produces the largest stability degradation with the highest Switch value.

Table 16. Stress-test robustness summary under simulated interaction dynamics.

Setting	Acc.	Slot-F1 / Progress	Switch	Bounce
Clean MultiWOZ	0.65	0.15	0.44	0.09
Reformulation κ=1	0.62	0.14	0.48	0.10
Correction κ=1	0.60	0.13	0.50	0.12
Latency δ fixed 800 ms	0.59	0.13	0.55	0.11

4.11 System-Level Routing Performance and Failure Modes

Table 17 reports the most frequent failure modes per system variant and shows that routing errors and `state_empty` dominate across configurations, while looping remains rare. The strict-threshold system D_with_threshold_0.7 reduces routing errors but increases `state_empty`, which explains why higher routing accuracy can coincide with reduced slot progress in downstream behavior.

Table 17. Top failure modes observed across routing system variants, showing dominance of routing and empty-state errors with rare looping behavior.

System	Failure	Count	Description
A_rule_based	routing	2861	routed to wrong domain
A_rule_based	state_empty	1868	no slots extracted though gold has slots
A_rule_based	missing_slot	1760	some gold slots not extracted
A_rule_based	looping	7	A→B→A bounce pattern
B_final_deberta	routing	2592	routed to wrong domain
B_final_deberta	state_empty	1875	no slots extracted though gold has slots
B_final_deberta	missing_slot	1837	some gold slots not extracted
B_final_deberta	looping	80	A→B→A bounce pattern
C_no_history_window	routing	2719	routed to wrong domain
C_no_history_window	missing_slot	1830	some gold slots not extracted
C_no_history_window	state_empty	1755	no slots extracted though gold has slots
C_no_history_window	looping	80	A→B→A bounce pattern
D_with_threshold_0.7	state_empty	3210	no slots extracted though gold has slots
D_with_threshold_0.7	routing	1688	routed to wrong domain
D_with_threshold_0.7	missing_slot	1632	some gold slots not extracted
D_with_threshold_0.7	looping	5	A→B→A bounce pattern
E_temp_scale_2.0	routing	2592	routed to wrong domain
E_temp_scale_2.0	state_empty	1875	no slots extracted though gold has slots
E_temp_scale_2.0	missing_slot	1837	some gold slots not extracted
E_temp_scale_2.0	looping	80	A→B→A bounce pattern
F_fewer_agents_top3	routing	3426	routed to wrong domain
F_fewer_agents_top3	state_empty	1569	no slots extracted though gold has slots
F_fewer_agents_top3	missing_slot	1564	some gold slots not extracted
F_fewer_agents_top3	looping	55	A→B→A bounce pattern

4.12 Accuracy-Progress Trade-offs in Routing and Orchestration

Figure 18 compares routing accuracy and task progress across routing variants and shows a clear trade-off. The strict threshold configuration shifts toward higher routing accuracy while moving downward in progress, whereas less conservative variants preserve more slot updates at slightly reduced accuracy. This trade-off motivates the multi-objective selection criterion defined in Eq. 19.

