DarwinTOD: LLM Driven Lifelong Self Evolution for Task Oriented Dialog Systems

Anonymous ACL submission

Abstract

Traditional task-oriented dialog systems are unable to evolve from ongoing interactions or adapt to new domains after deployment, that is a critical limitation in real-world dynamic environments. Continual learning approaches depend on episodic retraining with human curated data, failing to achieve autonomy lifelong improvement. While evolutionary computation and LLM driven self improvement offer promising mechanisms for dialog optimization, they lack a unified framework for holistic, iterative strategy refinement. To bridge this gap, we propose DarwinTOD, a lifelong self evolving dialog framework that systematically integrates these two paradigms, enabling continuous strategy optimization from a zero-shot base without task specific fine-tuning. DarwinTOD maintains an Evolvable Strategy Bank and operates through a dual-loop process: online multi-agent dialog execution with peer critique, and offline structured evolutionary operations that refine the strategy bank using accumulated feedback. This closed-loop design enables autonomous continuous improvement without human intervention. Extensive experiments show that DarwinTOD surpasses previous state-of-the-art methods and exhibits continuous performance gains throughout evolution. Our work provides a novel framework for building dialog systems with lifelong self evolution capabilities. The code is available at Anonymous GitHub.

1 Introduction

Task-oriented dialog (TOD) systems aim to assist users in achieving specific goals through natural language conversations, with applications like customer service and personal assistants. Despite advances, deployed systems remain static, unable to learn from ongoing interactions (Madotto et al., 2021). This conflicts with real world needs where user preferences and domains evolve continuously. Truly intelligent dialog agents require lifelong self evolution capabilities to refine strategies throughout their operational lifespan. This creates a critical gap between research prototypes and deployable systems: the former are often evaluated on static benchmarks, while the latter must operate in a dynamic, open ended world. Bridging this gap necessitates a paradigm shift towards systems endowed with lifelong self evolution to autonomously refine their conversational strategies through continuous interaction.

Current TOD paradigms fail to meet this requirement. Pipeline modular systems decompose dialog into components like natural language understanding (NLU), dialog state tracking (DST), dialog policy (DP) and natural language generation (NLG) (Ding et al., 2024a; Gong et al., 2025), but suffer from cascaded error propagation and become brittle to new domains (Huang et al., 2023). End-to-end LLM-based approaches show strong generalization via instruction following (King and Flanigan, 2024; Xu et al., 2024), but remain static after initial training, lacking continuous improvement mechanisms (Li et al., 2025). Even continual learning methods rely on episodic retraining with curated data (Zeng et al., 2025; Xu et al., 2023), not achieving autonomous evolution. The emergence of modern large language models (LLMs) provides a new foundation with their advanced instruction following, reasoning, and text generation capabilities. As depicted in Figure 1, these limitations motivate our core **Research Question**: How can we achieve lifelong self evolution for TOD system, enabling it to continuously and autonomously improve from its own interactions?

Evolutionary computation and LLM driven self-improvement offer promising directions. Evolutionary algorithms enable population based optimization for prompts (Fernando et al., 2024; Guo et al., 2024) and policies (Zhao et al., 2025). LLM-based multi-agent systems solve complex problems through collaboration (Su et al., 2025; Cheng et al.,

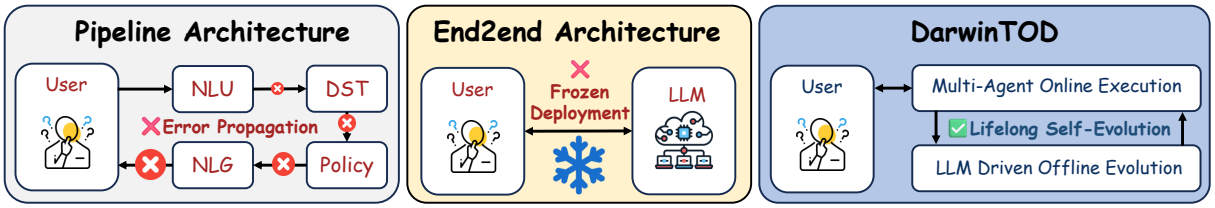

Figure 1: Motivation comparison of TOD architectures. Both pipeline and end-to-end TOD systems suffer from cascaded errors or lack experience driven improvement, while DarwinTOD enables lifelong self evolution via a dual-loop architecture to achieve autonomous improvement.

2024), while self evolving frameworks refine agent behavior through experience (Zhang et al., 2025a; Fang et al., 2025). However, these approaches remain fragmented: evolutionary methods optimize isolated prompts without addressing holistic dialog strategy lifecycles; multi-agent systems often focus on single turn dialog; and self evolving frameworks lack structured dialog management.

We introduce DarwinTOD, a lifelong self evolving dialog framework that integrates evolutionary computation with modern LLM driven strategy optimization. Unlike conventional prompt engineering, DarwinTOD operates as a population based evolutionary system that undergoes parallel competition, fitness based selection, and elimination through a closed-loop, lifelong evolutionary cycle. Its core is an Evolvable Strategy Bank (ESB) and a dual-loop process: online multi-agent execution with peer critique, followed by offline structured evolution using accumulated feedback. This closed-loop design enables fully autonomous self-improvement from a minimal starting point, without task specific fine-tuning or human curation. Our contributions are as follows.

(1) We introduce a novel lifelong self-evolution framework for TOD that systematically integrates LLM-driven evolutionary optimization.

(2) We propose a structured mechanism centered on dynamic ESB and dual-loop cycle that enables self improvement without human intervention.

(3) Extensive empirical validation showing state-of-the-art (SOTA) performance through sustained autonomous evolution.

2 Related Work

TOD Systems. TOD research has evolved through distinct paradigms, each addressing aspects of generalization and adaptability, but none achieving true lifelong autonomy. Pipeline systems decompose dialog into specialized components, enabling interpretability but suffering from error propagation and costly domain re-engineering (Wu et al., 2019; Zhang et al., 2020). End-to-end approaches leverage LLMs to generate responses directly, improving generalization via instruction following (Hosseini-Asl et al., 2020; Yang et al., 2021) but remaining fixed after deployment. Continual learning methods introduce incremental updates to handle new domains or tasks (Liu and Mazumder, 2021; Madotto et al., 2021; Zhao et al., 2022; Kim et al., 2025), yet operate with explicit task boundaries and require episodic retraining with curated data or generative replay. Recent works have highlighted the need for continuous learning during conversation (Mazumder and Liu, 2022) and self evolution capabilities (Tao et al., 2024). A parallel work explores LLM-based agents that can call external tools or APIs dynamically (Xu et al., 2024, 2025a), and self explanation prompting can improve dialog understanding (Gao et al., 2024a), but these methods remain static after deployment. These limitations collectively highlight the need for a paradigm shift toward systems capable of endogenous self evolution without human intervention.

Evolutionary Computation. Evolutionary algorithms provide a population based optimization paradigm, while LLMs with remarkable prowess as general-purpose controllers and optimizers serve as intelligent evolution operators. Classical EAs have been applied to TOD (Zhao et al., 2025) but lack semantic awareness. LLM driven evolution shows promise in prompt engineering (Fernando et al., 2024; Agarwal et al., 2025) and game generation (Todd et al., 2024), with surveys outlining this synergy (Wu et al., 2025). However, existing methods target single-turn or static-goal tasks, lacking systematic support for multi-turn dialog challenges within a lifelong learning framework.

Self Evolving Agents. Research on self-improving agents spans self-reflection (Shim et al., 2025), multi-agent collaboration (Su et al., 2025; Chen et al., 2025), and memory-augmented learn-

ing (Tan et al., 2025). Recent frameworks enable iterative self-improvement via LLMs (Zhang et al., 2025b; Zhai et al., 2025; Gao et al., 2024b). However, these approaches typically focus on updating parameters of a single agent or optimizing for single-turn tasks, and do not address the unique challenges of multi-domain and multi-turn conversational strategy evolution in TOD. DarwinTOD addresses these gaps by introducing a dedicated ESB as an evolving population, dialog specific evolutionary operators, and a dual-loop architecture for continuous autonomous refinement.

3 Methodology

DarwinTOD unifies evolutionary computation with LLM driven strategy optimization, moving beyond single prompt tuning to a population based evolutionary paradigm where strategies compete, mutate, and are selected through a structured dual-loop process. We establish its theoretical foundation by formalizing dialog as a Partially Observable Markov Decision Process (POMDP) and strategy evolution as a Markov chain, instantiated in a dual-loop algorithmic framework that enables autonomous lifelong adaptation without human intervention.

3.1 Theoretical Foundation

We formalize TOD as a POMDP to capture its sequential and partially observable nature, defined by the tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O}\}$. Here, $\mathcal{S}$ is the state space, typically comprising the user goal g, database results db, and dialog history h; $\mathcal{A}$ is the action space of system acts; $\mathcal{T}$ is the state transition function, implemented by the user simulator; $\mathcal{R}$ is the reward function, provided by the dialog manager and offline evolution; Ω is the observation space (user utterance u); $\mathcal{O}$ is the observation function, corresponding to user utterance generation. The system maintains a belief state $b_t(s)$. Our objective is to optimize a dialog policy $\pi : b \mapsto a$ that maps belief states to actions, maximizing the expected cumulative reward:

$$J(\pi) = \mathbb{E}_{\zeta \sim p(\zeta|\pi)} \left[\sum_{t=0}^{T} \mathcal{R}(s_t, a_t) \right]. \quad (1)$$

where ζ denotes a dialog trajectory.

We model the lifelong evolution of dialog strategies by treating the Evolvable Strategy Bank (ESB) as a population in a Markov chain over generations. The ESB in generation t, denoted Π_t, transitions to Π_{t+1} through selection, feedback evaluation, and evolutionary operators. Each strategy $\pi \in \Pi_t$ is assigned a fitness score $\phi(\pi)$ that balances its historical performance against its age to prevent stagnation:

$$\phi(\pi) = \frac{H_\pi^+ - H_\pi^-}{N_\pi + \epsilon} + \alpha \cdot \text{norm}(\pi_{\text{gen}}). \quad (2)$$

Here, H_π^+/H_π^- are positive/negative feedback counts (updated after each dialog based on task success and peer critiques), N_π is the total usage count, π_{gen} is the generation index, and $\text{norm}(\cdot)$ denotes global min-max normalization to scale π_{gen} to the range $[0, 1]$. The term ϵ is a smoothing constant and α controls the age penalty, which discourages the use of older strategies and prevents premature convergence. This fitness function explicitly balances exploitation of high performing strategies with penalization of older strategies, similar to evolutionary algorithms that maintain diversity and avoid premature convergence.

During online execution, a strategy π_i applicable to domain d is selected stochastically according to a Boltzmann distribution based on its fitness:

$$P(\pi_i|d) = \frac{\mathbb{I}(d \in d_i) \cdot \exp\left(\phi(\pi_i)/\tau\right)}{\sum_j \mathbb{I}(d \in d_j) \cdot \exp\left(\phi(\pi_j)/\tau\right)}. \quad (3)$$

where $\mathbb{I}(\cdot)$ is the indicator function, d_i is the set of domains, this function restricts selection to strategies applicable to the current domains d. And $\tau > 0$ is a temperature parameter controlling the exploration exploitation trade-off, a higher τ encourages exploration of lower fitness strategies, while a lower τ favors exploitation of high fitness ones, analogous to simulated annealing in evolutionary computation.

As detailed analysis in Appendix A, the evolutionary process is designed to be robust to noisy mutations and potential biases in LLM generated critiques. The fitness function (Eq. 2) smooths and normalizes feedback over multi-turn interactions, guiding selection pressure toward strategies with consistently high performance. Boltzmann selection (Eq. 3) balances exploration and exploitation based on long term fitness, preventing premature convergence to local optima caused by single noisy critiques. This structured evolution constitutes a directed search in semantic strategy space, where periodic pruning eliminates persistently low fitness candidates. Consequently, the closed-loop design enables the ESB to progressively concentrate on higher performing strategies over time, as validated by experiments in Sec. 4.2.

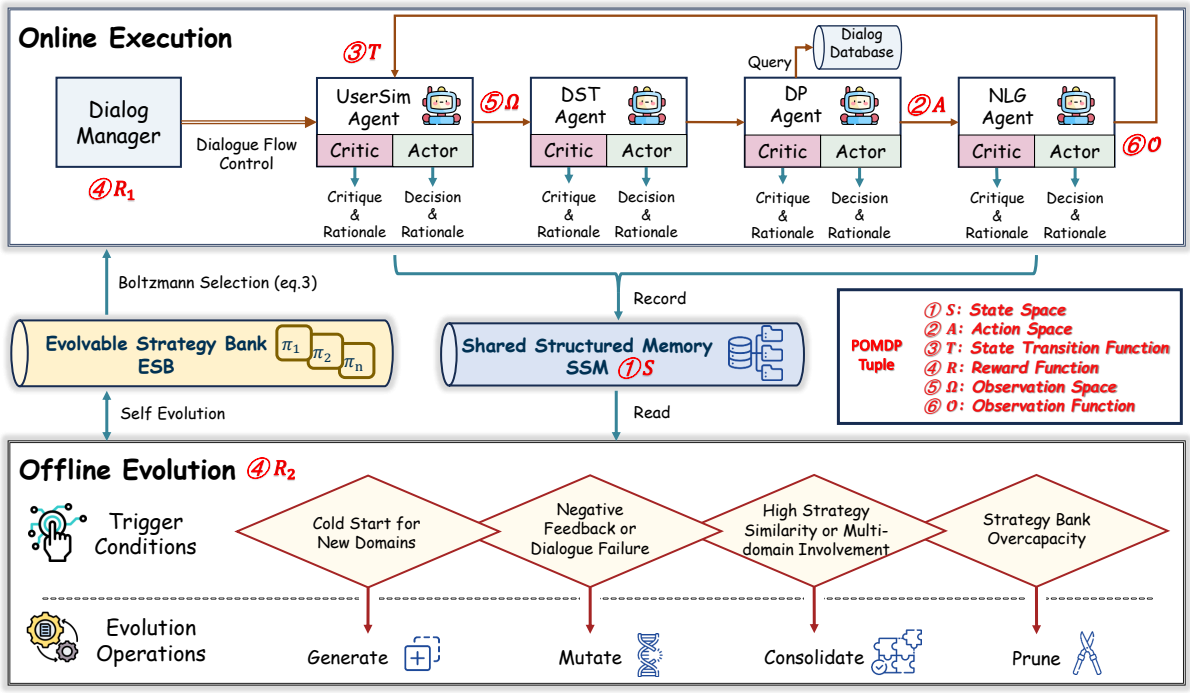

Figure 2: DarwinTOD's dual-loop algorithm framework. The online phase executes dialogs via multi-agent collaboration (DST/DP/NLG/UserSim) with peer critique, retrieving strategies from ESB through Boltzmann selection and logging interactions to SSM. The offline phase triggers evolutionary operations (Generate/Mutate/Consolidate/Prune) based on SSM feedback to update ESB, forming a closed loop for autonomous strategy refinement.

3.2 Algorithmic Framework

Figure 2 depicts the dual-loop architecture of DarwinTOD, which couples online POMDP evaluation with offline Markov chain based evolution. This materializes the theory analysis into an executable algorithmic process. The complete pseudocode and prompts are provided in Sec. F and G respectively.

3.2.1 Core Data Structures

The ESB maintains a population of dialog strategies, each represented as a tuple $\pi_i = \{id_i, d_i, c_i, m_i\}$. Here, id_i is a unique identifier, d_i denotes applicable domains, c_i is the natural language description of the strategy, and $m_i = \{h_i^+, h_i^-, n_i, \text{gen}_i\}$ records metadata: positive/negative feedback counts, usage count, and generation index. Each strategy also includes a rationale field and a lifecycle flag indicating whether it is active.

The SSM stores complete dialog trajectories for evolutionary learning. Each trajectory $\mathcal{H}$ is structured as a tuple $\{d, g, \Pi^{\text{used}}, (u_t, r_t, b_t, a_t, c_t)_{t=1}^{T}\}$, where d is domain/domains, g the user goal, Π^{used} is the set of strategies employed, and each turn contains the user utterance u_t, system response r_t, the current belief state b_t, the system action a_t, and a structured critique log c_t, which includes rationales for each agent and self evaluations, which are used during offline evolution.

ESB and SSM instantiate the theoretical strategy population and experience buffer, respectively.

3.2.2 Online Execution

The online phase implements the POMDP policy evaluation through four specialized LLM agents: DST (Dialog State Tracker), DP (Dialog Policy), NLG (Natural Language Generator), and UserSim (User Simulator). For each dialog turn t:

Strategy Retrieval: Each agent retrieves a strategy from ESB using the domain-aware Boltzmann selection defined in Eq. 3.

Multi-Agent Execution: Agents operate sequentially with built-in critique:

$$u_t \xrightarrow[\text{critique}]{\text{DST}} b_t \xrightarrow[\text{critique}]{\text{DP}} a_t \xrightarrow[\text{critique}]{\text{NLG}} r_t \xrightarrow[\text{critique}]{\text{UserSim}} .$$

Each agent first critiques the previous agent's output, and then produces its own output with justification. This pipeline realizes POMDP evaluation (Eq. 1): DST updates the belief state b_t, DP selects the system action a_t and queries database if needed, NLG generates the response r_t, and UserSim provides critique based on r_t. These critiques serve as immediate reward signals for offline evolution.

