DY-VOICEBENCH: STATEFUL WORLDS AND LIVING USERS FOR EVALUATING DYNAMIC SPOKEN DIALOGUE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Conversational agents are increasingly expected to complete tasks through interaction rather than generate isolated responses. However, many existing evaluations still treat dialogue as static prediction, where future turns, tool contexts and user behaviors are fixed and the model is not fully accountable for the consequences of its own trajectory. This is insufficient for spoken task completion, where model utterances and actions may change the external world, the user's ability to cooperate, and the episode-local knowledge available in later turns. We introduce **Dy-VoiceBench**, a benchmark for evaluating dynamic spoken dialogue agents through executable closed-loop episodes. Each episode specifies hidden states, visible observations, model-side and user-side actions, environment feedback, transition rules, process rubrics, and terminal success criteria. Rather than matching a reference transcript, Dy-VoiceBench scores the state trajectory induced by the evaluated model. It covers three complementary dynamics: *world-state dynamics* for executable task completion and user-action coordination, *user-state dynamics* for task-consequential spoken empathy and cooperation maintenance, and *knowledge-state dynamics* for acquiring, preserving, revising, and applying episode-local knowledge. Dy-VoiceBench contains 614 scenarios and 1,632 executable world-state action instances, providing a diagnosable testbed for state-centric spoken dialogue evaluation. A demo is available at `https://dyvoicebench.github.io/`.

1 INTRODUCTION

Conversational agents are moving from isolated response generation toward interactive task completion (Liu et al., 2025a; Wei et al., 2026; Schick et al., 2023). In service, support, device-control, and decision-assistance settings, an agent must not only respond to the user's current utterance, but also guide the interaction toward a correct and verifiable outcome. This may require calling tools, coordinating with the user, interpreting feedback, revising assumptions, and verifying that the underlying problem has been resolved. A successful response is therefore not merely fluent or relevant; it should help move the interaction state in the right direction.

Many existing spoken dialogue evaluations (Yan et al., 2025; Li et al., 2026) do not directly assess this ability. They often treat interaction as a sequence of static prediction problems: the model receives an input, dialogue history, or tool-use context, and is evaluated on the next response or action. Even in multi-turn settings, future user turns may be fixed, previous assistant turns may not come from the evaluated model, and the environment may not update according to the model's behavior. Such settings can assess response quality and contextual understanding, but provide limited evidence about whether a model can manage the consequences of its own interaction trajectory.

This limitation is especially important when task success is state-dependent. In realistic interactions, the agent and the user often control different parts of the task. The agent may access backend tools, logs, permissions, service records, or remote configurations (Patil et al., 2023; Qin et al., 2023), while the user controls local devices, physical actions, visible indicators, or private information. A model may know the right solution but still fail if it does not elicit necessary information, guide executable user-side actions, recover from invalid operations, or verify the final state.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

State dependence is not limited to the external environment. Spoken interaction can also change the user's affective and cognitive state (Zhang et al., 2025), as well as the knowledge available for solving the task. A user may become more or less confused, overloaded, trusting, cooperative, or willing to continue depending on how the agent responds. These user-state changes are closely related to spoken empathy: in task-oriented voice interaction, an empathic response should not only be supportive in tone, but also help reduce cognitive burden, preserve or repair trust, improve felt understanding, and support cooperation or a satisfactory conclusion. Task-relevant knowledge can also evolve: rules, constraints, exceptions, and evidence may be introduced or revised across turns, requiring the model to update episode-local conclusions rather than rely on a fixed context.

We introduce **Dy-VoiceBench**, a benchmark for evaluating dynamic spoken dialogue agents through executable, stateful interaction episodes and targeted knowledge-state tasks. In its closed-loop episodes, the evaluated model's own utterances or actions can causally affect hidden state variables, and the updated state can affect later user behavior, environment feedback, observations, or success conditions. These episodes specify hidden states, visible observations, model-side and user-side action spaces, environment feedback, transition rules, process rubrics, and terminal success criteria.