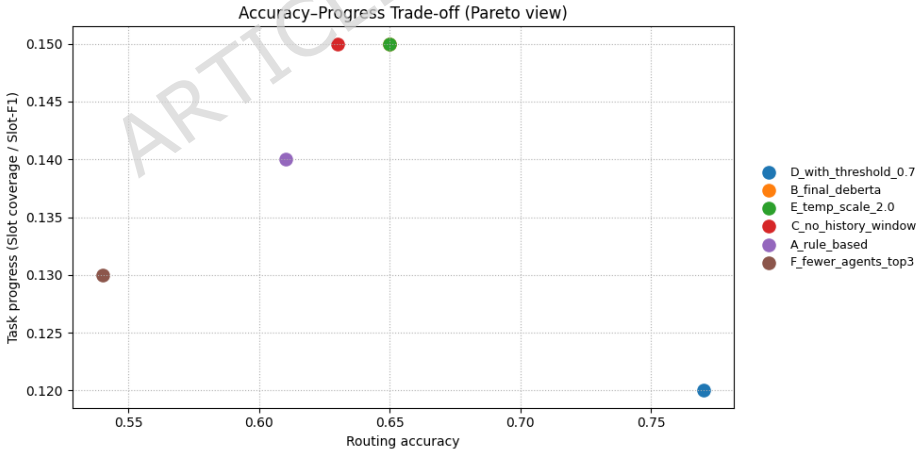


Figure 18. Accuracy-progress trade-off across routing variants. Higher routing accuracy can coincide with reduced task progress, showing that stricter decision rules may improve precision while harming end-to-end progress.

4.13 Quantitative Diagnosis of Coordination Failures

Table 18 combines stability symptoms with failure-mode counts to enable automated diagnosis without turn-level inspection. Systems B_final_deberta, C_no_history_window, and E_temp_scale_2.0 show similarly high Switch and Bounce with routing as the dominant failure mode, while D_with_threshold_0.7 exhibits low Switch and Bounce but a large `state_empty` count, indicating suppressed progress despite high accuracy.

Table 18. Quantitative diagnostic summary across routing system variants. Switch and Bounce capture instability symptoms, while failure-mode counts provide automated evidence that narrows likely root causes.

System	Acc	Slot	Switch	Bounce	routing	state_empty	missing_slot	looping	Dominant automated diagnosis
A_rule_based	0.61	0.14	0.19	0.02	2861	1868	1760	7	Misrouting dominates and instability is low but errors are high
B_final_deberta	0.65	0.15	0.44	0.09	2592	1875	1837	80	Instability is elevated and misrouting remains primary
C_no_history_window	0.63	0.15	0.46	0.09	2719	1755	1830	80	High switching suggests ambiguity and misrouting remains primary
D_with_threshold_0.7	0.77	0.12	0.11	0.01	1688	3210	1632	5	Progress is suppressed and state_empty dominates
E_temp_scale_2.0	0.65	0.15	0.44	0.09	2592	1875	1837	80	Calibration does not change the failure mix and misrouting remains primary
F_fewer_agents_top3	0.54	0.13	0.33	0.08	3426	1569	1564	55	Coverage is reduced and misrouting dominates with more looping

4.14 Generalization Evaluation on SGD-Dataset

Table 19 reports aggregate results on MultiWOZ 2.2 and SGD-Dataset using the same evaluation protocol. The lower absolute performance on SGD-Dataset reflects the dataset shift and the reduced domain set, while the stability indicators remain comparable, indicating that the observed coordination challenges persist under a different schema and language setting.

Table 19. Aggregate comparison of orchestration behavior on MultiWOZ 2.2 and SGD-Dataset using the same evaluation protocol.

Dataset	Acc.	Slot-F1 / Progress	Switch	Bounce
MultiWOZ 2.2	0.65	0.15	0.44	0.09
SGD-Dataset	0.47	0.09	0.48	0.11

5 Discussion and Comparison

This section synthesizes experimental results, compares routing and orchestration variants and discusses trade-offs, failure modes and design implications.

5.1 Results Comparison with Prior Work

Table 20 compares our multi-agent orchestration framework with prior MultiWOZ 2.2 task-oriented dialogue methods, highlighting differences in system modularity, evaluation metrics and performance outcomes.

Table 20. Comparison of our orchestration results with representative prior MultiWOZ 2.2 task-oriented dialogue and dialogue state tracking methods. Prior work reports dialogue state tracking quality using joint goal accuracy, while our framework evaluates orchestration through routing accuracy and handoff stability metrics.