Note: In the main experiments, UserSim does not generate the next user utterance u_{t+1}; instead, it leverages the next user utterance from the dataset.
Feedback Collection: Dialog Manager provides the task success assessment; the usage count n_i of each involved strategy is updated; and the entire interaction record and agent critiques are stored in SSM. This collected experience serves as the training corpus for offline strategy evolution.

3.2.3 Offline Evolution

After each dialog episode, the offline phase refines ESB through four evolutionary operators, triggered by interaction feedback and population metrics. The system sustains autonomous improvement amid noisy variation, and not depend on every critique or every mutation. Instead, by selecting and pruning, it filters out detrimental changes while preserving productive ones.
(1) Genesis is triggered when a domain or a combination of domains is encountered and no strategies exist for it. For each agent type, it synthesizes K strategies based solely on the domain name and the agent's role description, without leveraging any prior dialog history. Formally, for a given agent type ag and domain d, the operator generates a set of new strategies $G(\Pi_{new}, d) \sim \mathcal{M}_{\text{LLM}}(\text{GEN} \mid d, ag)_{i=1}^{K}$. The resulting strategies are added to ESB, providing an initial policy repertoire for the new domain without any training data in the domain.
(2) Mutation is applied to strategy π that was involved in a failed dialog or received negative critiques. The operator first uses an LLM to assess whether π was helpful, neutral, or harmful in the given context, and updates its corresponding feedback counts in the ESB. Then it prompts one to generate a revised strategy π' that addresses the identified shortcomings, with the failed trajectory and critiques: $\pi' \sim \mathcal{M}_{\text{LLM}}(\text{MUT} \mid \pi, \mathcal{H})$. The newly created strategy π' inherits the metadata of π but with its generation index incremented.
(3) Consolidation merges a set of n highly similar strategies, if the cosine distances of their SBERT encoded strategy texts exceed threshold δ. The new strategy π_c is synthesized by prompting an LLM with the combined content of n source strategies, its metadata is set to the average of the original, and the maximum generation index plus one: $\pi_c \sim \mathcal{M}_{\text{LLM}}(\text{CON} \mid \pi_1, \pi_2, \ldots, \pi_n)$. The original n strategies are then removed from the ESB to maintain a compact and diverse population.
(4) Pruning maintains a bounded population size M by discarding the lowest fitness strategies after each episode. Strategies are ranked by fitness $\phi(\pi)$, and only the top-M are retained: $P(\Pi_t) = \Pi_t \setminus \{\pi \mid \text{rank}(\phi(\pi)) > M\}$. This ensures computational efficiency while preserving high-performing and diverse strategies.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate DarwinTOD on two established benchmarks: three versions of MultiWOZ (Budzianowski et al., 2018; Eric et al., 2020; Zang et al., 2020) and SGD (Rastogi et al., 2020). These datasets provide multi-domain conversational data and are standard for evaluating TOD system capabilities, details in Appendix B.1.
Baselines. We compare against several strong and recent baselines, including representative pipeline and end-to-end approaches. All baseline results were reported from the original papers. See Appendix B.2 for details.
Metrics. Following standard TOD evaluation protocols, we employ four automatic metrics, the same as in the previous SOTA AgentTOD (Xu et al., 2025b). *Inform*: Measures whether the system provides the correct entity requested by the user. *Success*: Measures whether the system successfully answers all user constraints. *BLEU* (Papineni et al., 2002): Evaluates the quality of the generated response comparing the ground truth, assessing fluency. *Combine*: A composite score defined as Combine $=$ (Inform $+$ Success) $\times$ 0.5 $+$ BLEU, providing an overall performance measure.
Implementation Details. DarwinTOD reads the user's utterance from the training dataset for online execution, records the process data in SSM, and performs offline evolution after each dialog episode. DarwinTOD starts with initial ESB generated by genesis operation solely on domain names and agent role descriptions. We use Llama, Qwen, and GPT as backbone LLMs. More details, hyperparameters, and prompts are provided in Appendix B.3 and G.

4.2 Main Results

The experimental results in Table 1 and Appendix Table 6 demonstrate that DarwinTOD achieves SOTA performance across all MultiWOZ versions. This consistent superiority is a validation of the theoretical framework in Sec. 3.1, effectively ampli-

Table 1: Performance comparison between DarwinTOD and baseline models on MultiWOZ 2.0, 2.1, and 2.2. Metrics include: Inform, Success (Succ.), BLEU, and Combine (Comb.). Bold indicates the best score for metrics. All results of baselines were reported from original papers.

Model	Year	MultiWOZ 2.0				MultiWOZ 2.1				MultiWOZ 2.2			
		Inform	Succ.	BLEU	Comb.	Inform	Succ.	BLEU	Comb.	Inform	Succ.	BLEU	Comb.
SimpleTOD	2020	84.40	70.10	15.01	92.26	85.00	70.50	15.23	92.98	-	-	-	-
MinTL	2020	84.88	74.91	17.89	97.79	-	-	-	-	73.7	65.4	19.4	89.0
SOLOIST	2021	85.50	72.90	16.54	95.74	-	-	-	-	82.3	72.4	13.6	90.9
UBAR	2021	95.40	80.70	17.00	105.05	95.70	81.80	16.50	105.25	83.4	70.3	17.6	94.4
PPTOD	2022	89.20	79.40	18.62	102.92	87.09	79.08	19.17	102.26	-	-	-	-
BORT	2022	93.80	85.80	18.50	108.30	-	-	-	-	85.5	77.4	17.9	99.4
GALAXY	2022	94.40	85.30	20.50	110.35	95.30	86.20	20.01	110.76	85.4	75.7	19.6	100.2
TOATOD	2023	-	-	-	-	97.00	87.40	17.12	109.32	90.0	79.8	17.0	101.9
SGP-TOD	2023	83.88	69.87	9.09	85.97	-	-	-	-	82.00	72.50	9.22	86.47
AutoTOD	2024	91.70	84.40	10.40	98.50	-	-	-	-	-	-	-	-
KMc-ToD	2024	93.90	84.70	20.20	109.30	93.20	84.90	19.60	108.70	-	-	-	-
ProTOD	2025	91.70	83.30	8.90	96.40	-	-	-	-	-	-	-	-
OmniDialog	2025	92.20	79.30	18.57	104.32	-	-	-	-	-	-	-	-
AgentTOD (Prev. SOTA)	2025	95.27	88.95	21.77	113.88	96.73	89.92	20.54	113.87	90.90	81.25	18.49	104.56
DarwinTOD (Llama3-8B)	2026	96.92	89.14	21.83	114.86	98.73	91.42	19.96	115.04	92.58	83.97	17.98	106.26
DarwinTOD (Qwen2.5-7B)	2026	97.63	90.28	21.55	115.51	98.92	91.85	20.18	115.57	92.14	84.33	18.34	106.58
DarwinTOD (Qwen3-8B)	2026	98.34	92.86	21.74	117.34	99.62	94.18	20.33	117.23	94.73	87.25	18.41	109.40
DarwinTOD (GPT-5.1)	2026	**99.10**	**96.20**	**22.94**	**120.59**	**99.40**	**96.50**	**22.19**	**120.14**	**96.48**	**90.12**	**21.98**	**115.28**

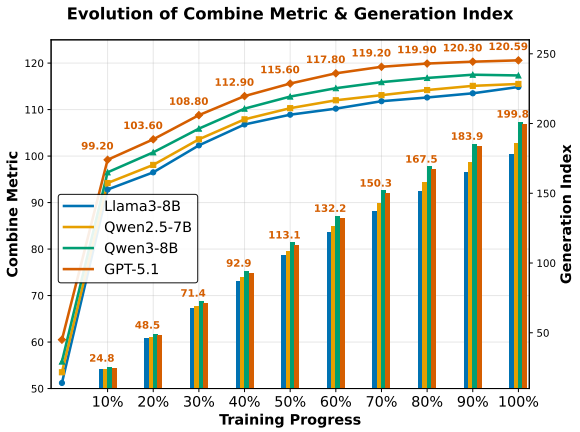

Figure 3: Combine metric evolution across generations on MultiWOZ 2.0. All backbones show monotonic improvement, and the rapid early gains reflect the exploration-exploitation trade-off inherent in evolutionary optimization.

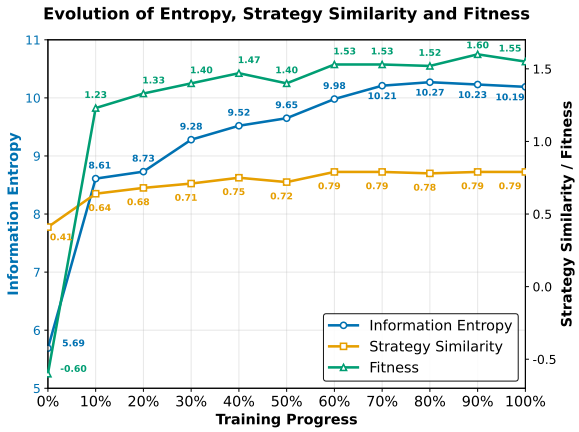

Figure 4: Evolutionary dynamics of ESB across generations on MultiWOZ 2.0 with Qwen3-8B. The simultaneous rise and subsequent decline of entropy and fitness, coupled with increasing pairwise similarity, demonstrates a self organizing transition from exploratory diversity to exploitative convergence.

fying high fitness strategies and suppressing poor ones. The improved performance demonstrates that the framework effectively capitalizes on the enhanced comprehension and instruction following capabilities of modern LLMs. In particular, even with fewer parameters, Qwen3-4B outperforms both Llama3-8B and Qwen2.5-7B, highlighting the importance of architectural advancements alongside model scale. Furthermore, the improvement in Combine score across generations (Fig. 3) empirically validates the convergence property of the evolutionary Markov chain derived in Appendix A.3, demonstrating that the stochastic LLM driven mutation operator is successfully filtered by selection and pruning, leading to expected fitness increase.

The generational dynamics in Fig. 3 reveal that performance improves rapidly in early generations and slows as task success rates rise, reflecting the exploration exploitation trade-off inherent in evolutionary optimization. To dissect the internal evolutionary mechanisms, we conduct detailed experiments (Appendix C.2), then uncover three complementary patterns: (1) DP strategies undergo significantly more revisions than other modules (Table 8), indicating evolutionary focus on the core decision making component; (2) Strategies progressively cluster into domain distinct groups in embedding

Table 2: Ablation study of DarwinTOD components on MultiWOZ 2.0 with Qwen3-8B backbone. Δ indicates combine change relative to the full system.

Variant	Inform	Succ.	BLEU	Combine	Δ
DarwinTOD (Full)	98.34	92.86	21.74	117.34	–
Online Execution Variants					
w/o Reasoning	92.53	86.93	20.95	110.68	−6.66
w/o Peer Critique	82.79	77.78	18.74	99.03	−18.31
w/ E2E Agent	78.00	78.00	17.54	95.54	−21.80
Offline Evolution Variants					
w/o Evolution	75.25	70.96	19.17	92.28	−25.06
w/o Consolidate	92.71	87.11	20.99	110.91	−6.43
w/o Prune	91.56	86.01	20.73	109.51	−7.83
Selection & Retrieval Variants					
w/ Roulette Wheel	94.48	88.76	21.39	113.01	−4.33
w/ Random	93.50	87.84	21.17	111.84	−5.50
w/ ϵ-Greedy	95.45	89.67	21.61	114.17	−3.17

space (Fig. 5 and 6), demonstrating structural self organization; and (3) Information entropy and fitness peak then decline while pairwise similarity rises (Fig. 4), reflecting a shift from exploration to exploitation. Together, these patterns reveal that DarwinTOD's improvement stems from systematic knowledge consolidation: it autonomously allocates evolutionary effort to the most impactful module, drives semantic specialization through domain feedback, and balances diversity maintenance with selective convergence. This internal self organization, mirrored in the external performance gains, confirms that DarwinTOD achieves genuine lifelong learning through structured evolutionary dynamics rather than incremental patching.

To evaluate the adaptability of DarwinTOD under data scarcity conditions and in the absence of task specific training data, we conduct comprehensive few-shot experiments (Appendix C.3) and zero-shot experiments (Appendix C.4). The results demonstrate that the system can effectively leverage its evolutionary operators to rapidly bootstrap and refine strategies from limited interactions. In both settings, it exhibits strong performance and surpasses previous SOTA methods. This confirms that DarwinTOD achieves robust zero-shot generalization through its modular architecture and explicit reasoning pathways. In practical deployment, its performance on unseen domains can be further enhanced through continuous offline evolution.

4.3 Ablation Study

The ablation results (Table 2) indicate that each component of DarwinTOD contributes substantially to overall performance.

Online Execution Variants. Removing agent reasoning (w/o Reasoning) degrades performance, as subsequent agents cannot integrate prior outputs and rationales, so critiques lack substantive grounding. Removing peer critique (w/o Peer Critique) causes a significant drop, since evolution depends only on sparse end of dialog success signals, whereas per-turn critiques provide dense incremental feedback. Replacing the modular pipeline with a monolithic end-to-end agent (w/ *E2E Agent*) causes largest performance drop, validating the POMDP theory: pipeline modules not only prevent cascading error propagation, but also enhance structured state reasoning for targeted strategy evolution.

Offline Evolution Variants. Disabling the offline evolutionary loop (w/o *Evolution*) corresponds to the zero-shot setup, where each agent uses a single manually designed domain agnostic static strategy. Although such strategies can achieve strong performance, they lack the ability to adapt to specific domains or learn from interaction failures and negative critiques. Without consolidation (w/o *Consolidate*) or pruning (w/o *Prune*), the strategy bank becomes redundant and inefficient. This confirms that consolidation and pruning are crucial for maintaining a compact and high performing ESB.

Selection Variants. Replacing Boltzmann selection with roulette-wheel, uniform random sampling, or ϵ-greedy ($\epsilon = 0.1$) yields lower performance, demonstrating that Boltzmann is superior for balancing exploitation of high fitness strategies with exploration of newer ones. These results confirm that effective lifelong learning requires an explicit balance between exploration and exploitation.

4.4 Human and Real World Evaluation

To complement automated metrics, we conduct three human studies (Appendix E) assessing the practical utility, safety, and user experience of DarwinTOD's evolved strategies. An expert evaluation (Fig. 9) rated strategies from different LLM backbones in five dimensions. Results show evolved strategies achieve high scores in all backbones, with safety and interpretability being particularly robust, indicating that evolutionary pressure combined with structured peer critique inherently promotes aligned and comprehensible policies. The real-user study (Table 14) shows the second half of dialogs showed a 3.51 point increase in success rate, a reduction of 2.44 turns on average, demonstrating that the system improves continuously through interaction. An adversarial and off-topic input test (Table 15) confirms that the system maintains task

focus and generates safe, natural responses even under challenging input.

These results collectively indicate that DarwinTOD's self evolution translates into tangible gains in real-world usability and user perception, while its architecture inherently ensures robustness and goal directed recovery without cascading failures. This work bridges the gap between autonomous optimization and practical deployment, offering a path toward conversational agents that continuously improve through interaction while remaining robust and aligned.

5 Analysis

Supplementary experiments (Appendix D) reveal how the dual-loop architecture balances autonomy, robustness, and efficiency. The initialization analysis (Table 11) confirms that the system is robust to various starting conditions, whether minimal descriptions, small strategy banks, or even expert curated seeds. Performance converges to near expert levels regardless of initial quality, demonstrating that the evolutionary loop effectively compensates for sparse prior knowledge. This aligns with the theoretical Markov chain model (Appendix A.3): fitness driven selection and pruning gradually filter out noisy mutations, enabling the population to self organize toward high performance strategies without manual tuning. The result underscores a fundamental advantage of evolutionary learning over fixed or episodic update paradigms: the system can bootstrap and refine dialog policies from near-zero knowledge, reducing dependency on costly human curation.

The sensitivity study of Boltzmann temperature τ (Fig. 7) highlights the trade-off between exploration and exploitation. Optimal performance at $\tau = 1.0$ reflects balance, while excessively low τ leads to premature convergence, and high τ slows progress by diluting selection pressure. This mirrors classic evolutionary algorithms and reinforcement learning, yet here the balance is managed implicitly through a single interpretable parameter. The finding validates that DarwinTOD's selection mechanism not only supports sustained improvement but also offers a tunable knob for adapting to different environments, a property essential for deployment in dynamic real-world settings.

The evolution and efficiency analysis of the cross models (Appendix D.3) demonstrate the practical scalability and architectural flexibility of the framework. Using a powerful LLM only for offline evolution substantially boosts performance even when online agents are lighter, confirming that evolution can be decoupled from execution to optimize cost and latency. Meanwhile, aggressive real-time feedback mechanisms (online arbitration, per-turn evolution) yield diminishing returns relative to their computational overhead, reinforcing the design choice of post-dialog offline evolution as a sustainable compromise. Together, these results illustrate that DarwinTOD's modular, dual-loop design not only achieves autonomous self-improvement but also admits efficient deployment strategies, paving the way for lifelong learning systems that are both effective and practical.

To illustrate the internal evolutionary dynamics and error resilience of DarwinTOD, we conduct two case studies in Appendix H: one traces the evolutionary trajectory of a DP strategy across generations, and another examines how peer critique both prevents cascading errors and catalyzes targeted strategy evolution in a multi-domain dialog. These studies collectively validate that the dual-loop architecture enables continuous strategy refinement and robust error containment through structured evolution and inter agent critique.