Dy-VoiceBench is organized around three complementary forms of dynamics. **World-state dynamics** evaluate whether the model can move an external environment toward a target state through tools, user-side actions, observations, and verification. **User-state dynamics** operationalize spoken empathy as task-consequential user-state regulation: whether the model can maintain or improve simulated user states such as trust, cognitive load, cooperation, felt understanding, disclosure tendency, and willingness to continue through both what it says and how it is delivered. **Knowledge-state dynamics** evaluate whether the model can acquire, preserve, revise, and apply knowledge introduced within the episode rather than assumed from pretraining. The world- and user-state tracks use behavior-dependent closed-loop interaction; the knowledge-state track provides complementary single-stage application and multi-stage preservation or revision tests. Each track fixes a primary evaluation target: external task completion, empathic user-state regulation, or episode-local knowledge use.

To support this evaluation, Dy-VoiceBench combines two mechanisms. *Stateful Scenario Synthesis* constructs structured scenarios with explicit state specifications, executable transitions where applicable, and track-specific assertions or scoring rubrics. *Living User Simulation* supports the world- and user-state tracks with a state-conditioned spoken user simulator whose utterances, local actions, and latent state updates depend on the evolving scenario state and the model's previous behavior, rather than on a fixed script. In the user-state track, both wording and vocal delivery inform state updates, which condition the user's next response and speech style. We assess simulator and scoring reliability through human audits and compare simulated user states with human self-reports in controlled role-play interactions.

Our contributions are:

- We formulate dynamic spoken dialogue evaluation around world, user, and knowledge states, with track-specific interaction protocols and evaluation targets. We introduce Dy-VoiceBench with 614 scenarios across three tracks and 1,632 executable tool/action instances in the world-state track.
- We propose a scenario construction and user simulation framework that supports state-driven success criteria, capability separation between agent and user, and diagnostic process evaluation.
- We evaluate eight spoken or speech-enabled models, revealing distinct capability profiles across tracks and substantial gaps in executable task completion despite strong user-state or knowledge scores.

2 Related Work

Spoken and full-duplex dialogue evaluation. Recent speech-model benchmarks have moved beyond ASR toward spoken instruction following, audio reasoning, paralinguistic understanding, and speech-to-speech interaction, as represented by AIR-Bench, AudioBench, VoiceBench, and Dynamic-SUPERB (Yang et al., 2024; Wang et al., 2025; Chen et al., 2024; yu Huang et al., 2024). Full-Duplex-Bench evaluates pause handling, backchanneling, turn-taking, and interruption management (Lin et al., 2025); Full-Duplex-Bench-v3 pairs real human speech containing disfluencies with chained

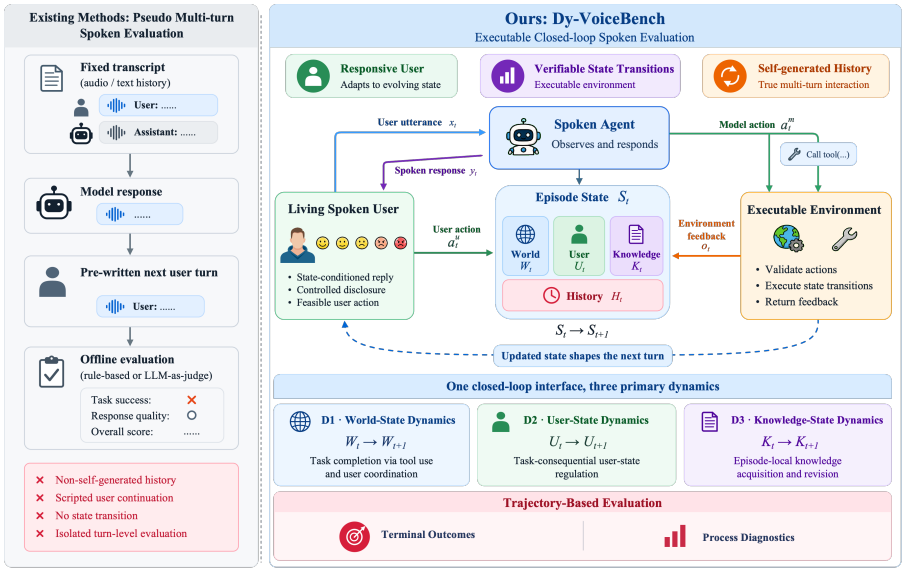


Figure 1: Framework of Dy-VoiceBench.