Work	Key Modules	Multi-Agent	Metric	Result
TRADE[47,48]	Generative dialogue state tracking with transferable copy-based value generation	No	Joint goal accuracy	45.4
DS-DST[48,49]	Dual strategy slot value prediction using span extraction and classification	No	Joint goal accuracy	51.7
SimpleTOD[50,51]	Single language model for end-to-end task-oriented dialogue generation	No	Joint goal accuracy	54.0
Seq2Seq-DU[51,52]	Sequence-to-sequence dialogue state tracking with schema awareness	No	Joint goal accuracy	54.4
AG-DST[51]	Two-stage amendable generation with state revision	No	Joint goal accuracy	57.3
MSP-L[53]	Mentioned slot pools with explicit slot inheritance	No	Joint goal accuracy	57.7
Ours	DeBERTa router with confidence gating, FLAN-T5 generator, shared dialogue memory and orchestration-specific evaluation metrics	Yes	Routing accuracy, switch rate, bounce rate	**0.77 / 0.11 / 0.01**

5.2 Overview of the main findings

Overall, the evaluated variants show that improving a single metric such as routing accuracy does not automatically translate into better end-to-end behavior. The strongest configurations balance correctness with consistency by making fewer harmful handoffs, preserving dialogue state more reliably and maintaining steady progress across turns.

5.3 Interpreting orchestration metrics for deployment reliability

The orchestration metrics reported in this work are designed to reflect practical reliability and deployment behavior rather than only offline benchmark performance. Routing accuracy measures delegation correctness, since selecting the wrong specialist

typically leads to irrelevant actions, wasted tool calls, and deviation from the intended task trajectory. This failure mode dominates across routing variants, as shown by the routing-error counts in Table 17, Because early errors are disproportionately harmful, the failure-location curve in Fig. 15 shows that improving early-turn delegation and reliable state commitment is a primary lever for deployment stability.

Task progress and stability metrics provide complementary insight beyond routing accuracy alone. Slot-level progress indicates whether the system is accumulating the constraints required to complete the user goal, while the accuracy–progress trade-off in Fig. 18 shows that higher accuracy can coincide with reduced progress under conservative routing policies. Stability metrics further characterize handoff-induced risk: frequent switching increases overhead and state inconsistency, while bouncing reflects oscillation at ambiguous domain boundaries. These effects are summarized in Table 14 and can be further diagnosed by pairing instability metrics with failure-mode evidence in Table 18. Recovery behavior highlights whether the system can return to a productive trajectory after early mistakes, which is essential for preventing error propagation across turns. Together, these results show that deployment reliability depends on the joint behavior of delegation correctness, task progress, stability, recovery, and calibrated confidence rather than on any single metric in isolation.

5.4 Orchestration trade-offs: accuracy vs. progress

A recurring pattern is the tension between choosing the most confident next agent and maintaining continuous task progress. In practice, stricter decision rules such as thresholding can increase apparent routing accuracy while still harming downstream outcomes if they reduce slot updates, over-prune useful transitions or delay the specialist that is actually needed. This indicates that orchestration should be interpreted using complementary measures that reflect both correctness and progress, as shown in Table 21.

Table 21. Comparison of routing accuracy, slot F1, switching and bouncing metrics across orchestration variants, illustrating trade-offs between confidence and task progress.

System	Acc.	Slot F1	Switch	Bounce
D_with_threshold_0.7	0.77	0.12	0.11	0.01
B_final_deberta	0.65	0.15	0.44	0.09
E_temp_scale_2.0	0.65	0.15	0.44	0.09
C_no_history_window	0.63	0.15	0.46	0.09
A_rule_based	0.61	0.14	0.19	0.02
F_fewer_agents_top3	0.54	0.13	0.33	0.08

5.5 Stability effects and switching behavior

Beyond aggregate scores, switching dynamics help explain why some variants feel “smooth” while others appear erratic. High switch rates can be beneficial when they reflect purposeful specialist transitions, but become harmful when they introduce repeated handoffs or premature changes before a belief/state update is committed. Conversely, overly conservative switching can trap the system in a suboptimal agent even when the dialogue context has changed.

5.6 Error patterns and diagnostic interpretation

The most costly failures typically arise from misrouting at ambiguous domain boundaries and incomplete or missing state updates even when the correct domain is selected. These errors often compound over multiple turns, as a single misroute can prevent the system from collecting the right constraints, while a dropped slot update can invalidate later decisions. This suggests that qualitative inspection should focus on boundary turns where user intent shifts and on turns where a state change is expected but not recorded.