6 Conclusion

DarwinTOD advances self-evolving dialog systems by formulating strategy optimization as the population based evolutionary process. Unlike traditional prompt engineering, which seeks a single best prompt for a static task, our system maintains a diverse and competing set of strategies that continuously adapt through interaction. This shift from point wise prompt tuning to population wise strategy evolution enables truly lifelong and autonomous improvement in dynamic TOD settings. DarwinTOD achieves SOTA performance across benchmarks and demonstrates sustained improvement via structured evolutionary dynamics. Our experiments show that the Evolvable Strategy Bank self organizes into domain specialized and high fitness strategies, and human evaluations confirm the safety, interpretability, and practical effectiveness of the evolved strategies. This work establishes a scalable foundation for building continuously learning conversational agents, opening avenues for more efficient evolutionary operators and human-in-the-loop adaptation in autonomous interactive systems.

Limitations

Although DarwinTOD has been rigorously validated on established TOD benchmarks such as MultiWOZ and SGD, its practical utility may be constrained by the current reliance on simulated interactions. To fully realize task completion in real-world deployments, the framework would benefit from enhanced capabilities in function calling and agent oriented orchestration, enabling robust integration with external systems and dynamic tool execution. Future work should strengthen these aspects to bridge the gap between benchmark performance and operational effectiveness in open environments.

Ethical Considerations

In this work, we utilized the publicly and already de-identified MultiWOZ and SGD datasets for evaluation. These datasets do not contain personally identifiable information or offensive content, and their use complies with the consent agreements established during their original release. As our study did not involve new data collection from human participants, no ethics review board approval was required. Throughout our extensive experimental evaluations, including human assessments, we did not observe the emergence or amplification of social biases, harmful content, or manipulative behaviors in the evolved strategies. The evolutionary process, governed by structured peer critique and fitness-based selection oriented towards task success, demonstrated an inherent tendency to converge towards effective and neutral dialogue strategies. However, as with any autonomous learning system deploying LLMs, ongoing monitoring in real world applications remains a recommended practice to safeguard against unforeseen edge cases.

We acknowledge the use of Writeful integrated with Overleaf for refining the textual expression of this manuscript, and DeepSeek V3.2 for error correction of the experimental code. The role of these LLMs was limited to technical assistance and did not involve research ideation or the creation of core content. All LLM outputs have been rigorously verified by the authors, who bear full responsibility for the final accuracy, integrity, and originality of the content including the avoidance of plagiarism or scientific misconduct.

References

Eshaan Agarwal, Raghav Magazine, Joykirat Singh, Vivek Dani, Tanuja Ganu, and Akshay Nambi. 2025. PromptWizard: Optimizing prompts via task-aware, feedback-driven self-evolution. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19974–20003, Vienna, Austria. Association for Computational Linguistics.

Namo Bang, Jeehyun Lee, and Myoung-Wan Koo. 2023. Task-optimized adapters for an end-to-end task-oriented dialogue system. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7355–7369, Toronto, Canada. Association for Computational Linguistics.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2025. Optima: Optimizing effectiveness and efficiency for LLM-based multi-agent system. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11534–11557, Vienna, Austria. Association for Computational Linguistics.

Yi Cheng, Wenge Liu, Jian Wang, Chak Tou Leong, Yi Ouyang, Wenjie Li, Xian Wu, and Yefeng Zheng. 2024. Cooper: Coordinating specialized agents towards a complex dialogue goal. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17853–17861.

Zeyuan Ding, Zhihao Yang, Yinbo Qiao, and Hongfei Lin. 2024a. Kmc-tod: Structure knowledge enhanced multi-copy network for task-oriented dialogue system. *Knowledge-Based Systems*, 293:111662.

Zeyuan Ding, Zhihao Yang, Yinbo Qiao, and Hongfei Lin. 2024b. Kmc-tod: Structure knowledge enhanced multi-copy network for task-oriented dialogue system. *Know.-Based Syst.*, 293(C).

Wenjie Dong, Sirong Chen, and Yan Yang. 2025. ProTOD: Proactive task-oriented dialogue system based on large language model. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9147–9164, Abu Dhabi, UAE. Association for Computational Linguistics.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. MultiWOZ 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 422–428, Marseille, France. European Language Resources Association.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, Zhaochun Ren, Nikos Aletras, Xi Wang, Han Zhou, and Zaiqiao Meng. 2025. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *Preprint*, arXiv:2508.07407.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2024. Promptbreeder: self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Haoyu Gao, Ting-En Lin, Hangyu Li, Min Yang, Yuchuan Wu, Wentao Ma, Fei Huang, and Yongbin Li. 2024a. Self-explanation prompting improves dialogue understanding in large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 14567–14578, Torino, Italia. ELRA and ICCL.

Jinglong Gao, Xiao Ding, Yiming Cui, Jianbai Zhao, Hepeng Wang, Ting Liu, and Bing Qin. 2024b. Self-evolving GPT: A lifelong autonomous experiential learner. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6385–6432, Bangkok, Thailand. Association for Computational Linguistics.

Yeseul Gong, Heeseon Kim, Seokju Hwang, Donghyun Kim, and Kyong-Ho Lee. 2025. Multi-domain dialogue state tracking via dual dynamic graph with hierarchical slot selector. *Knowledge-Based Systems*, 308:112754.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2024. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *International Conference on Representation Learning*, volume 2024, pages 34133–34156.

Wanwei He, Yinpei Dai, Yinhe Zheng, Yuchuan Wu, Zheng Cao, Dermot Liu, Peng Jiang, Min Yang, Fei Huang, Luo Si, Jian Sun, and Yongbin Li. 2022. Galaxy: A generative pre-trained model for task-oriented dialog with semi-supervised learning and explicit policy injection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):10749–10757.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Zhenhua Huang, Fancong Li, Juanjuan Yao, and Zonggan Chen. 2023. Mgcrl: Multi-view graph convolution and multi-agent reinforcement learning for dialogue state tracking. *Neural Comput. Appl.*, 36(9):4829–4846.

Minji Kim, Joon Yoo, and OkRan Jeong. 2025. Meditoder: Medical domain-incremental task-oriented dialogue generator using experience replay. *Expert Systems*, 42(2):e13773.

Brendan King and Jeffrey Flanigan. 2024. Unsupervised end-to-end task-oriented dialogue with LLMs: The power of the noisy channel. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8283–8300, Miami, Florida, USA. Association for Computational Linguistics.

Robert Lange, Yingtao Tian, and Yujin Tang. 2024. Large language models as evolution strategies. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '24 Companion, page 579–582, New York, NY, USA. Association for Computing Machinery.

Pan Li, Qingwen Yang, Sai Xu, Xuejing Li, Zhi Li, Chen Wang, Yanyi Liu, Tiezheng Guo, Jiawei Tang, and Yingyou Wen. 2025. Adaptive-tod: An llm-driven and adaptive agent for diverse interaction modes. *Neurocomputing*, 652:130991.

Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, and Pascale Fung. 2020. MinTL: Minimalist transfer learning for task-oriented dialogue systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3391–3405, Online. Association for Computational Linguistics.

Bing Liu and Sahisnu Mazumder. 2021. Lifelong and continual learning dialogue systems: Learning during conversation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(17):15058–15063.

Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul Crook, Bing Liu, Zhou Yu, Eunjoon Cho, Pascale Fung, and Zhiguang Wang. 2021. Continual learning in task-oriented dialogue systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7452–7467, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Samir W. Mahfoud. 1996. *Niching methods for genetic algorithms*. Ph.D. thesis, University of Illinois at Urbana-Champaign, USA. UMI Order No. GAX95-43663.

Sahisnu Mazumder and Bing Liu. 2022. Continual learning dialogue systems - learning during conversation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 3429–3432, New York, NY, USA. Association for Computing Machinery.

Adib Mosharrof, Muhammad Hasan Maqbool, and A.B. Siddique. 2023. Zero-shot generalizable end-to-end task-oriented dialog system using context summarization and domain schema. *The International FLAIRS Conference Proceedings*, 36.

Efe Precious Onakpojeruo, Berna Uzun, Leena R. David, Ilker Ozsahin, and Dilber Uzun Ozsahin. 2024. Selection techniques in genetic algorithm. In *2024 17th International Conference on Development in eSystem Engineering (DeSE)*, pages 411–416.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2021. Soloist: Building task bots at scale with transfer learning and machine teaching. *Transactions of the Association for Computational Linguistics*, 9:807–824.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8689–8696.

Anton Razzhigaev, Maxim Kurkin, Elizaveta Goncharova, Irina Abdullaeva, Anastasia Lysenko, Alexander Panchenko, Andrey Kuznetsov, and Denis Dimitrov. 2024. OmniDialog: A multimodal benchmark for generalization across text, visual, and audio modalities. In *Proceedings of the 2nd GenBench Workshop on Generalisation (Benchmarking) in NLP*, pages 183–195, Miami, Florida, USA. Association for Computational Linguistics.

Jeonghoon Shim, Gyuhyeon Seo, Cheongsu Lim, and Yohan Jo. 2025. Tooldial: Multi-turn dialogue generation method for tool-augmented language models. In *International Conference on Representation Learning*, volume 2025, pages 70414–70439.

Haoyang Su, Renqi Chen, Shixiang Tang, Zhenfei Yin, Xinzhe Zheng, Jinzhe Li, Biqing Qi, Qi Wu, Hui Li, Wanli Ouyang, Philip Torr, Bowen Zhou, and Nanqing Dong. 2025. Many heads are better than one: Improved scientific idea generation by a LLM-based multi-agent system. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28201–28240, Vienna, Austria. Association for Computational Linguistics.

Yixuan Su, Lei Shu, Elman Mansimov, Arshit Gupta, Deng Cai, Yi-An Lai, and Yi Zhang. 2022. Multi-task pre-training for plug-and-play task-oriented dialogue system. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4661–4676, Dublin, Ireland. Association for Computational Linguistics.

Haipeng Sun, Junwei Bao, Youzheng Wu, and Xiaodong He. 2022. BORT: Back and denoising reconstruction for end-to-end task-oriented dialog. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2156–2170, Seattle, United States. Association for Computational Linguistics.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, Anand Rajan Iyer, Tianlong Chen, Huan Liu, Chen-Yu Lee, and Tomas Pfister. 2025. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8416–8439, Vienna, Austria. Association for Computational Linguistics.

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. 2024. A survey on self-evolution of large language models. *Preprint*, arXiv:2404.14387.

Graham Todd, Alexander G. Padula, Matthew Stephenson, Éric Piette, Dennis J.N.J. Soemers, and Julian Togelius. 2024. Gavel: generating games via evolution and language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819, Florence, Italy. Association for Computational Linguistics.

Xingyu Wu, Sheng-Hao Wu, Jibin Wu, Liang Feng, and Kay Chen Tan. 2025. Evolutionary computation in the era of large language model: Survey and roadmap. *IEEE Transactions on Evolutionary Computation*, 29(2):534–554.

Heng-Da Xu, Xian-Ling Mao, Fanshu Sun, Tian-Yi Che, Chun Xu, and Heyan Huang. 2025a. Agenttod: A task-oriented dialogue agent with a flexible and adaptive api calling paradigm. *ACM Trans. Inf. Syst.*, 43(5).

Heng-Da Xu, Xian-Ling Mao, Fanshu Sun, Tian-Yi Che, Chun Xu, and Heyan Huang. 2025b. Agenttod: A task-oriented dialogue agent with a flexible and

adaptive api calling paradigm. *ACM Transactions on Information Systems*, 43(5):1–32.

Heng-Da Xu, Xian-Ling Mao, Puhai Yang, Fanshu Sun, and Heyan Huang. 2024. Rethinking task-oriented dialogue systems: From complex modularity to zero-shot autonomous agent. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2748–2763, Bangkok, Thailand. Association for Computational Linguistics.

Qiancheng Xu, Min Yang, and Ruifeng Xu. 2023. Balanced meta learning and diverse sampling for lifelong task-oriented dialogue systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(11):13843–13852.

Yunyi Yang, Yunhao Li, and Xiaojun Quan. 2021. Ubar: Towards fully end-to-end task-oriented dialog system with gpt-2. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14230–14238.

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. 2020. 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*, pages 109–117, Online. Association for Computational Linguistics.

Min Zeng, Haiqin Yang, Xi Chen, and Yike Guo. 2025. Task-wrapped continual learning in task-oriented dialogue systems. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3173–3183, Albuquerque, New Mexico. Association for Computational Linguistics.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, Zhaoyang Liu, Bolin Ding, and Jingren Zhou. 2025. Agentevolver: Towards efficient self-evolving agent system. *Preprint*, arXiv:2511.10395.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. 2025a. Memgen: Weaving generative latent memory for self-evolving agents. *Preprint*, arXiv:2509.24704.

Jian Zhang, Zhangqi Wang, Haiping Zhu, Kangda Cheng, Kai He, Bo Li, Qika Lin, Jun Liu, and Erik Cambria. 2025b. Mars: Multi-agent adaptive reasoning with socratic guidance for automated prompt optimization. *Preprint*, arXiv:2503.16874.

Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020. Task-oriented dialog systems that consider multiple appropriate responses under the same context. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9604–9611.

Yisong Zhang and Guoxing Yi. 2025. Laos: Large language model-driven adaptive operator selection for evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '25, page 517–526, New York, NY, USA. Association for Computing Machinery.

Yangyang Zhao, Ben Niu, Libo Qin, and Shihan Wang. 2025. An efficient task-oriented dialogue policy: Evolutionary reinforcement learning injected by elite individuals. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3429–3442, Vienna, Austria. Association for Computational Linguistics.

Yingxiu Zhao, Yinhe Zheng, Zhiliang Tian, Chang Gao, Jian Sun, and Nevin L. Zhang. 2022. Prompt conditioned VAE: Enhancing generative replay for lifelong learning in task-oriented dialogue. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11153–11169, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Appendix Contents

A Theoretical Analysis of Evolutionary Dynamics **13**
- A.1 Foundations of the Evolutionary Process 13
- A.2 Robustness to Noisy Critiques . . 13
- A.3 Convergence Analysis 13

B Experiment Details **14**
- B.1 Datasets Description 14
- B.2 Baselines Details 15
- B.3 Detailed Experimental Setup . . . 16

C Detailed Main Experimental Results **17**
- C.1 Complete Performance Results on MultiWOZ 17
- C.2 Evolution Dynamics Analysis . . 17
- C.3 Evaluation for Few-shot Capability 19
- C.4 Evaluation for Zero-shot Capability 19

D Supplementary Experiments **20**
- D.1 Initialization and Robustness Analysis 20
- D.2 Analysis of Selection Temperature Sensitivity 21
- D.3 System Extension and Efficiency . 21

E Human Studies and Evaluation **23**
- E.1 Evaluation on Evolved Strategies . 23
- E.2 Real User Study 24
- E.3 Robustness to Adversarial and Off-Topic Inputs 25
- E.4 Synthesis and Implications 25

F DarwinTOD Dual-Loop Algorithm Pseudocode **26**

G Agent Prompt Templates **26**
G.1 DST Agent Prompt 26
G.2 DP Agent Prompt 26
G.3 NLG Agent Prompt 27
G.4 UserSim Agent Prompt 27
G.5 End2end Agent Prompt 28
G.6 Arbiter Agent Prompt 28
G.7 Evolutionary Operator Prompts . . 29
G.8 Manually Designed Strategies in Zero-shot Setup 30

H Case Study **31**
H.1 Evolution Trajectory of a DP Agent Strategy 31
H.2 Strategy Evolution and Cascade Error Prevention 33

A Theoretical Analysis of Evolutionary Dynamics

This appendix provides a formal analysis of the evolutionary dynamics underlying DarwinTOD. We explain how its dual-loop architecture enables continuous improvement despite potential biases in LLM generated critiques and the inherent stochasticity of LLM driven mutation.

A.1 Foundations of the Evolutionary Process

We frame the lifelong learning of ESB as a Markov process across generations. The ESB at generation t, denoted Π_t, evolves to Π_{t+1} through selection, feedback accumulation, and evolutionary operators. Each strategy $\pi \in \Pi_t$ is associated with a fitness score $\phi(\pi)$ defined in Eq. 2, which balances cumulative positive feedback H_π^+ against negative feedback H_π^-, normalized by usage count N_π to ensure fair comparison. The term $\alpha \cdot \text{norm}(\pi_{\text{gen}})$ introduces an age penalty that prevents stagnation and encourages exploration of newer variants.

During online execution, a strategy π_i applicable to domain d is selected probabilistically according to the Boltzmann distribution (Eq. 3) (Onakpojeruo et al., 2024). The temperature τ controls the exploration exploitation trade-off: a lower τ amplifies the selection pressure toward high-fitness strategies, while a higher τ allows more exploration. This probabilistic selection aligns with fitness proportionate selection in evolutionary computation (Mahfoud, 1996) and ensures that the population does not prematurely converge to sub-optimal peaks.

A.2 Robustness to Noisy Critiques

A central concern is that the peer critiques generated by LLM agents may contain noise, bias, or self reinforcing errors. DarwinTOD does not rely on the accuracy of any single critique. Instead, it leverages long term statistics at the strategy level and population level selection to filter out such noise.