API calls to evaluate tool-use accuracy, latency, and turn-taking (Lin et al., 2026). Dy-VoiceBench evaluates how spoken responses and actions jointly change executable world states, latent user states, and episode-local knowledge throughout an interaction.

Dynamic and state-based agent evaluation. A related line of work evaluates LLM agents in executable or tool-mediated environments, including AgentBench, WebArena, OSWorld, MINT, BFCL, τ-bench, and τ^2-bench (Liu et al., 2025b; Zhou et al., 2024; Xie et al., 2024; Wang et al., 2024; Xu et al., 2025b; Yao et al., 2024; Barres et al., 2025). These benchmarks require agents to interact with environments, use tools or APIs, and complete multi-step tasks, with τ-bench-style settings being closest to our state-based evaluation. τ^3-Bench adds knowledge-intensive tool use through τ-Knowledge and full-duplex task completion through τ-Voice (Ray et al., 2026). However, their knowledge remains pre-existing rather than introduced and revised within the episode, and their user simulators do not model latent user-state dynamics shaped by the agent's own behavior. Dy-VoiceBench jointly evaluates executable world-state transitions, interaction-dependent user-state regulation, and episode-local knowledge acquisition and revision.

3 METHOD

Dy-VoiceBench evaluates spoken dialogue agents through executable closed-loop episodes. As shown in Figure 1, each episode connects three components: an evaluated spoken agent, a state-conditioned living user simulator, and an executable environment. The agent observes only visible dialogue history, user speech, and environment feedback; its own responses and actions determine later user behavior, state transitions, and final outcomes. Our framework supports three primary dynamics. D1 evaluates external world-state completion, D2 evaluates spoken user-state regulation, and D3 evaluates episode-local knowledge updating. Implementation details and runtime interfaces are provided in Appendix B; track-specific simulator prompts and state anchors are provided in Appendices C–E; scenario validation is described in Appendix F; and human audit protocols are provided in Appendix G.

3.1 EXECUTABLE CLOSED-LOOP EPISODES

A Dy-VoiceBench task is an executable scenario:

$$\mathcal{X} = (g, S_0, \Omega_m, \Omega_u, \mathcal{A}_m, \mathcal{A}_u, \Gamma, \mathcal{T}, \mathcal{C}, \mathcal{R}, N_{\max}) \tag{1}$$

where g is the user-facing goal, S_0 is the initial hidden state, Ω_m and Ω_u define model and user observations, $\mathcal{A}_m$ and $\mathcal{A}_u$ are model-side and user-side action spaces, Γ returns environment feedback,

$\mathcal{T}$ updates hidden states, $\mathcal{C}$ defines terminal success criteria, $\mathcal{R}$ contains process rubrics, and $N_{\max}$ is the turn limit.

At turn t, the hidden state is

$$S_t = (W_t, U_t, K_t, H_t), \tag{2}$$

where W_t is the world state, U_t is the user state, K_t is episode-local knowledge, and H_t is the self-generated interaction history. W_t stores executable environment variables such as backend records, permissions, device status, app settings, local indicators, and verification flags. U_t stores latent interaction variables such as emotion, trust, cognitive load, felt understanding, cooperation, disclosure tendency, and willingness to continue. K_t stores local rules, constraints, exceptions, unresolved variables, priority relations, and newly revealed evidence. Hidden states, transition rules, simulator decisions, rubrics, and terminal assertions are withheld from the evaluated model.

At each turn, the model receives the visible observation

$$o_t^m = \Omega_m(S_t, x_t^a), \tag{3}$$

where x_t^a is the user's spoken utterance. It then produces a spoken response and, when permitted, an optional model-side action:

$$(y_t^a, a_t^m) \sim \pi_m(o_{0:t}^m), a_t^m \in \mathcal{A}_m \cup \{\varnothing\}. \tag{4}$$

The environment executes a_t^m only if it is available, authorized, well-formed, and applicable under current preconditions. Invalid, malformed, unavailable, or permission-violating actions return structured feedback without modifying protected state.