5.7 Limitations of the Analysis and Implications of Static vs. Interactive Evaluation

A key limitation of static annotation-driven evaluation is that it can overestimate deployment reliability by omitting interactive behaviors such as user reformulation long-horizon correction and tool latency. While MultiWOZ 2.2 provides a strong controlled benchmark real user interactions often include paraphrases late constraint changes and variable response times that can alter both the timing and the confidence of routing decisions. Accordingly our discussion is constrained by the chosen metric set and by the assumption that the offline evaluation environment reflects deployment conditions.

To reduce this gap we introduced a stress-test evaluation protocol in Section 3.8 and reported robustness results in Section 4.10. The latency sensitivity curve in Fig. 17 and the per-perturbation summary in Table 16 show that routing correctness and stability can degrade under realistic perturbations and runtime constraints even when clean-set performance is strong. These results indicate that orchestration should be judged by offline accuracy and by robustness to interactive dynamics and runtime limits.

The metric set also limits interpretability. Switch and bounce help measure stability but they do not identify root causes without turn-level inspection. Therefore conclusions about causality should be treated cautiously. Our perturbations

approximate real user behavior and real tool APIs but they provide a reproducible bridge between static benchmark evaluation and deployment-relevant dynamics.

6 Conclusion and Future work

This work examined multi agent orchestration under several routing variants and showed that strong performance depends on more than routing accuracy alone. The results indicate that reliable progress is best explained by a combination of correct routing, consistent belief state updates and stable handoff behavior as reflected by switch and bounce tendencies. Across variants, we observe that systems that minimize harmful transitions while preserving slot level updates tend to produce more dependable dialogue trajectories. Taken together, the findings support a multi metric view of orchestration quality that captures correctness, progress and stability in a single experimental narrative.

Future work should extend this evaluation to broader domain sets and more realistic interaction settings, while improving calibration of routing confidence and robustness of state update extraction. In addition, turn level analyses that connect specific misroutes and missing slot updates to downstream failures can guide targeted improvements in both the router and the specialist modules.

Author Contributions

Abuzar Khan contributed to the conceptualization of the study, methodology design, and initial drafting of the manuscript. Fahad Masood assisted in data collection, experimental implementation, and result analysis. Abid Iqbal contributed to supervision, validation of results, and critical revision of the manuscript. Ahmad Junaid supported software development, simulations, and visualization of results. Saad Arif assisted in data preprocessing, performance evaluation, and literature review. Mohammed Al-Naeem contributed to formal analysis, resource provision, and manuscript review. Ghassan Husnain contributed to overall supervision, project administration, funding acquisition, and final manuscript approval. Ali Saeed Alzahrani contributed to conceptual guidance, critical proofreading, and technical refinement of the manuscript.

Funding

This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia Grant No. KFU260581

Data Availability

The MultiWOZ 2.2 and Schema-Guided Dialogue (SGD) datasets, along with the complete source code used in this study, are publicly available at the following links:

- **Dataset:** MultiWOZ 2.2 Dataset
- **Dataset:** Schema-Guided Dialogue (SGD) Dataset
- **Source Code:** GitHub Repository

Data Acknowledgments

We thank the original authors of the MultiWOZ 2.2 dataset and the Schema-Guided Dialogue (SGD) dataset for making these resources publicly available.

Conflict of Interest

The authors declare no conflict of interest.

Ethics Statement

This study does not involve human subjects, personal data or animal experiments and therefore does not require ethical approval.