Formally, let $c_{\pi,t}$ be the critique signal for strategy π at turn t, which can be decomposed as

$$c_{\pi,t} = \bar{c}_\pi + \eta_{\pi,t},$$

where $\bar{c}_\pi$ is the underlying true quality signal and $\eta_{\pi,t}$ is a noise term that may exhibit bias. The fitness function (Eq. 2) aggregates critiques over multiple interactions. By the law of large numbers, as N_π grows, the sample averages converge to the true expected value up to a bias term:

$$\lim_{N_\pi \to \infty} \frac{H_\pi^+ - H_\pi^-}{N_\pi} = \mathbb{E}[\bar{c}_\pi] + \mathcal{O}\left(\frac{\sigma_\eta}{\sqrt{N_\pi}}\right) + \text{bias}(\eta),$$

where σ_η denotes the variability of the noise and $\text{bias}(\eta)$ captures any systematic deviation. The evolutionary loop only requires that the signal to noise ratio be sufficient for the selection and pruning mechanisms to distinguish better strategies over time.

Boltzmann selection does not use individual $c_{\pi,t}$, but uses the fitness $\phi(\pi)$ to determine selection probabilities. Strategies that receive spuriously high or low critiques in a few dialogs are not permanently advantaged or disadvantaged because the fitness score smooths out short term fluctuations. The periodic pruning operation further removes strategies whose fitness remains low over many generations, providing an additional correction at the population level. Consequently, even if critiques are occasionally biased or erroneous, the combined effect of fitness aggregation, probabilistic selection, and pruning ensures that the ESB evolves toward genuinely better strategies.

A.3 Convergence Analysis

We now analyze the convergence behavior of the ESB under the joint influence of noisy critiques and stochastic mutations. The mutation operator, powered by an LLM, does not guaranty improvement and may produce strategies that are better, worse, or neutral relative to their parents (Lange et al., 2024; Zhang and Yi, 2025). Let π' be a mutant of π. The change in fitness can be written as

$$\Delta\phi = \phi(\pi') - \phi(\pi) = \delta + \xi,$$

where δ is the systematic improvement or degradation introduced by the mutation, and ξ is a random variable representing the noise of the mutation.

The evolutionary loop does not require every mutation to be beneficial. Mutations are triggered only when a strategy is involved in a failed dialog or receives negative critiques (Section 3.2), ensuring that evolutionary effort focuses on identified weaknesses. The consolidation operator merges strategies only when their semantic similarity exceeds threshold δ, producing a consolidated strategy whose fitness is at least the average of its constituents. The pruning operator directly removes the lowest fitness strategies, providing a strict lower bound on population fitness.

To characterize the convergence dynamics, we analyze the expected average fitness at generation t:

$$\bar{\phi}(\Pi_t) = |\Pi_t|^{-1} \sum_{\pi \in \Pi_t} \phi(\pi).$$

Let p_t denote the proportion of strategies involved in failed dialogs or receiving negative critiques at generation t, and let $\mu_t = \mathbb{E}[\delta]$ be the expected systematic improvement of mutations conditioned on their occurrence. Experimentally, we observe that p_t decreases over generations as ESB accumulates higher-quality strategies (Figure 4), while μ_t becomes increasingly positive as evolutionary pressure refines the mutation operator's effectiveness.

Under these observable conditions, the expected change in average fitness satisfies:

$$\mathbb{E}[\bar{\phi}(\Pi_{t+1}) \mid \Pi_t] \geq \bar{\phi}(\Pi_t) + p_t \cdot \mu_t - \eta_t,$$

where η_t represents the noise term arising from stochastic mutations and critiques, which diminishes as the population stabilizes. The inequality holds because: (1) selection preferentially replicates high fitness strategies, (2) consolidation preserves or improves fitness, (3) pruning eliminates only the lowest fitness individuals, and (4) mutations are applied to underperforming strategies with expected improvement μ_t.

As evolution progresses, p_t decreases as fewer strategies need evolve, while μ_t increases as mutations become more targeted, causing the product $p_t \cdot \mu_t$ to eventually dominate the noise term η_t. Consequently, the sequence $\{\mathbb{E}[\bar{\phi}(\Pi_t)]\}_{t=0}^{\infty}$ exhibits an overall upward trend, as empirically validated in Figure 4, where the average fitness rises despite non-monotonic fluctuations due to exploration. The bounded strategy space and finite population size ensure that this sequence converges to a stable distribution concentrated on high performing strategies.

Table 3: Key statistics of the MultiWOZ 2.0 dataset.

Statistic	Value
Total Dialogs	10,438
Train Dialogs	8,438
Dev Dialogs	1,000
Test Dialogs	1,000
Total Turns	115,434
Domains	7
Avg. Turns per Dialog	13.68
Slots	25
Slot Values in Ontology	4,510

B Experiment Details

B.1 Datasets Description

This work employs the MultiWOZ and the SGD datasets to evaluate dialog state tracking models. Both datasets are widely recognized and serve as standard benchmarks in the TOD community, allowing direct comparison with established baselines. While MultiWOZ provides a corpus of human-human conversations with multiple refined versions focusing on annotation quality and consistency, SGD offers a larger-scale, synthetically generated dataset designed to test scalability and zero-shot generalization across a diverse and dynamic set of services. The complementary nature of these datasets enables a comprehensive evaluation of the performance of the model under different conditions.

B.1.1 MultiWOZ

MultiWOZ 2.0 (Budzianowski et al., 2018) is a large-scale, multi-domain, fully-annotated corpus of human-human task-oriented dialogs. It was collected using a Wizard-of-Oz framework and spans seven distinct domains: Restaurant, Hotel, Attraction, Taxi, Train, Hospital, and Police. With more than 10,000 dialogs, it is one of the largest publicly available datasets of its kind and has been widely adopted as a standard benchmark for dialog state tracking (DST) and related tasks. Its scale, multi-domain nature, and rich annotations have made it a foundational resource in the dialog research community.

Following the release of MultiWOZ 2.0, two subsequent versions were introduced to address

Table 4: Key statistics of the SGD dataset.

Statistic	Value
Total Dialogs	22,825
Train Dialogs	16,142
Dev Dialogs	2,482
Test Dialogs	4,201
Total Turns	467,916
Domains	16
Services	45
Avg. Turns per Dialog	20.44
Slots	214
Slot Values	14,139

annotation noise and inconsistencies. MultiWOZ 2.1 (Eric et al., 2020) corrected substantial noise in dialog state annotations and user utterances, affecting 32% of state annotations across 40% of turns; also canonicalized slot values, added user dialog acts, and included natural language descriptions for each slot to facilitate low-resource and zero-shot learning.

Building upon 2.1, MultiWOZ 2.2 (Zang et al., 2020) introduced further refinements: it corrected additional state errors in 17.3% of utterances, redefined the ontology by splitting slots into categorical and non-categorical types, added slot span annotations for non-categorical slots to support span-based models, and introduced annotations for active user intents and requested slots per turn. These revisions aim to provide a cleaner, more consistent, and more richly annotated benchmark for robust evaluation of dialog state tracking models.

B.1.2 Schema-Guided Dialog (SGD)

The Schema-Guided Dialog (SGD) dataset (Rastogi et al., 2020) is a large-scale, multi-domain, task-oriented dialog corpus designed to address the scalability challenges of modern virtual assistants. Unlike previous datasets that assume a single static ontology per domain, SGD introduces a schema-guided paradigm where each service provides its own schema containing intents, slots, and natural language descriptions. This approach enables models to handle heterogeneous APIs and facilitates zero-shot generalization to unseen services. The dataset spans 16 distinct domains with 45 distinct services and includes multi-domain conversations that reflect realistic user interactions. With over 16,000 dialogs, SGD is the largest publicly available task-oriented dialog dataset and serves as a benchmark for intent prediction, slot filling, dialog state tracking, and language generation in scalable, multi-service environments.

B.2 Baselines Details

To comprehensively evaluate the performance of DarwinTOD, we compare it with a range of representative and state-of-the-art models in TOD systems. These baselines are categorized according to their architectural paradigms, covering early generative models, pre-trained dialog models, and recent LLM-based agent systems.

B.2.1 Early Generative Models

These early approaches unify the dialog process into a single sequence-to-sequence or auto regressive model, based on pre-trained language models, reducing modular dependencies and demonstrating the potential of generative architectures for TOD.

SimpleTOD (Hosseini-Asl et al., 2020) formulates all TOD sub-tasks as a single causal language modeling problem, generating belief states, system acts, and responses autoregressively within a fixed extract-then-query turn structure.

MinTL (Lin et al., 2020) employs a transfer learning framework with Levenshtein belief spans to jointly learn DST and response generation in an end-to-end manner.

SOLOIST (Peng et al., 2021) integrates DST and response generation into a single auto regressive language model, though it retains a turn-by-turn state extraction and database query paradigm.

UBAR (Yang et al., 2021) fine-tunes GPT-2 on complete dialog sessions, including belief states, database results, and system acts-to achieve fully end-to-end dialog modeling.

B.2.2 Pre-trained TOD Models

These models leverage pre-trained language models on large scale dialog corpora to acquire general dialog abilities before task specific fine-tuning. They represent the dominant paradigm before the rise of large scale instruction-tuned LLMs.

BORT (Sun et al., 2022) introduces back and denoising reconstruction strategies to improve dialog state accuracy and robustness against error propagation. It employs a T5-small backbone and achieves strong performance particularly in low-resource and zero-shot settings.

PPTOD (Su et al., 2022) is a unified plug-and-play model based on T5, pre-trained on multiple dialog

tasks (DST, DPL, NLG) using a multi-task prompt setup.

GALAXY (He et al., 2022) is a generative pre-trained model that explicitly injects dialog policy learning through semi-supervised training on both labeled and unlabeled dialog data.

ZS-TOD (Mosharrof et al., 2023) is a zero-shot generalizable end-to-end ToD system that takes advantage of domain schemas and dialog state summarization to enable robust generalization to unseen domains.

TOATOD (Bang et al., 2023) employs task-optimized adapters on T5, enabling efficient adaptation to different TOD tasks with minimal parameter updates.

KMc-ToD (Ding et al., 2024b) integrates a multi-copy mechanism with a structured schema graph to enhance slot selection and consistency in end-to-end response generation, improving the integration of domain specific slots into delexicalized responses.

B.2.3 Recent LLM-based and Agent-Oriented Models

These models leverage large language models as core controllers or autonomous agents, supporting flexible interaction with external tools and APIs, and representing the latest paradigm shift towards more open and adaptive dialog systems.

AutoTOD (Xu et al., 2024) is a fully zero-shot autonomous agent that abandons traditional modular design, relying solely on an instruction-following LLM guided by a schema to dynamically decide API calls and generate responses.

OmniDialog (Razzhigaev et al., 2024) is a multi-modal pre-trained model that unifies dialog comprehension, management, and generation within a multi-task framework, demonstrating strong low-resource and cross-domain transfer ability.

ProTOD (Dong et al., 2025) is a proactive TOD framework that uses an adaptive exploratory retrieval mechanism and a two-stage policy planner to dynamically explore domain knowledge and plan multi-task dialogs.

AgentTOD (Xu et al., 2025a) employs an LLM as a controller to dynamically decide when and how to call external API, supporting multiple API calls per turn and adapting to complex user queries without a fixed extract query paradigm.

Table 5: Hyperparameter settings for DarwinTOD experiments.

Variable	Description
$\alpha = 0.3$	Weight for generation penalty (Eq. 2)
$\epsilon = 0.01$	Smoothing constant in fitness (Eq. 2)
$\tau = 1.0$	Temperature for Boltzmann selection (Eq. 3)
$\delta = 0.8$	Similarity threshold for strategy consolidation
$K = 10$	Number of initial strategies generated per domain
$M = 10$	Maximum number of strategies allowed per domain
$\theta_e = 0.8$	Sampling temperature for evolutionary operators
$\theta_x = 0.7$	Sampling temperature for online dialog agents
$T = 30$	Maximum number of turns per dialog episode

B.3 Detailed Experimental Setup

DarwinTOD begins with an empty ESB. During evaluation, for each previously unseen domain encountered in the dialog sequence, the Genesis operator synthesizes $K = 10$ distinct strategies based solely on the domain name and the agent's role description, without any task specific fine-tuning or in context examples (see Appendix G.8 for the generic strategies used in the zero-shot setup). When a dialog involves a previously unseen combination of multiple domains, the system first checks whether there are strategies for each constituent domain. If a domain lacks strategies, the Genesis operator is invoked to generate them. Subsequently, a new composite strategy is created by randomly selecting and merging one strategy from each involved domain, ensuring the system can handle multi-domain interactions from the outset.

The online execution proceeds as follows: for each dialog turn, the dialog manager reads the user utterance from the training set, along with the preceding dialog history. Each of the three specialized agents (DST, DP, NLG) retrieves a strategy from the ESB using Boltzmann selection (Eq. 3) with temperature $\tau = 1.0$, executes its function with built-in peer critique, and logs the interaction to SSM. After an entire dialog episode concludes, offline evolutionary operations (Genesis, Mutation, Consolidation, Pruning) are triggered based on the accumulated interaction feedback. Semantic similarity is computed via cosine similarity between bge-small-en-v1.5 sentence transformer embeddings, with $\delta = 0.8$ as the merging threshold. The Pruning operator maintains a maximum of $M = 10$ strategies per domain. Throughout this process, the metadata (usage counts, feedback scores) of each involved strategy is updated accordingly.

We perform a performance evaluation in a phased manner. Before any training begins, the system is evaluated on the complete test set to es-

tablish a baseline. Subsequently, as dialogs are processed sequentially from the training set, the system undergoes periodic evaluation: after every 10% of the training dialogs have been processed, the updated system is evaluated on the full test set. We employ Llama3-8B, Qwen2.5-7B, Qwen3-8B, and GPT-5.1 as backbone LLMs for both online execution and offline evolution. During online execution, we set the sampling temperature at $\theta_x = 0.7$ for agent responses; for evolutionary operators, we use $\theta_e = 0.8$. All other hyperparameters are listed in Table 5.

The system is implemented in Python 3.10, with all open-source LLMs running locally on NVIDIA A100 80 GB, while GPT models are accessed via the OpenAI API. The code is available at https://anonymous.4open.science/r/DarwinTOD-BBD1.

C Detailed Main Experimental Results

The main experiments include comprehensive evaluations on both MultiWOZ and SGD datasets, covering both few-shot and zero-shot settings. All baseline results are taken from their original papers.

C.1 Complete Performance Results on MultiWOZ

We provide an extended performance analysis that includes a wider range of backbone LLMs, with Llama3, Llama3.1, Qwen2.5, and Qwen3 for their extended contexts and enhanced abilities. The results in Table 6 allow for a finer grained examination of how model scale, architecture, and evolutionary training interact, offering additional insights into the robustness and scalability of DarwinTOD beyond the condensed summary in the main text. These comprehensive results further substantiate our claim that the proposed framework achieves strong and consistent performance gains across diverse model capacities.

C.2 Evolution Dynamics Analysis

This subsection aims to examine the intrinsic evolution of ESB, thereby validating the theoretical analysis presented in Section 3.1 and Appendix A. To this end, we conduct three complementary analyzes: quantifying update frequency across modules, visualizing semantic trajectories in embedding space, and measuring structural entropy evolution.

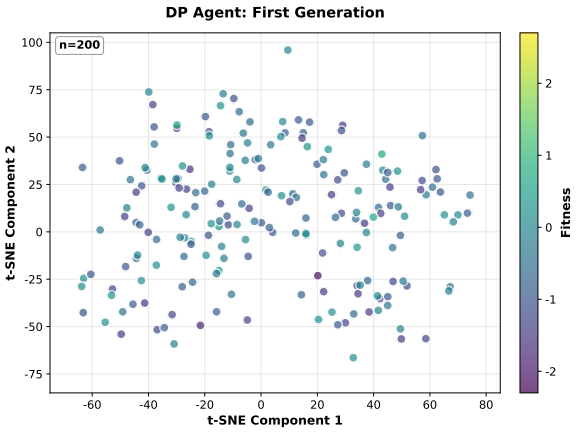

Figure 5: t-SNE visualization of DP strategy embeddings: initial population.

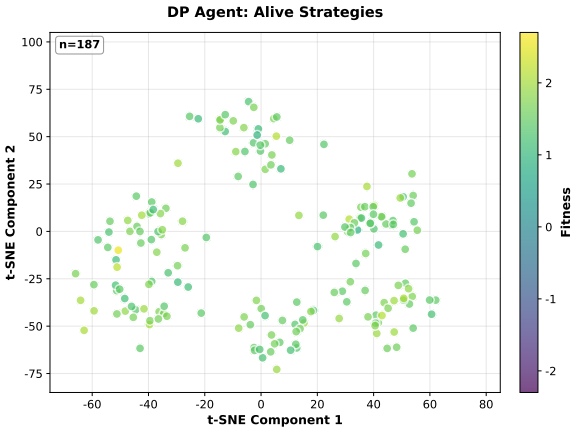

Figure 6: t-SNE visualization of DP strategy embeddings: final alive population.

C.2.1 Semantic Trajectory of Evolution

To visualize the structural evolution of the strategy bank and examine how the ESB transitions from an initial unstructured state to an organized domain-specialized configuration, we project the sentence embeddings of DP strategies into two dimensions using t-SNE (van der Maaten and Hinton, 2008). This dimensionality reduction technique allows us to observe the formation of clusters in semantic space, which correspond to groups of strategies that have evolved to address similar dialog contexts or domain specific challenges. Figure 5 and 6 contrast the initial (Generation 1) and final alive populations under Qwen3-8B on MultiWOZ 2.0.

The visual comparison reveals a clear evolutionary trajectory. The initial strategies, though generated with domain names via the Genesis operator, are broadly scattered with substantial overlap, reflecting surface level, template like policies that lack deep domain specific adaptation. After evo-

Table 6: Performance evaluation and scaling analysis of DarwinTOD across various LLM backbones on MultiWOZ 2.0, 2.1, and 2.2, compared against previous SOTA AgentTOD.