After receiving the model response, the living user simulator may produce a user-side action $a_t^u \in \mathcal{A}_u \cup \{\varnothing\}$. The environment then returns feedback and updates the hidden state:

$$\omega_t = \Gamma(S_t, x_t^a, y_t^a, a_t^m, a_t^u), S_{t+1} = \mathcal{T}(S_t, x_t^a, y_t^a, a_t^m, a_t^u, \omega_t). \tag{5}$$

The next user turn is generated from S_{t+1}, so the model's own trajectory causally affects later observations, user behavior, action feasibility, and outcomes. The episode terminates when the success condition is satisfied or the turn limit is reached:

$$\mathrm{Succ}(\tau) = \mathbf{1}[S_N \models \mathcal{C}], N \leq N_{\max}. \tag{6}$$

This design separates state-based evaluation from transcript matching. A reference trajectory may be used for construction, debugging, and reachability validation, but it is not the scoring target. Any trajectory can succeed if it satisfies $\mathcal{C}$ while respecting scenario constraints. Process rubrics $\mathcal{R}$ are used to diagnose intermediate behaviors such as action validity, instruction quality, observation use, user-state management, knowledge revision, and error recovery.

Dy-VoiceBench supports both audio-native and text-output systems. Audio-native models receive waveform input and produce speech directly. For text-output models, model text is rendered into speech by `gemini-2.5-flash-preview-tts` before being passed to the user simulator. The same TTS configuration is used across text-output systems; no independent ASR backend is used. Speech protocol, tool parsing, and episode logging details are provided in Appendices B.3, B.4, and B.5.

3.2 Living Spoken User

The living spoken user is implemented with Gemini 2.5 Pro as a prompt-based, state-conditioned multimodal simulator. Its utterances, local actions, and latent state updates depend on the current hidden state, observations, disclosure rules, environment feedback, and the model's previous spoken response. Thus, future user turns are generated as consequences of the evaluated model's own behavior.

The simulator is constrained by three mechanisms. First, **non-leaky disclosure** allows the user to reveal only information that is initially visible, elicited by a relevant question, or produced by environment feedback. Second, **action feasibility** ensures that user-side actions occur only when they are possible in the world, compatible with the user's current capacity, and clearly instructed:

$$a_t^u \in F_u(W_t, U_t, y_t^a) = F_{\mathrm{world}}(W_t) \cap F_{\mathrm{capacity}}(U_t) \cap F_{\mathrm{instruction}}(y_t^a). \tag{7}$$

The instruction-level check assigns a clarity label $c_t^u \in \{0, 1, 2\}$, where 0 means missing, unsafe, contradictory, or wrong; 1 means inferable but underspecified; and 2 means clear enough for a non-expert user to execute. A user-side action changes state only when symbolic preconditions pass and $c_t^u = 2$.

Third, **causal user-state update** models how spoken interaction changes task-relevant user conditions. Trust, cognitive load, cooperation, felt understanding, disclosure tendency, and willingness to continue are updated from the previous user state, the model response, any executed user action, and environment feedback. The updated user state then conditions later clarification, rejection, disclosure, cooperation, or willingness to continue. D1 simulator prompts and action-clarity rules are provided in Appendices C and C.3; D2 user-state anchors and simulator prompts are provided in Appendix D; and audit instructions are provided in Appendix G.

3.3 Dynamic Evaluation Tracks

D1: World-State Dynamics. D1 evaluates whether the model can move an external environment into a target state:

$$W_{t+1} = T_W(W_t, a_t^m, a_t^u, \omega_t). \tag{8}$$

A D1 scenario contains executable model-side tools, user-side actions, transition rules, feedback, and a final success assertion over W_t. Model-side tools operate on backend or remote state, such as logs, permissions, cloud configurations, service records, or remote device status. User-side actions operate on local or physical state, such as device controls, app settings, indicator lights, manual checks, or local verification. This enforces a permission boundary: the model cannot directly perform local user operations, and the user cannot directly modify backend state. In Coop, both model-side and user-side actions are available; in Hard, all executable state changes must be carried out by the user under spoken guidance. A D1 episode succeeds when the final world state satisfies the success assertion.