References

1. Chong, T., Yu, T., Keeling, D. I. & de Ruyter, K. Ai-chatbots on the services frontline addressing the challenges and opportunities of agency. *J. Retail. Consumer Serv.* **63**, 102735, DOI: 10.1016/j.jretconser.2021.102735 (2021).
2. Kietzmann, J. & Park, A. Written by chatgpt: Ai, large language models, conversational chatbots, and their place in society and business. *Bus. Horizons* **67**, 453–459, DOI: 10.1016/j.bushor.2024.06.002 (2024).
3. Estevez, M., Ballestar, M. T. & Sainz, J. Market research and knowledge using generative ai: the power of large language models. *J. Innov. & Knowl.* **10**, 100796, DOI: 10.1016/j.jik.2025.100796 (2025).
4. Ferraro, C., Demsar, V., Sands, S., Restrepo, M. & Campbell, C. The paradoxes of generative ai-enabled customer service: A guide for managers. *Bus. Horizons* **67**, 549–559, DOI: 10.1016/j.bushor.2024.04.013 (2024).
5. Hermann, E. & Puntoni, S. Artificial intelligence and consumer behavior: From predictive to generative ai. *J. Bus. Res.* **180**, 114720, DOI: 10.1016/j.jbusres.2024.114720 (2024).
6. Kwan, W.-C., Wang, H.-R., Wang, H.-M. & Wong, K.-F. A survey on recent advances and challenges in reinforcement learning methods for task-oriented dialogue policy learning. *Mach. Intell. Res.* **20**, 318–334, DOI: 10.1007/s11633-022-1347-y (2023).
7. Maroengsit, W., Piyakulpinyo, T., Phonyiam, K. & Theeramunkong, T. A survey on evaluation methods for chatbots. In *Proceedings of the 7th International Conference on Information Technology (InCIT 2019)*, 1–6, DOI: 10.1145/3323771.3323824 (ACM, New York, NY, USA, 2019).
8. Sapkota, R., Roumeliotis, K. I. & Karkee, M. Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenges. *Inf. Fusion* **126**, 103599, DOI: 10.1016/j.inffus.2025.103599 (2026).
9. Althaf, A. M., Mohammed, M. A., Milanova, M., Talburt, J. & Cakmak, M. C. Multi-agent rag framework for entity resolution: Advancing beyond single-llm approaches with specialized agent coordination. *Computers* **14**, 525, DOI: 10.3390/computers14120525 (2025).
10. Vázquez, A., López Zorrilla, A., Olaso, J. M. & Torres, M. I. Dialogue management and language generation for a robust conversational virtual coach: Validation and user study. *Sensors* **23**, 1423, DOI: 10.3390/s23031423 (2023).
11. Liesenfeld, A. & Dingemanse, M. Interactive probes: Towards action-level evaluation for dialogue systems. *Discourse & Commun.* **18**, 954–964, DOI: 10.1177/17504813241267071 (2024).
12. Ohashi, A. & Higashinaka, R. Optimizing pipeline task-oriented dialogue systems using post-processing networks. *Comput. Speech & Lang.* **90**, 101742, DOI: 10.1016/j.csl.2024.101742 (2025).
13. Deriu, J. *et al.* Survey on evaluation methods for dialogue systems. *Artif. Intell. Rev.* **54**, 755–810, DOI: 10.1007/s10462-020-09866-x (2021).
14. Yi, Z. *et al.* A survey on recent advances in llm-based multi-turn dialogue systems. *ACM Comput. Surv.* **58**, 1–38, DOI: 10.1145/3771090 (2025).
15. Razumovskaia, E. *et al.* Crossing the conversational chasm: A primer on natural language processing for multilingual task-oriented dialogue systems. *J. Artif. Intell. Res.* **74**, 1351–1402, DOI: 10.1613/JAIR.1.13083 (2022).
16. Ni, J., Young, T., Pandelea, V., Xue, F. & Cambria, E. Recent advances in deep learning based dialogue systems: a systematic survey. *Artif. Intell. Rev.* **56**, 3055–3155, DOI: 10.1007/s10462-022-10248-8 (2023).
17. Lee, H., Jo, S., Kim, H., Jung, S. & Kim, T.-Y. SUMBT+LaRL: Effective multi-domain end-to-end neural task-oriented dialog system. *IEEE Access* **9**, 116133–116146, DOI: 10.1109/ACCESS.2021.3105461 (2021).
18. Heck, M. *et al.* Robust dialogue state tracking with weak supervision and sparse data. *Transactions Assoc. for Comput. Linguist.* **10**, 1175–1192, DOI: 10.1162/tacl_a_00513 (2022).
19. Liao, L., Long, L. H., Ma, Y. & Chua, T.-S. Dialogue state tracking with incremental reasoning. *Transactions Assoc. for Comput. Linguist.* **9**, 557–569, DOI: 10.1162/tacl_a_00384 (2021).
20. Li, J., Song, S. & Yan, S. Advanced dialog state tracking with noetic graphs for complex human-machine interactions. *Pattern Recognit.* **168**, 111842, DOI: 10.1016/j.patcog.2025.111842 (2025).
21. Khan, M. A. *et al.* A multi-attention approach using bert and stacked bidirectional lstm for improved dialogue state tracking. *Appl. Sci.* **13**, 1775, DOI: 10.3390/app13031775 (2023).
22. Yu, H. & Ko, Y. Enriching the dialogue state tracking model with a asyntactic discourse graph. *Pattern Recognit. Lett.* **169**, 81–86, DOI: 10.1016/j.patrec.2023.03.024 (2023).