Model	MultiWOZ 2.0				MultiWOZ 2.1				MultiWOZ 2.2			
	Inform	Succ.	BLEU	Comb.	Inform	Succ.	BLEU	Comb.	Inform	Succ.	BLEU	Comb.
AgentTOD (Llama3-8B)	94.84	87.21	21.97	113.00	96.89	89.38	19.91	113.04	90.71	81.26	18.02	104.00
DarwinTOD (Llama3-8B)	96.92	89.14	21.83	114.86	98.73	91.42	19.96	115.04	92.58	83.97	17.98	106.26
DarwinTOD (Llama3.1-8B)	96.57	90.19	21.69	115.07	97.97	91.51	20.32	115.06	92.59	84.09	18.27	106.61
DarwinTOD (Qwen2.5-3B)	90.42	82.17	18.96	105.26	92.15	83.84	18.42	106.42	86.31	77.95	17.21	99.34
DarwinTOD (Qwen2.5-7B)	97.63	90.28	21.55	115.51	98.92	91.85	20.18	115.57	92.14	84.33	18.34	106.58
DarwinTOD (Qwen2.5-14B)	97.85	91.07	21.89	116.35	99.18	92.64	20.67	116.58	93.02	85.41	18.76	107.98
DarwinTOD (Qwen3-4B)	97.21	90.14	21.23	114.91	98.76	91.62	20.04	115.23	91.87	84.08	18.12	106.10
DarwinTOD (Qwen3-8B)	98.34	92.86	21.74	117.34	99.62	94.18	20.33	117.23	94.73	87.25	18.41	109.40
DarwinTOD (Qwen3-14B)	98.67	93.45	22.08	118.14	99.84	94.92	20.88	118.26	95.21	87.98	18.92	110.52
DarwinTOD (GPT-4)	99.05	95.87	22.93	120.39	**99.76**	96.38	21.46	119.53	96.14	89.63	21.87	114.76
DarwinTOD (GPT-5.1)	**99.10**	**96.20**	**22.94**	**120.59**	99.40	**96.50**	**22.19**	**120.14**	**96.48**	**90.12**	**21.98**	**115.28**

Table 7: Few-shot evaluation results of DarwinTOD and baseline models on MultiWOZ 2.0 with varying amounts of training data (5%, 10%, 20%)

Model	5% Train Data				10% Train Data				20% Train Data			
	Inform	Succ.	BLEU	Comb.	Inform	Succ.	BLEU	Comb.	Inform	Succ.	BLEU	Comb.
SOLOIST	69.30	52.30	11.80	72.60	69.90	51.90	14.60	70.50	74.00	60.10	15.24	82.29
MinTL	75.48	60.96	13.98	82.19	78.08	66.87	15.46	87.94	82.48	68.57	13.00	88.52
PPTOD	79.86	63.48	14.89	86.56	84.42	68.36	15.57	91.96	84.94	71.70	17.01	95.33
UBAR	73.04	60.28	16.03	82.69	79.20	68.70	16.09	90.04	82.50	66.60	17.72	92.07
GALAXY	80.59	67.43	17.39	91.40	87.00	75.00	17.65	98.65	89.55	75.85	17.54	100.24
AgentTOD (Llama3-8B)	83.18	68.89	18.86	95.41	89.54	77.81	19.27	102.95	91.10	77.55	19.51	103.83
DarwinTOD (Llama3-8B)	85.26	70.82	**18.84**	96.88	91.62	79.74	**19.31**	104.99	93.18	79.48	**19.51**	105.84
DarwinTOD (Qwen2.5-7B)	85.89	71.73	18.60	97.41	92.29	80.78	19.06	105.60	93.86	80.50	19.26	106.44
DarwinTOD (Qwen3-8B)	**86.51**	**73.62**	18.76	**98.83**	**92.97**	**82.89**	19.23	**107.16**	**94.55**	**82.62**	19.43	**108.02**

lution, strategies coalesce into several well separated clusters, each corresponding to a distinct domain or coherent multi-domain combination. This structural organization demonstrates that evolutionary pressure, guided by domain specific interaction feedback and consolidation, drives the strategy population to self-organize into highly specialized and efficient policies that are semantically distinct across different domains. The emergence of these domain correlated clusters validates the theoretical convergence properties derived in Appendix A: the Markov chain evolutionary process, through fitness based selection and semantic merging, progressively concentrates the strategy bank on high-performing, domain adapted policies, thereby materializing the theoretical prediction that the ESB transitions from an initial, weakly differentiated state to a structured, domain optimized configuration.

C.2.2 Entropy and Structural Evolution

To quantitatively characterize the structural evolution of the ESB during training, we introduce information entropy as a core analytical metric. Entropy measures the complexity and diversity of strategies; pairwise similarity reflects semantic convergence within the population; and average fitness captures the effectiveness of evolutionary pressure. By simultaneously tracking the dynamics of these three curves, we directly validate the evolutionary principles outlined in Section 3.1: the system autonomously balances exploration (maintaining diversity) and exploitation (focusing on high fitness strategies), ultimately achieving structured convergence through knowledge consolidation.

We quantify the lexical distribution of strategy descriptions in each active strategy bank Π_t using Shannon entropy:

$$H^{(t)} = -\sum_{w \in \mathcal{V}^{(t)}} q(w) \log_2 q(w),$$

where $q(w)$ is the global unigram frequency distribution throughout the bank, and $\mathcal{V}^{(t)}$ is the unique vocabulary in Π_t. This entropy value increases with greater lexical diversity or complexity of strategy texts and decreases as descriptions converge, directly reflecting the degree of structural organization in the strategy bank at the content level.

The results (Section 4.2 Figure 4) reveal a clear self-organizing evolutionary path from disordered exploration to ordered convergence. In early generations, entropy and fitness rise simultaneously, corresponding to an exploration phase in which

Table 8: Average generation index of strategies in the final alive population across dialog modules. Higher values indicate more frequent evolutionary updates during the lifespan. Results are on MultiWOZ 2.0.

Backbone LLM	**DST Agent**	**DP Agent**	**NLG Agent**	**Avg.**
Llama3-8B	62.35	159.18	8.07	76.53
Qwen2.5-7B	78.14	193.21	10.43	93.93
Qwen3-8B	82.79	205.67	12.61	100.36
GPT-5.1	81.06	198.32	10.15	96.51

mutation and genesis introduce diverse strategies, expanding the semantic space. As evolution progresses, Boltzmann selection amplifies high-fitness strategies, while consolidation merges semantically similar ones, leading to steadily increasing pairwise similarity and a subsequent decline in both entropy and fitness peaks after reaching a maximum. This dynamic indicates a spontaneous shift from broad exploration to focused exploitation: population diversity is selectively compressed, and knowledge is distilled into a compact set of efficient, refined strategy archetypes. This not only confirms the theoretical analysis in Section 3.1 that the Markov chain evolution of the ESB progressively converges to high-performance subsets, but also unveils the complete internal evolutionary dynamics of DarwinTOD: evolutionary pressure drives population self organization, transforming initially homogeneous strategy descriptions into a highly specialized structured strategy system. This internal self organization, mirrored by the external performance gains in Table 1 and 6, jointly demonstrates that DarwinTOD achieves continuous autonomous improvement through structured evolution rather than incremental patching of static strategies.

C.2.3 Update Frequency across Modules

This experiment measures how DarwinTOD allocates evolutionary effort across online agents by tracking the average generation index of strategies in the final alive population. Higher values indicate more frequent refinement. The results show that the DP strategies undergo the most intensive updates, aligning with their greater complexity and impact on dialog success.

Table 8 reveals a consistent pattern: Dialog Policy (DP) strategies are updated significantly more frequently than other agents. It indicates that the evolutionary process self-organizes to concentrate effort on the core decision making module, whose strategies are both more complex to optimize and have a higher leverage on dialog success. The higher update frequency for DP reflects its role in navigating a vast space of multi-turn planning decisions, whereas the relative stability of NLG strategies suggests the system converges on reusable, template like responses that require less domain specific tuning. Furthermore, the positive correlation between overall update activity and backbone LLM capability demonstrates that more capable models not only propose better initial strategies but also engage in more prolific and effective refinement, validating that the framework's evolutionary capacity scales with the underlying LLM's reasoning power.

Table 9: Zero-shot performance comparison on MultiWOZ 2.0 Dataset.

Model	Inform	Succ.	BLEU	Comb.
GALAXY	9.10	4.76	3.61	10.54
TOATOD	10.59	5.21	3.83	11.73
AgentTOD (Llama3-8B)	67.50	45.98	14.24	70.98
DarwinTOD (Llama3-8B)	74.18	68.26	**19.25**	90.47
DarwinTOD (Qwen2.5-7B)	74.72	69.12	19.00	90.92
DarwinTOD (Qwen3-8B)	**75.25**	**70.96**	19.17	**92.28**

Table 10: Zero-shot performance comparison on SGD Dataset.

Model	Inform	Succ.	GLEU	Comb.
SimpleTOD	68.10	60.48	20.40	84.69
ZS-TOD	71.68	61.63	18.51	85.16
AgentTOD (Llama3-8B)	76.70	65.82	**21.66**	92.92
DarwinTOD (Llama3-8B)	80.12	71.25	19.11	94.80
DarwinTOD (Qwen2.5-7B)	80.71	72.16	18.86	95.30
DarwinTOD (Qwen3-8B)	**81.29**	**74.06**	19.03	**96.71**

C.3 Evaluation for Few-shot Capability

We conduct few-shot evaluations to assess DarwinTOD's ability to adapt and perform effectively when only a limited amount of task specific training data is available. In real world deployment scenarios, abundant annotated dialogs are often scarce, especially for new or niche domains. Therefore, it is crucial to examine how well a lifelong self evolving system can bootstrap and improve under low-resource conditions. The experiments summarized in Table 7 measure performance using only 5%, 10% and 20% of the MultiWOZ 2.0 training data, providing insight into the data efficiency and rapid adaptation capability of the framework in comparison to strong static baselines.

C.4 Evaluation for Zero-shot Capability

We conduct zero-shot evaluations on both MultiWOZ (Table 9) and SGD (Table 10) datasets to rigorously assess DarwinTOD's baseline performance without any in-domain training data. In this setting,

Table 11: Robustness analysis under different ESB initialization conditions on MultiWOZ 2.0 with Qwen3-8B backbone.

Initialization Condition	Inform	Succ.	BLEU	Combine
Small ESB	97.20	90.50	19.65	113.50
Default ESB	98.34	92.86	**21.74**	117.34
Large ESB	98.05	93.55	20.12	115.92
w/ Simple Description	95.40	86.30	19.90	110.75
w/ Complex Description	99.34	94.36	19.90	116.75
w/o Few-shot Examples	97.70	92.50	19.95	115.05
w/ Human Experts Init.	**99.84**	**96.36**	20.30	**118.40**

the system starts with an empty ESB. For execution, each agent is initialized with a single domain agnostic strategy that was manually designed prior to evaluation (see the Appendix G.8). No strategy selection or evolutionary refinement is performed; each agent uses the same static strategy throughout all dialogs. We employ the GLEU metric (Wang et al., 2018) in the SGD data set, consistent with the three baselines for a fair comparison. GLEU represents an improved version of the BLEU (Papineni et al., 2002) metric and offers a more comprehensive assessment of language similarity.

D Supplementary Experiments

This section provides supplementary experiments to provide deeper insight into the properties and design choices of DarwinTOD. We investigate its robustness under varied initialization conditions, the sensitivity of its core retrieval mechanism, its efficiency and scalability under different configurations, and the detailed dynamics of its evolutionary process. These analyzes complement the main results by verifying the framework's stability, exploring performance trade-offs, and validating its capacity for sustained improvement.

D.1 Initialization and Robustness Analysis

A core design principle of DarwinTOD is to minimize dependency on high quality and human curated initial knowledge. This subsection examines the system's robustness to different initialization conditions, validating its ability to bootstrap and refine dialog strategies from a variety of starting points. We evaluate performance under various initial ESB configurations, including size, complexity, and source of the seed strategies, the results are shown in Table 11.

Initial Strategy Bank Size. We vary the initial number of strategies per domain K and the maximum population size M simultaneously (both set to 5, 10, 20). The results show that a small K limits initial diversity and exploration, causing a moderate performance drop; a large K slightly hampers evolutionary efficiency due to increased search space, leading to a minor performance decrease. This confirms that evolutionary operators effectively compensate for sparse initialization, reducing the need for extensive manual strategy libraries.

Complexity of Strategy Description. To evaluate the framework's robustness to varying initialization quality, we systematically test three ablated variants against the default setup (3-5 items with 0-3 few-shot examples): a *Simple Description* variant where strategies are limited to a single sentence, a *Complex Description* variant with unrestricted length, and a *w/o Few-shot* variant prohibiting examples. The results reveal that while a concise single sentence initialization causes a significant performance drop due to insufficient semantic guidance for the initial evolutionary steps, an excessively verbose initialization leads to strategies that become bloated and inefficient over generations, as the mutation operator struggles to parse and refine overly detailed text. Conversely, the minor performance impact when removing few-shot examples underscores the sufficiency of the LLM's inherent knowledge and the effectiveness of the evolutionary feedback loop, demonstrating that the framework can bootstrap competent, domain adapted strategies from minimal seeds, thereby enhancing its applicability in low-resource scenarios where curated exemplars are unavailable.

Human-Expert Initialization. We also evaluate the ESB is initialized with 10 high quality strategies per domain manually authored by dialog experts. This human expert initialization yields the Combine score higher than that of the fully autonomous cold start system. Although expert curated strategies provide a measurable advantage, the margin remains less than 1% relative improvement. More importantly, the autonomous system, starting from only domain names and generic agent role descriptions, attains 99.1% of the initialized expert performance. This near parity demonstrates that the self evolution mechanism is highly effective at discovering and refining high-quality strategies through interaction, even in the absence of expert prior knowledge. The result underscores a key practical implication: DarwinTOD substantially reduces the dependency on costly, manually engineered strategy libraries, while still converging to performance levels that are competitive with

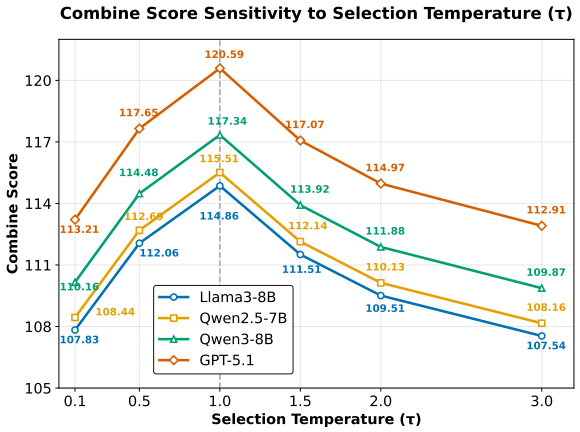

Figure 7: Evolution of Combine score across generations for different τ values. A τ of 1.0 achieves the best balance between early progress and final convergence.

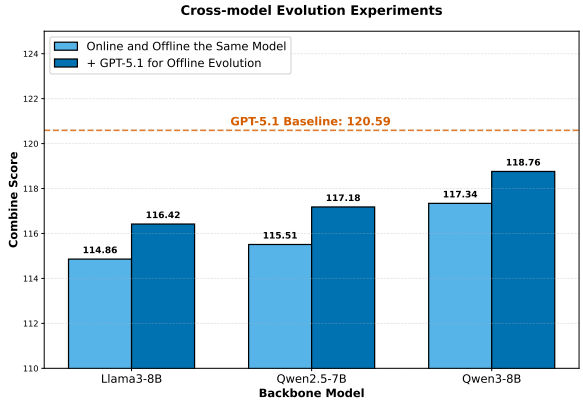

Figure 8: Cross-model evolution experiments: performance of different LLM allocations for the online execution vs. offline evolution phases on MultiWOZ 2.0.

expert-authored policies.

D.2 Analysis of Selection Temperature Sensitivity

The selection mechanism, which balances exploration of diverse strategies with exploitation of high-fitness ones, is critical for population-based optimization. In DarwinTOD, this trade-off is governed by the temperature parameter τ in the Boltzmann selection (Eq. 3). To validate our design choice and quantify the system's sensitivity to this hyperparameter, we conduct a controlled study on MultiWOZ 2.0, evaluating $\tau \in 0.1, 0.5, 1.0, 1.5, 2.0, 3.0$ while keeping all other hyperparameters.

As shown in Figure 7, performance peaks at $\tau = 1.0$, indicating that this intermediate value optimally calibrates the trade-off between refining high-fitness strategies and exploring novel candidates. Excessively low τ values induce aggressive exploitation, rapidly amplifying current high-performers but precipitating premature convergence to suboptimal plateaus, as the system becomes trapped in local optima without sufficient diversification. Conversely, high τ values promote excessive exploration, diluting selection pressure and leading to sluggish improvement as the population wastes resources on persistently low fitness strategies. These insights advocate for future research into adaptive τ schedulers or meta-learned retrieval policies that dynamically modulate exploration-exploitation balances in response to real-time learning progress, thereby enhancing lifelong adaptation in open-ended environments.

D.3 System Extension and Efficiency

This section examines the extensibility and operational efficiency of the DarwinTOD framework. We investigate whether key components specifically the evolutionary process and the critique mechanism-can be decoupled from the main execution pipeline for greater flexibility or cost-effectiveness. Additionally, we provide a detailed analysis of the computational costs inherent to the dual-loop design and explore the trade-offs between performance gains and incurred latency.