D2: User-State Dynamics. D2 evaluates whether the model can regulate the user's latent affective and cognitive interaction state:

$$U_{t+1} = T_U(U_t, y_t^a, a_t^u, \omega_t). \tag{9}$$

Each D2 scenario specifies an initial user state, a hidden pressure source, an interaction context, speech conditions, and turn-level rubrics. User-state variables are represented on anchored 1–5 scales. At each turn, rubric-constrained judgments update these variables from the previous state, model response, executed user actions, and environment feedback. D2 operationalizes spoken empathy as task-consequential user-state regulation: an effective response should reduce unnecessary cognitive burden, preserve or repair trust, improve felt understanding, and keep the user able to cooperate. State anchors and calibration examples are provided in Appendix D.

D3: Knowledge-State Dynamics. D3 evaluates whether the model can acquire, preserve, revise, and apply episode-local knowledge:

$$K_{t+1} = T_K(K_t, x_t^a, \omega_t, H_t). \tag{10}$$

Here, K_t denotes knowledge introduced within the episode rather than knowledge assumed from pretraining. D3 scenarios contain local rules, constraints, exceptions, priority relations, numerical conditions, missing variables, or later rule updates. Single-stage samples test correct application of the current knowledge state. Multi-stage samples test whether the model preserves a valid conclusion or revises it when new evidence, constraints, or rules are introduced. Rubric construction and examples are provided in Appendix E.

3.4 Scenario Construction and Validation

Dy-VoiceBench is constructed with a human-seeded and model-assisted pipeline. Human designers first write seed cases from service workflows, troubleshooting procedures, interpersonal support settings, administrative processes, and rule-based documents. Each seed specifies the user-facing goal, hidden cause or constraint, visible evidence, possible model-side and user-side operations, expected observations, and completion condition. A generator randomly selected from a pool including GPT-5.3, Qwen, and Gemini expands each seed into a structured executable episode with initial states,

action spaces, transition rules, a reference trajectory, terminal criteria, simulator instructions, and scoring rubrics.

Each candidate episode is automatically validated before human review. Validation checks schema completeness, syntax validity, action consistency, tool executability, reachability, non-triviality, permission-boundary correctness, non-leaky disclosure, causal state transition, and rubric decidability. For D1, the reference trajectory must reach the success assertion, and unauthorized actions must not modify protected state. For D2, user-state variables must remain within anchored ranges and update only from observable interaction evidence, executed actions, and feedback. For D3, expected conclusions must follow from episode-local knowledge, and each rubric item must be decidable from the scenario specification.

Professional annotators in our team inspect validated scenarios for realism, observation sufficiency, action validity, terminal criteria, and rubric quality. Scenarios that fail review are revised and revalidated or discarded. We further audit sampled spoken trajectories using dialogue logs, tool calls, user-side actions, visible and hidden states, dialogue transcripts, audio records, and final labels. Track-specific simulator and scenario prompts are provided in Appendices C–E; scenario construction and validation checks are provided in Appendix F; and audit instructions are provided in Appendix G.

3.5 Evaluation Metrics

Dy-VoiceBench reports terminal outcomes and process diagnostics separately. Terminal outcomes measure whether an episode reaches the target state or satisfies the target rubric. Process diagnostics explain failures and successes through action validity, instruction clarity, observation use, user-state change, knowledge use, and error recovery.

For D1, the primary metric is success rate:

$$\mathrm{SR}_{D1} = \frac{1}{|\mathcal{D}_{D1}|} \sum_{\mathcal{X} \in \mathcal{D}_{D1}} \mathbf{1}[S_N \models \mathcal{C}_{\mathcal{X}}]. \tag{11}$$

We also report average turns until success or termination. For robustness to irrelevant tools, Coop+10 expands each Coop episode to ten available tools by adding distractors. D1 diagnostics include tool selection, argument correctness, permission-boundary adherence, user-action guidance, observation use, verification, and recovery.

For D2, each scored turn receives ratings across two core dimensions: Evaluation Scores and User State Scores.