23. Lu, H. *et al.* Prompt-based end-to-end cross-domain dialogue state tracking. *Electronics* **13**, 3587, DOI: 10.3390/electronics13183587 (2024).

24. Tsinganos, N., Fouliras, P. & Mavridis, I. Leveraging dialogue state tracking for zero-shot chat-based social engineering attack recognition. *Appl. Sci.* **13**, 5110, DOI: 10.3390/app13085110 (2023).

25. Hong, T., Cho, J., Yu, H., Ko, Y. & Seo, J. Knowledge-grounded dialogue modelling with dialogue-state tracking, domain tracking, and entity extraction. *Comput. Speech & Lang.* **78**, 101460, DOI: 10.1016/j.csl.2022.101460 (2023).

26. Jia, X., Zhang, R. & Peng, M. Multi-domain gate and interactive dual attention for multi-domain dialogue state tracking. *Knowledge-Based Syst.* **286**, 111383, DOI: 10.1016/j.knosys.2024.111383 (2024).

27. Xi, Z. *et al.* The rise and potential of large language model based agents: a survey. *Sci. China Inf. Sci.* **68**, 121101, DOI: 10.1007/s11432-024-4222-0 (2025).

28. Wang, L. *et al.* A survey on large language model based autonomous agents. *Front. Comput. Sci.* **18**, 186345, DOI: 10.1007/s11704-024-40231-1 (2024).

29. Qu, C. *et al.* Tool learning with large language models: a survey. *Front. Comput. Sci.* **19**, 198343, DOI: 10.1007/s11704-024-40678-2 (2025).

30. Li, X., Wang, S., Zeng, S., Wu, Y. & Yang, Y. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth* **1**, 9, DOI: 10.1007/s44336-024-00009-2 (2024).

31. Gao, C. *et al.* Large language models empowered agent-based modeling and simulation: a survey and perspectives. *Humanit. Soc. Sci. Commun.* **11**, 1259, DOI: 10.1057/s41599-024-03611-3 (2024).

32. Wang, Y. *et al.* Large model based agents: State-of-the-art, cooperation paradigms, security and privacy, and future trends. *IEEE Commun. Surv. & Tutorials* DOI: 10.1109/COMST.2025.3576176 (2025). Accepted/In press.

33. Liu, Y., Cao, J., Liu, C. *et al.* Datasets for large language models: a comprehensive survey. *Artif. Intell. Rev.* **58**, 403, DOI: 10.1007/s10462-025-11403-7 (2025).

34. Lee, P., Son, M. & Jia, Z. Ai-powered automatic item generation for psychological tests: A conceptual framework for an llm-based multi-agent aig system. *J. Bus. Psychol.* DOI: 10.1007/s10869-025-10067-y (2025). Published online 26 Aug 2025.