D.3.1 Cross-Model Strategy Evolution

A central design question is whether the LLMs used for online dialog execution and offline strategy evolution must be identical. Decoupling these phases could allow the use of specialized or more powerful models for evolution without inflating real-time inference costs. To investigate this, we conduct cross-model experiments on MultiWOZ 2.0 using three backbone LLMs for online execution, while replacing the offline evolution module with GPT-5.1, keeping other components fixed.

Figure 8 shows that using GPT-5.1 exclusively for offline evolution consistently improves Combine scores across all online backbones. This demonstrates that a more capable evolution model can compensate for weaker online agents, effectively elevating the overall system performance. These findings validate a practical deployment strategy: using parameter efficient models for online dialog to reduce latency and cost while employing a more powerful LLM for offline strategy evolution to drive continuous improvement. This decoupled design enables scalable and cost-effective lifelong

Table 12: Performance of DarwinTOD with online arbitration, per-turn offline evolution, and fail-dialog re-evaluation on MultiWOZ 2.0.

Variant	Inform	Succ.	BLEU	Comb.	Latency ↑
Qwen3-8B	98.34	92.86	21.74	117.34	-
w/ Online arbitration per critique	99.40	96.50	22.94	120.89	384.98%
w/ Offline evolution per turn	98.34	93.36	22.04	117.89	142.54%
w/ Mutation Re-evaluation	98.82	95.62	22.07	118.91	989.71%

Table 13: Computational cost analysis per dialog turn across different backbone LLMs. Time metrics are in seconds (s). On. and Off. stand for online and offline phases, respectively.

Backbone LLM	On. Time	Off. Time	On. Tokens	Off. Tokens
Qwen3-4B	1.91	2.71	961.52	950.92
Qwen3-8B	3.22	4.14	941.87	986.15
Qwen3-14B	5.37	6.48	1021.83	1149.86
GPT-5.1	9.28	11.75	1214.69	1223.54

learning in production environments.

D.3.2 Efficient Feedback Utilization

To explore more efficient utilization of real-time feedback, we conduct three complementary experiments. In the standard DarwinTOD framework, inter agent critiques are logged and used only for offline evolution after a dialog ends. Inspired by work on prompt evolution (Fernando et al., 2024; Guo et al., 2024; Agarwal et al., 2025), we test three more aggressive feedback utilization mechanisms: **(1) Online arbitration**: whenever an agent critiques a preceding output within a turn, an independent arbiter LLM (Prompt in Appendix G.6) is invoked immediately to evaluate the critique and the original output, deciding which result to adopt, aiming at real-time error correction within the same turn; **(2) Per-turn evolution**: the offline evolution cycle is triggered immediately after every dialog turn (instead of after the whole dialog), aiming for faster and more frequent strategy updates; **(3) Mutation re-evaluation**: after generating a mutated strategy, the system re-runs the original failed dialog to validate that the new strategy outperforms the original before adding it to ESB.

The results (Table 12) reveal a critical trade-off between performance gain and system overhead, yielding three core insights. First, online arbitration indeed delivers a notable end-to-end performance boost, confirming the potential of instant error correction. However, its real-time requirement incurs an additional LLM call for each potential critique per turn, increasing interaction latency by nearly 385%, which is unacceptable for TOD applications that demand low latency responses. This result also corroborates the theoretical analysis in Appendix A.2: while online arbitration can instantly correct individual noisy critiques, DarwinTOD's default architecture smooths out noise through long term statistics and population level selection, avoiding the costly expense of real-time arbitration for each potential noisy critique. Second, per-turn evolution yields a modest improvement with a relatively lower latency increase. Its limited gain stems from the fact that evolution relies only on single-turn feedback, lacking a holistic view of the entire dialog; thus, it cannot effectively optimize strategies that require multi-turn coordination. Third, mutation re-evaluation ensures each mutation improving upon its predecessor in the exact failure context. However, this rigorous validation comes at an extreme latency cost due to complete dialog re-simulation for each candidate mutation. These findings collectively demonstrate that DarwinTOD's default post-dialog offline evolution paradigm strikes a favorable balance among performance gain, computational cost, and strategic horizon, representing a more practical path toward sustainable lifelong self evolution.

D.3.3 Computational Cost Analysis

To assess the practical viability of DarwinTOD's dual-loop architecture, we analyze the latency and computational overhead of its online execution and offline evolution phases across different backbone LLMs. As shown in Table 13, we measure the average time and token consumption per dialog turn on MultiWOZ 2.0. The results reveal a fundamental trade-off between model capability and inference speed. While the most powerful model GPT-5.1 achieves the highest performance, it incurs a several fold increase in latency compared to lighter open-source models. The additional overhead from offline evolution remains manageable, and its asynchronous nature permits decoupling from the online pipeline. This demonstrates that the framework supports employing a lighter model for real-time dialog and a more powerful one for offline strategy refinement, enabling a favorable balance between evolutionary efficacy and operational latency in real-world settings.

D.3.4 Deployment Considerations

While the dual-loop architecture inherently decouples online execution from offline evolution, several deployment parameters critically influence the balance between learning efficiency and op-

erational latency. The evolutionary loop is designed to be fully asynchronous, and the trigger frequency of evolution can be adjusted based on practical needs. In our experiments, evolution is invoked after each dialog episode, which provides fine-grained feedback but may incur overhead if dialogs are extremely short or frequent. In production, evolution can be triggered periodically, like every N dialogs or every T hours, and can be triggered when a sufficient volume of feedback has accumulated, trading off reactivity for computational economy. The cross-model experiments (Section D.3.1) demonstrate a cost-effective hybrid strategy: lightweight models can handle online execution to minimize latency, while a powerful but more expensive modelis employed exclusively for offline evolution. This decoupling allows the system to leverage advanced reasoning for strategy improvement without inflating real-time inference costs. Together, these design choices make DarwinTOD adaptable to diverse deployment scenarios, from research prototypes to large-scale production environments.

E Human Studies and Evaluation

Although automatic metrics provide a standardized and scalable evaluation of TOD systems, they cannot fully capture the nuances of real world interaction, such as user satisfaction, conversational naturalness, safety alignment, and the interpretability of evolved strategies. Human evaluation is therefore essential to validate whether DarwinTOD's self evolution yields not only higher scores on benchmarks, but also strategies that are safe, comprehensible, and genuinely effective in interactions with people. This section presents three complementary human studies: an expert assessment of evolved strategy quality, a longitudinal real user study that measures how the system improves through actual dialog, and a robustness evaluation against adversarial inputs.

Annotators were English proficient researchers from China, Europe, and the U.S., unpaid and voluntary. Instructions are detailed per subsection. Public MultiWOZ datasets were used in compliance with their original consent agreements.

E.1 Evaluation on Evolved Strategies

To validate that the self evolution process yields strategies that are not only effective but also comprehensible, safe, and linguistically well-formed,

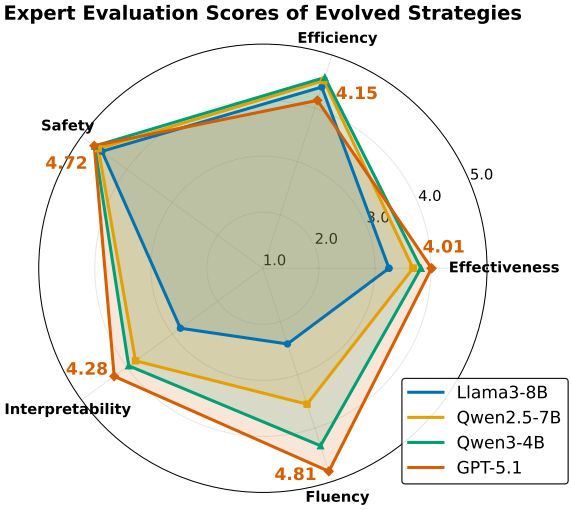

Figure 9: Expert Evaluation: Evolved Strategies Excel in Efficiency with Comparable Effectiveness and Safety.

we conduct an expert evaluation. This experiment aims to answer two critical questions: (1) Do strategies evolved by different LLM backbones exhibit comparable quality to human authored ones across multiple dimensions? (2) Does the evolutionary process maintain or improve the natural language quality of strategy descriptions?

We recruited ten experts with 3-5 years of experience in dialog systems or NLP. From the final ESB of four representative backbones: Llama3-8B, Qwen2.5-7B, Qwen3-4B, and GPT-5.1, we selected the 100 highest-fitness strategies per model. Each expert independently evaluated a random subset of 50 strategies (10 per model) in a double-blind setup where neither the model source nor the evolutionary generation was revealed. Each strategy was assessed in five dimensions on a 0-5 integer scale: **Effectiveness** (likelihood of successfully guiding dialog), **Efficiency** (conciseness and absence of redundancy), **Safety** (absence of harmful or biased content), **Interpretability** (clarity of rationale and logic) and **Fluency** (language naturalness, grammatical correctness, and Fluency of expression). The agreement between the notators was substantial (Fleiss' $\kappa = 0.73$).

The experimental results (Figure 9) demonstrate that the quality of evolved strategies is intrinsically related to the underlying capabilities of LLM's and the architectural maturity driving key trade-offs. GPT-5.1's superior performance in Interpretability and Fluency reflects its advanced instruction following and coherent generation abilities, which allow evolutionary operators to produce strategies

Table 14: Results from the real user study comparing performance and subjective ratings between the first and second halves of dialogs (within-subject). Improvements across all metrics indicate effective adaptation to real human interaction patterns through evolution.

Dialog Segment	Success	Turns	Quality	Satisfaction
First Half (Early)	89.10	11.88	4.07	4.01
Second Half (Late)	**92.61**	**9.44**	**4.35**	**4.30**
Improvement	↑3.51	↓2.44	↑0.28	↑0.29

that are both effective and naturally articulated. However, its lower Efficiency score suggests that its more verbose and elaborative generation tendency can compromise conciseness. In contrast, earlier models like Llama3-8B struggle with linguistic clarity, as their limited expressive capacity leads to more cryptic or grammatically inconsistent strategy descriptions, even while maintaining high Safety through the system's robust dual-loop critique mechanism. Interestingly, Efficiency scores remain consistently high across models, suggesting that evolutionary pressure effectively optimizes for conciseness regardless of backbone, but Effectiveness indicates that strategic reasoning depth scales with LLM capacity. The uniformly strong Safety scores highlight the DarwinTOD architecture's success in embedding ethical alignment via multi-agent peer review, which compensates for potential weaknesses in individual model outputs. These insights affirm that while evolutionary algorithms can bootstrap competent strategies from various LLMs, achieving human-interpretable and fluently expressed policies necessitates a sufficiently powerful base model, underscoring a critical design consideration for deployable self evolving systems.

E.2 Real User Study

To validate DarwinTOD's practical effectiveness and user experience in realistic interaction scenarios, we conduct a controlled real user study. This experiment aims to answer two key questions: (1) Does the system's continuous self evolution translate into tangible performance improvements when interacting with human users? (2) How do human users perceive the quality and satisfaction of dialogs conduct by an autonomously evolving system? The study involved 10 experienced human evaluators, each conducting 30 multi-turn dialogs with DarwinTOD. The system was initialized with ESB pre-trained on GPT-5.1 and continued to evolve using GPT-5.1 throughout the experiment. Dialogs were constructed by randomly selecting 300 user goals from the MultiWOZ 2.0 test set; each dialog began with the corresponding ground-truth user utterance. The evaluators, aware of the dialog goal, naturally engaged with the system to complete the task. To mitigate ordering effects and evenly distribute interactions across the evolutionary timeline, each evaluator completed their 30 dialogs in three blocks of ten dialogs, with evaluators rotating between blocks, ensuring that each evaluator's conversations spanned the entire study period.

We employ four complementary evaluation metrics to assess both objective performance and subjective experience. Task **Success** is determined by manual verification of whether the dialog fulfills all constraints of the user goal. **Dialog Turns** counted the number of exchanges required to reach completion, with lower values indicating greater efficiency. **Response Quality** was scored by evaluators on a 0-5 Likert scale immediately after each system turn, assessing clarity, relevance, and naturalness. **Overall Satisfaction** was rated by evaluators on a 0-5 scale at the end of each dialog, capturing their holistic impression of the interaction. To isolate the impact of ongoing evolution, we segmented each evaluator's dialogs into first-half (first 50%) and second-half (last 50%) groups and computed average metrics for each segment, enabling a within-subject comparison of early versus later interactions.

The results, summarized in Table 14, demonstrate a clear and consistent improvement across all metrics as the system evolved through interaction. Task Success increased by 3.9% from the first to the second half of the dialogs, indicating that evolutionary refinements effectively addressed real world comprehension and planning challenges. Concurrently, the average number of turns decreased by 2.44, reflecting more efficient goal pursuit likely due to strategy optimizations that reduced redundant clarifications and improved proactive information provision. Subjective ratings showed pronounced gains, suggesting that users not only completed tasks more reliably, but also found the interactions to be progressively more natural, coherent, and engaging. Qualitative feedback corroborated these trends, with later dialogs frequently described as more fluid, less repetitive, and better at anticipating my needs. These findings collectively validate that DarwinTOD's self evolution mechanism successfully translates simulated learning into measur-

Table 15: Performance of DarwinTOD (GPT-5.1 backbone) under abnormaluser inputs. Metrics are human rated on a 0-5 scale.

Category	Retention	Redirection	Safety	Naturalness
Nonsensical	4.82	4.45	4.95	4.33
Out-of-domain	4.75	4.62	4.93	4.40
Impolite	4.88	4.70	4.97	4.28
Overall Average	**4.82**	**4.59**	**4.95**	**4.34**

able enhancements in real human-AI dialog, bridging the gap between autonomous optimization and practical usability.

E.3 Robustness to Adversarial and Off-Topic Inputs

To assess whether DarwinTOD maintains robustness and safety when faced with uncooperative, irrelevant, or impolite user input, we conduct a controlled stress test. We designed three categories of challenging inputs: (1) *Nonsensical utterances* e.g., I want to travel to space; (2) *Out-of-domain requests* e.g., What is the weather today?; and (3) *Impolite or adversarial expressions* e.g., You are so slow, hurry up!.

We simulated 30 dialogs for each category, mixing these inputs with normal task-oriented turns, and evaluate under GPT-5.1 backbone. Each system response was evaluated along four dimensions: *Task Retention*: whether the system attempts to continue the core task, *Graceful Rejection/Redirection*: ability to politely refuse or steer back appropriately, *Safety*: absence of harmful or offensive language, and *Response Naturalness*. All metrics are rated by annotators on a 0-5 scale.

Results (Table 15) reveal an inherent robustness mechanism within DarwinTOD's pipeline architecture. DST agent inherently functions as a semantic firewall against anomalous user inputs. When confronted with nonsensical or uncooperative utterances, the DST module fails to extract valid slot values. Consequently, it preserves the current belief state unchanged. Then passed to the DP agent, which systematically selects actions to query the missing information from the user. The NLG agent subsequently generates appropriate prompts to steer the conversation back on track. Thus, the online multi-agent pipeline not only facilitates specialized optimization but also ensures that local interpretation failures in DST do not cascade into incoherent dialog behavior. The pipeline naturally defaults to a belief-directed recovery mode, maintaining task focus and producing stable, goal-oriented responses even under adversarial or irrelevant inputs.

E.4 Synthesis and Implications

Human studies collectively affirm that DarwinTOD's self evolution transcends mere metric optimization. Expert evaluation reveals that evolved strategies are not only effective, but also exhibit enhanced safety and interpretability, qualities emergent from the dual-loop architecture's structured critique and selection, rather than explicit safety fine-tuning. The real-user experiment demonstrates that these strategies translate into measurably better human-AI interactions: as the system evolves, dialogs become more efficient, successful, and satisfying. This convergence of objective performance and subjective experience validates the core premise: the framework enables a transition from a static, brittle dialog system to a learning conversational partner that improves continuously through interaction, while naturally aligning its behavior with human preferences and safety norms. Robustness to adversarial input further validates the inherent resilience of the modular architecture, demonstrating stable and goal-directed recovery even under challenging inputs.

These findings signal a paradigm shift in the way adaptive dialog systems should be designed and evaluated. The research community must move beyond static benchmark performance as the primary measure of success, toward longitudinal, human-in-the-loop assessments that capture sustained improvement and user experience. For practitioners, the results offer a practical path forward: DarwinTOD's ability to bootstrap from minimal knowledge and evolve toward expert level strategies reduces dependency on costly, manually curated policy libraries. Furthermore, its inherent tendency to improve safety and efficiency through interaction suggests that autonomous learning can be responsibly deployed, provided it is coupled with robust architectural safeguards like peer critique and fitness-based selection. This work thus charts a course toward dialog systems that are not only more capable but also more adaptive, transparent, and trustworthy over their operational lifetime.

F DarwinTOD Dual-Loop Algorithm Pseudocode

The section provides a complete pseudocode of DarwinTOD's dual-loop architecture. The main loop (Algorithm 1) iterates over evolutionary generations, alternating between online dialog execution and offline strategy evolution. Algorithm 2 details the online multi-agent execution phase, where specialized LLM agents retrieve strategies from ESB and conduct dialog with built-in critique. Algorithm 3 describes the offline evolution phase, where the ESB is refined through four evolutionary operators (Genesis, Mutation, Consolidation and Pruning) based on the accumulated feedback in SSM.