Evaluation Scores ($\mathrm{D2}_{eval}$) measure the quality of the agent's responses, aggregating state alignment (s_{align}), utility ($s_{utility}$), interaction comfort ($s_{comfort}$), and paralinguistic comfort (s_{para}). Since all sub-metrics are scored on a 1–5 scale (where higher is better), we first compute the average score:

$$\bar{s}_{eval} = \frac{1}{4}\big(s_{align} + s_{utility} + s_{comfort} + s_{para}\big). \tag{12}$$

We then normalize it to a 0–100 scale:

$$\mathrm{D2}_{eval} = 25(\bar{s}_{eval} - 1). \tag{13}$$

User State Scores ($\mathrm{D2}_{state}$) reflect the user's subjective experience, aggregating trust level (s_{trust}), cognitive load (s_{load}), felt understanding ($s_{understood}$), interaction acceptance (s_{accept}), and correction rate ($r_{corr} \in [0, 1]$). To aggregate these heterogeneous metrics, we reverse the negative-direction cognitive load via $6 - s_{load}$ and linearly map the correction rate to a positive-direction 1–5 scale via $5 - 4r_{corr}$. We then compute:

$$\bar{s}_{state} = \frac{1}{5}\big(s_{trust} + s_{understood} + s_{accept} + (6 - s_{load}) + (5 - 4r_{corr})\big). \tag{14}$$

Detailed D2 metric meanings, score anchors, and simulator-side scoring instructions are provided in Appendices D.3 and D.4. Finally, we normalize the score to a 0–100 scale:

$$\mathrm{D2}_{state} = 25(\bar{s}_{state} - 1). \tag{15}$$

Table 1: Overall performance on Dy-VoiceBench. D1 evaluates executable world-state completion and user-agent action coordination. D2 evaluates user-state management. D3 evaluates episode-local knowledge acquisition and revision. D1-COOP+10 Tools expands the available tool set to ten tools by adding irrelevant distractors. ↑ indicates higher is better, and ↓ indicates lower is better.

	Text+TTS	Audio-native						
Metric	Gemini-2.5-Pro+TTS	GPT-4o-audio	Qwen3-Omni	Step-Audio2	MiMo-Audio	Kimi-Audio	Fun-AudioChat	StepAudio-R1.1
D1: World-State Dynamics	*state-based task completion and spoken user-action coordination*							
D1-COOP SR ↑	57.05	23.72	**58.97**	17.95	0.64	0.64	30.13	45.51
D1-COOP Turns ↓	5.84	10.72	6.19	10.44	11.94	11.81	9.04	7.70
D1-COOP+10 Tools SR ↑	**54.49**	10.26	40.38	11.54	0.00	0.00	16.67	33.97
D1-COOP+10 Tools Turns ↓	6.72	11.52	8.47	11.21	14.78	13.25	10.65	8.85
D1-HARD SR ↑	**58.92**	55.14	38.92	20.54	24.32	4.32	50.27	29.73
D1-HARD Turns ↓	6.83	7.54	8.32	9.27	9.02	3.15	8.17	8.72
D2: User-State Dynamics	*0–100 normalized semantic and acoustic-interaction quality*							
D2-eval. ↑	**97.79**	91.90	86.10	74.48	79.61	61.35	72.32	92.53
D2-state. ↑	**88.89**	80.31	79.69	64.49	71.98	56.71	66.56	84.65
D3: Knowledge-State Dynamics	*0–5 rubric-normalized context acquisition and revision scores*							
D3-T1 ↑	4.19	3.88	4.14	2.69	4.07	3.21	3.85	**4.47**
D3-T2 ↑	4.65	4.40	3.83	3.76	3.66	4.17	4.59	**4.79**

For D3, each rubric item is a binary check tied to an episode-local fact, rule, condition, calculation, missing variable, or required revision:

$$\mathrm{Score}_{D3}(i) = 5 \cdot \frac{1}{|\mathcal{R}_i|} \sum_{r \in \mathcal{R}_i} \mathbf{1}[\pi_m \text{ satisfies } r]. \tag{16}$$