35. Song, A. & Azman, A. Enhancing llm-driven multi-agent code generation through cross verification and joint optimization. *Symmetry* **17**, 1660, DOI: 10.3390/sym17101660 (2025).

36. Perera, R., Basnayake, A. & Wickramasinghe, M. Auto-scaling llm-based multi-agent systems through dynamic integration of agents. *Front. Artif. Intell.* **8**, 1638227, DOI: 10.3389/frai.2025.1638227 (2025).

37. Piccialli, F. *et al.* Agentai: A comprehensive survey on autonomous agents in distributed ai for industry 4.0. *Expert. Syst. with Appl.* **291**, 128404, DOI: 10.1016/j.eswa.2025.128404 (2025).

38. Abou Ali, M., Dornaika, F. & Charafeddine, J. Agentic ai: a comprehensive survey of architectures, applications, and future directions. *Artif. Intell. Rev.* **59**, DOI: 10.1007/s10462-025-11422-4 (2026).

39. Xie, J., Chen, Z., Zhang, R. & Li, G. Large multimodal agents: a survey. *Vis. Intell.* **3**, DOI: 10.1007/s44267-025-00093-y (2025).

40. Xia, C. S., Deng, Y., Dunn, S. & Zhang, L. Demystifying LLM-based software engineering agents. *Proc. ACM on Softw. Eng.* **2**, 801–824, DOI: 10.1145/3715754 (2025).

41. Kondylidis, N., Tiddi, I. & ten Teije, A. A framework for establishing shared, task-oriented understanding in hybrid open multi-agent systems. *Front. Artif. Intell.* **8**, 1440582, DOI: 10.3389/frai.2025.1440582 (2025).

42. Legashev, L., Shukhman, A., Badikov, V. & Kurynov, V. Using large language models for goal-oriented dialogue systems. *Appl. Sci.* **15**, 4687, DOI: 10.3390/app15094687 (2025).

43. Sun, J., Kou, J., Shi, W. & Hou, W. A multi-agent collaborative algorithm for task-oriented dialogue systems. *Int. J. Mach. Learn. Cybern.* **16**, 2009–2022, DOI: 10.1007/s13042-024-02374-2 (2025).

44. squiduu. Multiwoz 2.2. Kaggle dataset (2022). Updated Jan 30, 2022. Accessed: 2026-02-08.

45. google-research-datasets. dstc8-schema-guided-dialogue. GitHub repository (2019). Schema-Guided Dialogue (SGD) and SGD-X datasets; CC BY-SA 4.0. Accessed: 2026-02-08.

46. Rastogi, A., Zang, X., Sunkara, S., Gupta, R. & Khaitan, P. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 8689–8696 (2020).

47. Wu, C.-S. *et al.* Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 808–819, DOI: 10.18653/v1/P19-1078 (Association for Computational Linguistics, 2019).

48. Zang, X. *et al.* 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*, 1442, DOI: 10.18653/v1/2020.nlp4convai-1.13 (Association for Computational Linguistics, 2020).

49. Zhang, J. *et al.* Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking. In *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*, 154–164 (Association for Computational Linguistics, 2020).

50. Hosseini-Asl, E., McCann, B., Wu, C.-S., Yavuz, S. & Socher, R. A simple language model for task-oriented dialogue (2020). 2005.00796.

51. Tian, X. *et al.* Amendable generation for dialogue state tracking. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, 62–70, DOI: 10.18653/v1/2021.nlp4convai-1.8 (Association for Computational Linguistics, 2021).

52. Feng, Y., Wang, Y. & Li, H. A sequence-to-sequence approach to dialogue state tracking. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 1691–1701, DOI: 10.18653/v1/2021.acl-long.135 (Association for Computational Linguistics, 2021).

53. Sun, X. *et al.* On tracking dialogue state by inheriting slot values in mentioned slot pools. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI)*, 4369–4377 (2022).