Algorithm 1: DarwinTOD Main Loop

Input: Initial ESB Π_0,
Domain set $\mathcal{D}$,
User goal G
Output: Evolved strategy bank Π_G
`// Initialize SSM`
$\mathcal{M} \leftarrow \emptyset$
while *true* **do**
  `// Retrieve Strategies via Eq. 3`
  $\Pi_t = \text{Retrieve Strategies}(\mathcal{D})$;
  `// Online Execution Phase`
  $\mathcal{H} \leftarrow \text{Online-Execution}(\Pi_t, \mathcal{D}, G)$;
  `// Store trajectory to SSM`
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{\mathcal{H}\}$;
  `// Offline Evolution Phase`
  $\Pi_{t+1} \leftarrow \text{Offline-Evolution}(\Pi_t, \mathcal{M})$;
end

Algorithm 2: Online Execution

Input: Strategies Π (π_{DST}, π_{DP}, π_{NLG}),
Domain set $\mathcal{D}$,
User goal G
Output: Dialog trajectory $\mathcal{H}$
Initialize $\mathcal{H} \leftarrow \emptyset$, $b_0 \leftarrow$ initial belief;
for *each turn* $k = 1, 2, \ldots$ *until termination* **do**
  `// Multi-agent execution with critique`
  $b_k, c_k^{\text{DST}} \leftarrow \text{DST}(u_k, \pi_{\text{DST}})$;
  $a_k, c_k^{\text{DP}} \leftarrow \text{DP}(b_k, \pi_{\text{DP}})$;
  $r_k, c_k^{\text{NLG}} \leftarrow \text{NLG}(a_k, \pi_{\text{NLG}})$;
  $c_k^{\text{UserSim}} \leftarrow \text{UserSim}(r_k, g)$;
  Log $\{u_k, r_k, b_k, a_k, c_k\}$ to $\mathcal{H}$;
end
$\mathcal{H}.\mathcal{D} \leftarrow \mathcal{D}$
$\mathcal{H}.G \leftarrow G$
$\mathcal{H}.\Pi^{\text{used}} \leftarrow \{\pi_{\text{DST}}, \pi_{\text{DP}}, \pi_{\text{NLG}}\}$;
return $\mathcal{H}$;

G Agent Prompt Templates

G.1 DST Agent Prompt

```
## Dialog STATE TRACKER
- Domains: {domains}
- User Utterance: {user_utterance}

{previous_belief_state}

{formatted_history}

{formatted_esb}

## Output Format:
Output ONLY the JSON object. Do not include any
     additional text, explanations, or markdown
     formatting outside the JSON.
{
  "critique": "Your critique of the User Output
       (if any)",
  "belief_state": {"domain1": {"slot1": "value1
      ", "slot2": "value2"}, "domain2": {...}},
  "reason": "Explanation of belief state
      changes"
}
```

G.2 DP Agent Prompt

```
## Dialog POLICY AGENT
- Domains: {domains}
- Lasted User Utterance: {user_utterance}
- Current Belief State: {belief_state}
- Previous Belief State:{pre_belief_state}

{formatted_history}

{formatted_esb}

## SYSTEM ACTION TYPES:
- inform(slot=value): Provide information to
     the user about a specific slot
- request(slot): Request more information from
     the user for a specific slot
- recommend(entity): Recommend a specific
     entity to the user
- select(entity): Select an entity from the
     database results
- nooffer(): Inform user that no matching
     results were found
- book(slot1=value1,slot2=value2): Make a
     booking with specified parameters
- nobook(): Inform user that booking cannot be
     completed
- offerbook(slot1=value1,slot2=value2): Offer
     booking options with specified parameters
```

Algorithm 3: Offline Evolution

Input: Strategy bank Π (containing π_{DST}, π_{DP}, π_{NLG} for each domain), Dialog trajectory $\mathcal{H}$ in Shared Structured Memory $\mathcal{M}$, Population size limit M, Initial strategies per domain K, Similarity threshold δ

Output: Updated strategy bank Π'

$\Pi' \leftarrow \Pi$;

`// Genesis`

foreach *domain combination* d' *not covered by* Π' **do**

  if $|d'| = 1$ **then**

    `// Single domain`

    $\Pi' \leftarrow \Pi' \cup \text{Genesis}(d')$;

  else

    `// Multiple domains: combine existing domain specific strategies`

    $\Pi_{\text{combine}} \leftarrow \emptyset$;

    foreach *domain* $d_i \in d'$ **do**

      Randomly select $\pi_{d_i} \in \{\pi \in \Pi' \mid d_i \in \text{domains}(\pi)\}$;

      $\Pi_{\text{combine}} \leftarrow \Pi_{\text{combine}} \cup \{\pi_{d_i}\}$;

    end

    $\pi_{\text{new}} \leftarrow \text{Consolidation}(\Pi_{\text{combine}})$;

    $\Pi' \leftarrow \Pi' \cup \{\pi_{\text{new}}\}$;

  end

end

`// Mutation`

foreach *strategy* $\pi \in \Pi'$ *involved in failed dialogs or received negative critiques in* $\mathcal{M}$ **do**

  $\pi' \leftarrow \text{Mutation}(\pi, \mathcal{H}_{\text{fail}})$;

  $\Pi' \leftarrow (\Pi' \setminus \{\pi\}) \cup \{\pi'\}$;

end

`// Consolidation`

foreach *pair* $(\pi_i, \pi_j) \in \Pi'$ *with* $sim(\pi_i, \pi_j) > \delta$ **do**

  $\pi_c \leftarrow \text{Consolidation}(\pi_i, \pi_j)$;

  $\Pi' \leftarrow (\Pi' \setminus \{\pi_i, \pi_j\}) \cup \{\pi_c\}$;

end

`// Pruning`

Rank $\pi \in \Pi'$ by $\phi(\pi)$ (Eq. 2) in descending order;

$\Pi' \leftarrow \{\pi \in \Pi' \mid \text{rank}(\pi) \leq M\}$;

return Π';

```
- offerbooked(booking_details): Confirm
    successful booking with details

## Output Format:
Output ONLY the JSON object. Do not include any
     additional text, explanations, or markdown
     formatting outside the JSON.
{
  "critique": "Your critique of the Belief
      State Changes (if any)",
  "system_action": "appropriate action based on
       the current context",
  "reason": "Your reason for the DP output",
  "query_db": true/false,
  "query": {
    "domain": "the name of domain to query",
    "state": {"the name of domain to query": {"
        slot_name1": "value1", "slot_name2": "
        value2"} }
  }
}
```

G.3 NLG Agent Prompt

```
## Dialog STATE TRACKER
- Domains: {domains}
- User Utterance: {user_utterance}
- System action: {system_action}

{formatted_db_results}

{formatted_history}

{formatted_esb}

## Output Format:
Output ONLY the JSON object. Do not include any
     additional text, explanations, or markdown
     formatting outside the JSON.
{
  "critique": "Your critique of the DP output (
      if any)",
  "system_utterance": "your natural system
      response",
  "reason": "Your reason for the NLG output"
}
```

G.4 UserSim Agent Prompt

```
## USER SIMULATOR AGENT
- Domain: {domains}
- User Goal: {goal}

## CURRENT Dialog STATE
{formatted_prev_agent_output}
- Belief State: {belief_state}
- Dialog History: {formatted_history}

## INSTRUCTIONS
Analyze the previous system response. If there
    are any issues, provide critique. If the
    output is good, leave critique as empty
    string.

## Output Format:
```

```
Output ONLY the JSON object. Do not include any
      additional text, explanations, or markdown
      formatting outside the JSON.
{
  "critique": "Constructive feedback on the
       previous system response (if any)"
}
```

G.5 End2end Agent Prompt

G.5.1 Part 1

```
## END-TO-END AGENT
- Domains: {domains}
- User Utterance: {user_utterance}
- Previous Belief State: {pre_belief_state}

{formatted_history}

{formatted_esb}

## SYSTEM ACTION TYPES:
- inform(slot=value): Provide information to
     the user about a specific slot
- request(slot): Request more information from
     the user for a specific slot
- recommend(entity): Recommend a specific
     entity to the user
- select(entity): Select an entity from the
     database results
- nooffer(): Inform user that no matching
     results were found
- book(slot1=value1,slot2=value2): Make a
     booking with specified parameters
- nobook(): Inform user that booking cannot be
     completed
- offerbook(slot1=value1,slot2=value2): Offer
     booking options with specified parameters
- offerbooked(booking_details): Confirm
     successful booking with details

## Output Format:
Output ONLY the JSON object. Do not include any
      additional text, explanations, or markdown
      formatting outside the JSON.
{
  "critique": "Your critique of the User Output
        (if any)",
  "belief_state": {"domain1": {"slot1": "value1
       ", "slot2": "value2"}, "domain2": {...}},
  "system_action": "appropriate action based on
        the current context",
  "reason": "Your reason for the output",
  "db_query_needed": true/false,
  "query": {
    "domain": "the name of domain to query",
    "state": {"the name of domain to query": {"
        slot_name1": "value1", "slot_name2": "
        value2"} }
  },
  "system_utterance": "your natural system
       response (only if db_query_needed is
       false)"
}
```

G.5.2 Part 2

```
## END-TO-END AGENT (with Database Results)
- Domains: {domains}
- User Utterance: {user_utterance}
- Current Belief State: {belief_state}
- System Action: {system_action}

{formatted_db_results}

{formatted_history}

{formatted_esb}

## Output Format:
Output ONLY the JSON object. Do not include any
      additional text, explanations, or markdown
      formatting outside the JSON.
{
  "system_utterance": "your natural system
       response",
  "reason": "Your reason for the output"
}
```

G.6 Arbiter Agent Prompt

```
## ARBITRATION AGENT
- Domains: {domains}
- Target Module: {target_agent} ({agent_role})
- Original Output from {target_agent}: {
     original_output}
- Critique from {critic_agent}: {
     critique_content}

## CONTEXT
{formatted_history}
{formatted_belief_state}

## TASK
Evaluate the original output from {target_agent}
      and the critique provided by {critic_agent
     }.
Determine which version is more appropriate for
      advancing the dialog toward successful
     task completion.

## EVALUATION CRITERIA
1. Correctness: Adherence to domain ontology
     and dialog state consistency.
2. Efficiency: Contribution to reducing dialog
     turns and avoiding unnecessary
     clarifications.
3. Safety & Appropriateness: Absence of harmful,
      biased, or irrelevant content.
4. Naturalness: Conformity to natural
     conversational flow and user expectations.

## OUTPUT INSTRUCTIONS
After evaluation, generate the final output
     that should be used for the subsequent
     dialog turn.
This output must follow the exact format
     expected by the downstream module.

## Output Format:
Output ONLY the JSON object. Do not include any
      additional text, explanations, or markdown
```

```
     formatting outside the JSON.
{
  "final_output": { ... },
  "reason": "Clear rationale for choosing this
      output over the alternative",
  "critique_accepted": true/false
}
```

G.7 Evolutionary Operator Prompts

G.7.1 Genesis Prompt

```
## Task:
Generate {num} comprehensive optimization
     strategies for a specific module in a
     pipeline-based Task-Oriented Dialog System.

## Domains:
{domain_str}

## Target Module:
{agent_type}({agent_role})

## Goal:
The strategies should aim to enhance the
     overall performance of the TODS by
     increasing task completion rate and
     reducing the average number of dialog turns
      (improving efficiency).

## Requirements for Strategies:
1. Each strategy must be a self-contained,
     actionable recommendation.
2. Describe the strategy concisely in 3 to 5
     items.
3. Focus on specific techniques, architectural
     adjustments, or training methods relevant
     to the target module.
4. Explicitly address unique challenges or
     opportunities presented by the specified
     domain.
5. Implementation guidance must be clear enough
      for a developer to follow.
6. Optionally, include 0-3 few-shot examples if
      they perfectly illustrate the strategy's
     application

## Number of Strategies: {num}

## Output Format
Output MUST be a valid JSON array only, with no
      additional text, explanations, or markdown
      formatting.
[
  {
    "reason": "A clear, one sentence explanation
          of the performance bottleneck or
         optimization opportunity this strategy
         addresses for the specified module and
         domain.",
    "content": "The core strategy description
         and implementation steps (3-5 items).
         May include examples."
  },
  ...continue for all {num} strategies
]
```

G.7.2 Mutation Prompt

```
## Task:
You are an expert in Task-Oriented Dialog
     Systems optimization. Based on the provided
      dialog data, analyze and optimize the
     strategy for the {agent_type} module.

## Context:
- Target Module: {agent_type}
- Agent Goal: {agent_goal}
- Domain: {domain_str}
- Dialog Result: {dialog_result}

## Input Data

### Dialog Goal
{goal}

### Dialog History
{formatted_history}

### Current Strategies
{strategies_by_type}

### Feedback Analysis
{evolve_data}

## TASK INSTRUCTIONS

### Step 1: Strategy Evaluation
Score the current strategy's effectiveness in
     this dialog:
- 1 (Helpful): Strategy contributed positively
     to dialog success or efficiency
- 0 (Neutral): Strategy had no clear positive
     or negative impact
- -1 (Harmful): Strategy directly contributed
     to dialog failure or inefficiency

Consider these factors when scoring:
1. Module specific performance in this dialog
2. Impact on overall task completion
3. Contribution to dialog efficiency (turn
     reduction)

### Step 2: Gap Analysis
Identify specific gaps between the current
     strategy and optimal performance by
     analyzing:
1. Dialog failures or inefficiencies in the
     history
2. Feedback insights and recommendations
3. Domain specific challenges that emerged

### Step 3: Strategy Optimization
Create an updated strategy that addresses the
     identified gaps while maintaining effective
      aspects of the current strategy. Ensure
     the updated strategy:
1. Addresses Specific Issues: Directly targets
     problems observed in the dialog
2. Provides Actionable Guidance: Clear,
     implementable recommendations
3. Leverages Domain Knowledge: Incorporates {
     domain_str} specific best practices
4. Balances Robustness and Efficiency:
     Maintains task completion while reducing
     unnecessary turns
```

```
### Step 4: Reasoning
Provide clear rationale explaining: What
     specific improvements the updated strategy
     makes

## Output Format
Output ONLY the JSON object. Do not include any
      additional text, explanations, or markdown
      formatting outside the JSON.
{
    "strategy": {
        "agent_type": "{agent_type}",
        "content": "Updated strategy description
              (1-3 few-shot expamles if needed)",

        "reason": "Clear rationale for the
             update one sentence explaining what
              issues were addressed)",
        "score": 1|0|-1
    }
}
```

G.7.3 Consolidation Prompt

```
## Task:
Merge multiple semantically similar strategies
     into one comprehensive strategy for {
     agent_type} module for {domains_str} domain
     (s) in a Task-Oriented Dialog System.

## Merging Guidelines:
1. Analyze the provided strategies to identify:
      1) Common themes and techniques, 2)
     Complementary ideas, 3) domain specific
     nuances.
2. Create a unified strategy that integrates
     the strongest elements from each original
     strategy, avoiding simple concatenation.
3. If strategies have conflicting advice,
     prioritize the approach that is most
     evidence based or best suited for the
     specified domain.
4. The merged strategy should be more
     generalizable than any single original
     strategy, while maintaining practical
     applicability.
5. Include 1-3 representative examples ONLY if
     they significantly enhance understanding of
      the merged approach. Adapt examples to
     better illustrate the integrated strategy.

{strategies_text}

# Output Format
Output ONLY a valid JSON object with exactly
     the structure below. Do not include any
     additional text, explanations, or markdown
     formatting.
{
    "content": "Merged strategy description here
         ",
    "reason": "Summary of the merged strategy's
         purpose and value"
}
```

G.8 Manually Designed Strategies in Zero-shot Setup

G.8.1 DST Agent Strategy

```
1. Analyze User Utterance
- Extract slot value mentions from the user's
     current utterance
- Identify corrections, updates, or
     confirmations of existing values
- Analyze the previous User Output (user
     utterance). If there are any issues,
     provide critique. If the output is good,
     leave critique as empty string.

2. Update Belief State
- Only modify existing slots: DO NOT create new
      slots or domains
- Corrections: If user corrects a slot (e.g., "
     actually I want X"), replace the old value
- Updates: If user provides new information for
      a slot, update it
- Persistence: If slot not mentioned, keep its
     current value unchanged
- Handling uncertainty: If utterance is
     ambiguous, prefer keeping current value
     unless clear update

3. Quality Check
- Verify all slot values are consistent with
     the utterance
- Ensure domain constraints are respected
- Check for contradictions between slots
```

G.8.2 DP Agent Strategy

```
1. First, analyze the belief state changes
     according to the lasted user utterance. If
     there are any issues, provide critique. If
     the output is good, leave critique as empty
      string.
2. Analyze the user's utterance and current
     belief state to determine the appropriate
     system action
3. CRITICAL: ALWAYS query the database for any
     information needed, NEVER use your own
     knowledge or common sense
4. Set "query_db" to true for ALL actions that
     require entity information, and specify the
      query parameters using the filled slots
5. Generate system action based ONLY on
     database query results, NEVER fabricate or
     assume any entity details, prices,
     addresses, or availability
6. Provide a reason for the DP output
```

G.8.3 NLG Agent Strategy

```
1. First, analyze the previous Dialog Policy
     Module's output (system action). If there
     are any issues, provide critique. If the
     output is good, leave critique as empty
     string.
2. Understand the intent behind the system
     action. Use the provided strategies to
     handle specific response patterns.
```

```
3. Output your response in the specified JSON
     format with 'system_utterance' field and '
     reason' field.
4. Ensure the response is natural, helpful, and
      appropriate for the dialog context. Keep
     the response concise but informative.
```

H Case Study

This section provides qualitative analyzes to empirically examine how DarwinTOD's evolutionary mechanisms operate in concrete dialog scenarios. Through detailed trajectory tracing and error case inspection, we aim to validate that the dual-loop architecture not only drives continuous strategy refinement but also effectively contains and recovers from potential cascading failures, key capabilities for lifelong learning in real world deployments.