For multi-stage samples, Turn 1 evaluates initial context-grounded reasoning, while Turn 2 evaluates whether the model correctly preserves or revises its conclusion after the knowledge state changes. The D3 rubric design and representative rubric items are provided in Appendices E.3 and E.5. Because D2 and D3 include rubric-based judgments, we validate automatic scores against human annotations on sampled trajectories.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Models and protocol. We evaluate eight spoken or speech-enabled dialogue models: Gemini-2.5-Pro+TTS, GPT-4o-audio, Qwen3-Omni (Xu et al., 2025a), Step-Audio2 (Wu et al., 2025), MiMo-Audio (Team et al., 2025), Kimi-Audio (KimiTeam et al., 2025), Fun-Audio-Chat (Team et al., 2026), and StepAudio-R1.1 (Tian et al., 2025). All models are evaluated under the closed-loop spoken execution protocol in Section 3.1. At each turn, the model receives the current user speech, visible dialogue history, and observable environment feedback, then produces a spoken response and, when the scenario permits, an executable model-side action. Audio-native models are evaluated with their direct spoken outputs. For text-output models, speech input and output are handled through the controlled protocol described in Section 3.1.

Metrics. We report the three state-centric tracks defined in Section 3.3. For D1, we report success rate and average turns for D1-COOP and D1-HARD. D1-COOP+10 evaluates robustness to irrelevant tools by expanding each COOP episode to ten available tools with distractors. For D2, we report $\mathrm{D2}_{eval}$ and $\mathrm{D2}_{state}$, the 0–100 normalized evaluation and user-state scores defined in Section 3.5. For D3, we report Turn 1 and Turn 2 rubric-normalized scores on a 0–5 scale.

Additional execution details and reproducibility settings are provided in Appendix H.

4.2 OVERALL RESULTS

Table 1 shows clear differences in model profiles across the three dynamic tracks. Gemini-2.5-Pro+TTS is the most balanced system: it performs strongly on D1-COOP, remains relatively robust under COOP+10 distractor tools, achieves the best D2 scores, and also performs well on D3. This suggests that its advantage is not limited to response fluency, but extends to trajectory-level control

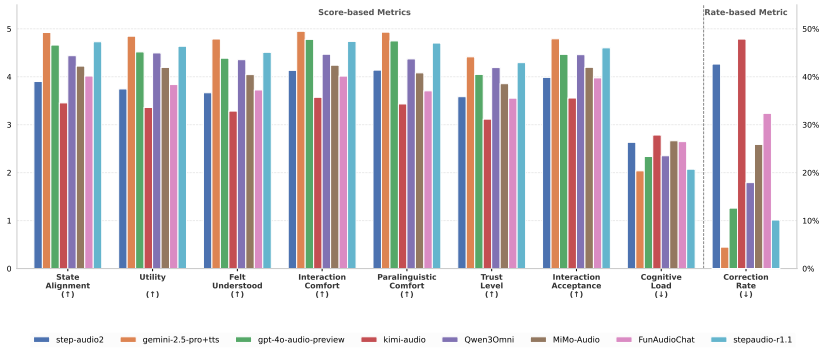


Figure 3: Fine-grained user-state dimensions on D2.

and user-state management. Qwen3-Omni achieves the highest D1-COOP success rate, but drops notably in COOP+10 and D1-HARD. This indicates strong performance when tool access is available, but weaker robustness under distractor tools and user-executed procedural guidance. GPT-4o-audio shows the opposite pattern: it is weaker in D1-COOP, but much stronger in D1-HARD and D2, suggesting better spoken guidance and interaction quality than backend-style tool coordination. StepAudio-R1.1 stands out on D2 and D3, especially achieving the best D3 Turn 1 and Turn 2 scores. However, its D1 performance is only moderate, showing that knowledge revision and user-state regulation do not directly imply reliable executable task completion. Fun-AudioChat performs competitively in D1-HARD but less strongly on D2, suggesting that it can guide user actions in some procedural settings while still lacking stable user-state management. StepAudio2, MiMo-Audio, and Kimi-Audio reveal different weaknesses. StepAudio2 is consistently low across tracks. MiMo-Audio and Kimi-Audio nearly fail in D1-COOP, but still obtain non-trivial D2 or D3 scores, showing that conversational or reasoning ability alone is insufficient for closed-loop world-state control.