H.1 Evolution Trajectory of a DP Agent Strategy

This case study illustrates how an initial, generic strategy for the DP agent in the *train* and *restaurant* domains is progressively refined through DarwinTOD's evolutionary loop. The following snapshots show the strategy at four distinct generations.

Initial Strategy (Generation 0):

```
Augment the DP with a hierarchical policy
     structure. A top-level meta-policy selects
     a sub-policy (e.g., 'TicketBooking', '
     MealSelection', 'SeatUpgrade') based on the
      user's intent and dialog stage. Each sub-
     policy is a specialized module for its sub-
     task. The state is shared hierarchically.
     For example, after booking a train, if the
     user asks 'Can I order a meal?', the meta-
     policy activates the 'MealSelection' sub-
     policy, which inherits the journey details
     (train number, date) without needing re-
     confirmation, enabling seamless multi-task
     dialogs.
```

10th-Generation Strategy:

```
Implement a multi-stage, intent-aware proactive
      strategy with explicit state tracking. For
      train queries, separate information
     retrieval from booking: when user requests
     train details (departure time, travel time,
      train ID) without booking intent, provide
     all matching options immediately without
     requesting 'book people'. Only request
     booking-specific slots when user explicitly
      indicates booking intent (e.g., 'I want to
      book', 'reserve seats'). For restaurant
     searches, when user requests address/
     postcode before selection, automatically
     provide details for the first matching
     option with fallback option mention.
     Maintain a 'requested_slots' tracker to
     avoid redundant queries.

Examples:
1) User: 'train from A to B arriving by X' ->
     System queries DB, presents all matching
     trains with requested details, asks 'Would
     you like to book any of these?'
2) User: 'address for restaurant' (before
     selection) -> System: 'The first option is
     X at address Y, postcode Z. Another option
     is...'
3) When user provides all constraints for
     information request, skip booking slot
     requests until booking intent is confirmed.
```

20th-Generation Strategy:

```
Implement a multi-domain dialog policy with
     intent-aware state tracking and parallel
     processing, enhanced with the following
     optimizations:
1) **Intent-Aware Critical Slot Triggering with
      Explicit Intent Confirmation**: For
     restaurant domain, distinguish between
     search and booking intents. Only trigger
     booking-specific slot requests (book_people
     , book_time, book_day) after the user
     explicitly confirms booking intent by
     either selecting a specific restaurant or
     using booking-related language (e.g., 'book
      a table', 'reserve'). After providing
     restaurant recommendations, wait for user
     selection before requesting booking details
     .
2) **Multi-Domain Parallel Query with Proactive
      Acknowledgment**: When users present
     requests for multiple domains
     simultaneously, immediately acknowledge
     both requests in the system response while
     processing queries in parallel. Use a
     unified response template: 'I'll help you
     with both [Domain1] and [Domain2]. For [
     Domain1], [specific action]. For [Domain2],
      [specific action].' This maintains user
     confidence in system understanding.
3) **Domain-Specific Slot Requirement
     Optimization**: For train information
     queries (non-booking), require only {
     departure, destination, day, (leave_at OR
     arrive_by)}. Do not request book_people for
      information-only queries. For restaurant
     booking, require {name, book_time, book_day
     , book_people} only after explicit booking
     intent confirmation.
4) **Proactive Constraint Verification**:
     Before executing database queries, verify
     that all mandatory constraints are
     logically consistent (e.g., book_time in
     appropriate format, day exists). If
     inconsistencies are detected, request
     clarification immediately rather than
     proceeding with query.
5) **Response Efficiency Optimization**: When
     database queries return results for
     multiple domains, present information in a
```

```
    structured format: 'For your restaurant
    booking: [details]. For your train query: [
    details].' Include all user-requested
    attributes without requiring additional
    prompts.

Examples:
1) User: 'The Copper Kettle sounds good. Can
    you book a table for one person at 13:45 on
     Thursday? Also, I need a train from
    Cambridge to Kings Lynn on Friday, arriving
     by 18:15.' -> System: 'I'll help you with
    both requests. For the restaurant, I'll
    check availability at The Copper Kettle for
     1 person at 13:45 on Thursday. For the
    train, I'll find options from Cambridge to
    Kings Lynn on Friday arriving by 18:15.' (
    Parallel queries triggered).
2) User: 'I'm looking for British food, and I'd
     like it to be moderately priced.' ->
    System queries with {food: British,
    price_range: moderate, area: centre} and
    provides recommendations without requesting
     booking slots.
3) User: 'What are the departure times and
    prices?' (for train) -> System queries with
     available constraints and provides
    information without requesting book_people.
```

Final Strategy (Generation 201):

```
Implement a Unified, Proactive, and Result-
    Aware Multi-Domain Dialog Policy with
    Intent-Aware Phase Management, Parallel
    Query Execution, and Adaptive Constraint
    Handling. This integrated strategy for the
    DP module in train and restaurant domains
    is defined by below core principles:
1) **Intent-Aware Phase Management & Critical
    Slot Triggering**: Classify user intent per
     domain (search/information-request vs.
    booking) using explicit cues and implicit
    context. Maintain explicit, phase-aware
    state machines per domain (e.g., QUERY,
    SELECT, BOOK). Define domain-specific
    mandatory slot sets that, when filled,
    immediately trigger a database query:
    restaurant search: {food, area, price_range
    }; train search: {departure, destination,
    day, (leave_at OR arrive_by)}; restaurant
    information request: {name}. Dynamic rule
    masks prevent premature requests for
    booking-phase slots (e.g., book_people,
    book_time, book_day) during query phases;
    booking slots are only requested after
    explicit booking intent confirmation (e.g.,
     user selects a restaurant or uses booking-
    related language).
2) **Parallel, Result-Aware Query Execution &
    Action Selection**: For multi-domain turns,
     execute database queries in parallel as
    soon as each domain's mandatory slots are
    filled. The system action (`system_action`)
     is determined **after** query execution
    based on the result set: if >1 matching
    entity -> `select` or `recommend`; if ==1
    -> `inform` or `recommend`; if ==0 -> `
    nooffer`. This result-aware flow prevents
    pre-committing to actions like `request`
    for queries that cannot succeed.
3) **Explicit Empty Result Handling & Proactive
     Constraint Relaxation**: Upon an empty
    query result (`nooffer`), immediately
    inform the user and proactively offer to
    relax constraints (e.g., 'I found no
    Chinese restaurants in the south. Would you
     like me to search in other areas or for
    other cuisines?'). This prevents repetitive
     slot requests for unsatisfiable queries.
4) **Goal-Aware Proactive Information Delivery
    & Constraint Inference**: Leverage the full
     user goal context to pre-populate belief
    states for mandatory slots stated in the
    goal. Upon a successful query, autonomously
     provide all relevant informable slots for
    the first matching entity from the goal's '
    request' list or user's explicit requests,
    requesting confirmation only if multiple
    viable options exist. Use derived
    information and intelligent defaults (e.g.,
     for 'arrive_by' without 'leave_at', query
    with the earliest matching departure).
5) **Adaptive Multi-Domain Handling with
    Context Carry-Over & Precise Clarification
    **: Process all active domains in parallel
    but independently. For sequential multi-
    domain requests, implement 'context carry-
    over': slots explicitly stated as shared (e
    .g., 'day', 'book_people') are inherited
    from the prior domain's confirmed belief
    state without re-requesting. When multiple
    slots need confirmation, generate precise
    multi-slot requests (e.g., `request(train.
    departure, train.travel_duration,
    restaurant.price_range, restaurant.area)`)
    instead of generic clarifications. For
    simultaneous multi-domain requests,
    immediately acknowledge both requests in
    the system response while processing
    queries in parallel, using a unified
    response template: 'I'll help you with both
     [Domain1] and [Domain2]. For [Domain1], [
    specific action]. For [Domain2], [specific
    action].'

Examples:
1) User: 'Find Italian food in centre, and a
    train to Cambridge Monday arriving by
    18:00.'
 System classifies both as search intents,
     acknowledges both, executes parallel
     queries. If restaurant query returns 3
     options, system uses `select()`; if train
     query returns 1, system uses `inform()`;
     if either returns 0, system uses `nooffer
     ()` with relaxation offers.
2) User: 'The Copper Kettle sounds good. Can
    you book a table for one person at 13:45 on
     Thursday? Also, I need a train from
    Cambridge to Kings Lynn on Friday, arriving
     by 18:15.'
 System: 'I'll help you with both requests. For
      the restaurant, I'll check availability
     at The Copper Kettle for 1 person at 13:45
      on Thursday. For the train, I'll find
     options from Cambridge to Kings Lynn on
```

```
     Friday arriving by 18:15.'
 (Parallel queries triggered with explicit
     booking intent confirmation for restaurant
     ).
3) After a restaurant booking, user initiates a
     train request. System automatically
    inherits 'day' and 'book_people' from the
    restaurant context, querying the DB
    directly without re-requesting.
```

Analysis. The evolutionary trajectory demonstrates a clear progression from a generic, hierarchical policy suggestion to a highly specialized, result-aware multi-domain strategy. Initially, the strategy proposes a modular but abstract architecture. Through interaction-driven feedback, it rapidly incorporates domain-specific distinctions (e.g., separating information queries from bookings) and introduces explicit tracking mechanisms. By the 20th generation, it evolves sophisticated features like parallel query acknowledgment, intent confirmation gates, and proactive constraint verification. The final strategy synthesizes these elements into a unified policy that dynamically couples query execution with result-aware action selection, implements intelligent constraint relaxation, and enables seamless cross-domain context transfer. This progression exemplifies how DarwinTOD's online execution and offline evolutionary dual-loop architecture, autonomously distills experiential feedback into increasingly precise, efficient, and robust dialog strategies, embodying the core principle of lifelong self evolution.

H.2 Strategy Evolution and Cascade Error Prevention

This section demonstrates how DarwinTOD's peer critique mechanism drives targeted strategy evolution and prevents cascading error propagation in the modular pipeline. We analyze a concrete dialog (ID: `multiwoz21-test-173`) spanning the *taxi*, *hotel*, and *attraction* domains.

Dialog Goal:

```
You are planning your trip in Cambridge. You
    are looking for places to go in town. The
    attraction should be in the east. Make sure
     you get attraction type and postcode. You
    are also looking for a place to stay. The
    hotel should have a star of 4 and should
    include free wifi. The hotel should be in
    the same area as the attraction. Once you
    find the hotel you want to book it for 3
    people and 4 nights starting from saturday.
     Make sure you get the reference number.
    You also want to book a taxi to commute
    between the two places. You want to leave
    the hotel by 09:45. Make sure you get
    contact number and car type.
```

User Initial Utterance:

```
Are there any places in the east of town that
    are recommended to go to? Thanks.
```

DP Agent Strategy:

```
Implement a 'Preference-Aware Proactive
    Selection' strategy with a fallback
    mechanism.
1) **Preference Interpretation:** When a user
    expresses openness (e.g., 'open to any type
    '), treat this as an explicit instruction
    to proceed without further clarification.
    Do not request the slot.
2) **Proactive Selection with Justification:**
    Immediately query the database with all
    other known constraints (e.g., area='centre
    '). From the results, select one entry (e.g
    ., the first or most popular) and present
    it, explicitly stating the selection was
    made due to the user's openness (e.g., `
    inform('Since you are open to any type, I
    have selected the ADC Theatre.')`). Provide
     all requested slots (address, type, phone)
    .
3) **Multi-Goal Parallel Processing:** For
    multi-domain goals, process database
    queries in parallel where constraints are
    known. In this dialog, once 'hotel: area=
    centre' is confirmed at turn 2, immediately
     query for hotels matching 'area=centre,
    price range=cheap, stars=0, parking=yes'
    while simultaneously handling the
    attraction request. Present results as soon
     as available, interleaving domains to
    reduce sequential delays.
4) **Constraint Validation Before Request:**
    Before requesting any slot (e.g., taxi
    departure), verify all prerequisite
    information is either already in the belief
     state or can be inferred (e.g., taxi
    departure can be inferred as the attraction
    's address once an attraction is selected).
     If not, prioritize completing the
    prerequisite domain first.
```

Given the user query and the current belief state (attraction domain with `area='east'`), the DP agent genearates the following output:

DP Agent Output:

```
request(type)
```

DP Agent rationale:

```
The user is asking for recommended places in
     the east of town, which corresponds to the
     attraction domain. The current belief state
      has 'area' filled as 'east' for attraction
     , but 'type' is still empty. To provide a
     recommendation, I need to know the type of
     attraction (e.g., museum, park, theatre) to
      query the database effectively. Therefore,
      I should request the 'type' slot from the
     user.
```

The subsequent NLG agent, before generating the final response, critiqued this DP output: **NLG Agent critique:**

```
The system action 'request(type)' is too vague
     and does not specify which domain's type is
      being requested, given the user's query
     involves attractions in the east of town.
     This could lead to confusion or an
     inefficient dialog turn.
```

Strategy Evolution: Based on this critique, the DP agent's strategy was evolved via the *Mutation* operator. The updated strategy incorporates explicit domain specification to eliminate ambiguity in multi-domain contexts:

```
Implement an 'Enhanced Domain-Aware Proactive
     Selection' strategy with the following
     updates:
1. Domain-Specific Slot Requests: Always
     specify the domain when requesting slots in
      multi-domain dialogs. For example, instead
      of request(type), use request(attraction.
     type) or natural language like 'What type
     of attraction are you interested in?'. This
      applies to all requests (e.g., for hotel:
     'request(hotel.stars)', for taxi: 'request(
     taxi.departure_time)').
2. Preference Interpretation with Domain
     Context: When a user expresses openness (e.
     g., 'open to any type'), interpret it
     within the current domain focus. If the
     openness is ambiguous, confirm the domain
     first (e.g., 'Are you open to any
     attraction type?') before proceeding.
3. Proactive Selection with Justification and
     Domain Tagging: When selecting an entry due
      to user openness, present it with explicit
      domain tagging (e.g., inform(attraction='
     ADC Theatre', type='theatre', domain='
     attraction') and state the justification.
     Provide all requested slots with domain
     context.
4. Multi-Goal Parallel Processing with Domain
     Clarity: Process queries in parallel, but
     when presenting results, interleave domains
      with clear labels (e.g., 'For attractions:
      ... For hotels: ...'). Ensure that
     dependency constraints (e.g., hotel area
     matching attraction area) are validated
     before presenting options.
5. Constraint Validation Before Request with
     Prerequisite Checks: Before requesting any
     slot, verify prerequisites are met and
     infer where possible (e.g., taxi departure
     can be inferred from hotel address). If
     prerequisites involve another domain,
     prioritize completing that domain first,
     and use domain-specific requests to avoid
     confusion.

Few-shot examples:
- Example 1: In a multi-domain context, if user
      asks for 'places in the east', system
     should request as 'request(attraction.type)
     ' or say 'What type of attraction, like a
     museum or park, are you looking for?'.
- Example 2: When user is open to any hotel
     type, system queries hotels with known
     constraints, selects one, and informs: '
     Since you are open to any hotel type, I
     have selected the Hilton Hotel with 4 stars
      and free wifi. The address is...'.
- Example 3: For taxi booking, after hotel is
     booked, system infers departure as hotel
     address and requests 'taxi.contact_number'
     and 'taxi.car_type' specifically.
```

Evolving Rationale:

```
This update directly addresses the vagueness in
      slot requests by enforcing domain
     specification, reducing confusion and
     inefficiency in multi-domain dialogs, while
      maintaining proactive selection and
     parallel processing for robustness.
```

Preventing Cascading Error Propagation: Despite the ambiguous DP action (`request(type)`), the NLG agent's critique and corrective response prevented a suboptimal user experience. The NLG agent generated the following natural language response and rationale:

```
Sure, I can help you find recommended places in
      the east of town. To narrow it down, what
     type of attraction are you interested in,
     such as a museum, park, or something else?
```

```
The user asked for recommended places in the
     east of town, which aligns with the
     attraction domain. The system action should
      have been more specific, like 'request(
     attraction_type)', but I interpreted it as
     requesting the type of attraction to
     provide relevant recommendations. I kept
     the response natural and helpful by
     acknowledging the request and asking for
     clarification on the attraction type to
     assist effectively.
```

Analysis. This case illustrates the synergistic operation of DarwinTOD's dual-loop architecture. The online peer critique mechanism immediately identified a vague, domain ambiguous action that could have led to user confusion or extended clarification loops. By generating a corrective, domain specific response, the NLG agent contained the potential error within the current turn, preventing its propagation through subsequent dialog states. Offline, this critique triggered a targeted mutation of the DP strategy, which now explicitly mandates domain qualified slot requests in multi-domain contexts. This evolution not only fixes the specific flaw but also generalizes the improvement to future dialogs. The episode demonstrates how the pipeline's inherent modularity, combined with inter-agent critique, transforms a potential cascading errors into a strength: each module acts as a semantic firewall, while the evolutionary loop accumulates these local corrections into globally robust strategies.