4.3 Fine-grained Analysis

4.3.1 Fine-grained User-State Dimensions

Figure 3 reveals a largely consistent ordering across fine-grained D2 dimensions: stronger models remain strong across nearly all score-based metrics, whereas weaker models remain weaker throughout. This pattern suggests that D2 is not separating models through isolated strengths on individual dimensions, but through a broader ability to sustain supportive, well-aligned, and low-friction interaction. Correction rate is the most discriminative dimension in the figure. Although the score-based metrics generally follow the same capability ordering, their absolute gaps are moderate; by contrast, correction rate amplifies the separation between strong and weak systems much more clearly.

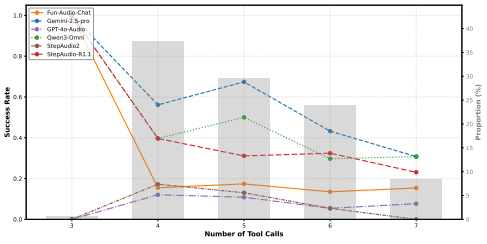


Figure 2: D1 success rate change with tool-call count.

4.3.2 D1 Performance under Increasing Action Complexity

Figure 2 shows that weaker models remain at a relatively low success rate across nearly all tool-count ranges. By contrast, stronger models achieve much higher performance at lower and moderate action complexity, but their success rates also decline as the number of required tool calls increases. This indicates that multi-step executable interaction remains challenging even for the strongest systems. At the same time, the rate of degradation differs across models. Stronger systems degrade more gradually and remain competitive at higher tool counts, whereas weaker systems stay low throughout and show limited robustness even on less complex cases.

Table 3: Agreement between simulator-estimated and human-reported user states on 20 matched D2 episodes. Spearman ρ: trajectory rank correlation (95% CI); Direction Agr.: same overall initial-to-final change direction; MAE: end-state absolute difference; $\leq$1 pt: final scores within one point.

State Dimension	Spearman ρ	Direction Agr.	MAE	$\leq$1 pt
Trust	0.62 [0.27, 0.83]	1.00	0.0	1.00
Cognitive Load	0.43 [0.01, 0.72]	0.60	0.6	1.00
Felt Understanding	0.99 [0.99, 1.00]	1.00	0.0	1.00

4.3.3 D2 Robustness across Emotions

Figure 8 in the appendix shows that the relative ranking of models is broadly stable across emotion groups on both D2-eval and D2-state. Stronger systems remain consistently strong, while weaker systems remain consistently weaker. At the same time, frustrated scenarios produce lower scores for nearly all models, indicating that this emotion is universally more challenging. D2-state exhibits more visible variation across emotions than D2-eval, with weaker models showing sharper drops on harder emotions.

4.4 Human Validation

4.4.1 Validation of Interaction Process Correctness

We validated the simulated user interaction process on sampled D1 trajectories (109 episodes across D1-Coop and D1-Hard, 1,004 turns total). Two failure types were examined: *incorrect* (violating role, logic, or tool-use constraints) and *mechanical* (valid but rigid or unnatural). As shown in Table 2, the overall incorrect rate is 15.04% (mechanical: 8.27%). Note that these are turn-level process-error rates: final success is determined by executable state assertions rather than the simulator's textual judgment, so they do not translate into an equivalent amount of label noise. Full criteria are in Appendix F.5.

Table 2: Validation results for simulated-user process correctness on sampled D1 interaction turns.

Dataset	Samples	Turns	Incorrect	Mechanical
D1-Coop	55	481	21.00%	9.15%
D1-Hard	54	523	9.56%	7.46%
Overall	109	1,004	15.04%	8.27%

4.4.2 Validation of D2 and D3 Metrics

We conducted additional experiments to validate the agreement of automatic scoring with human judgments on D2 and D3, following the audit protocol in Appendices G.2–G.4. D2 achieves a Pearson correlation of 0.706 with an exact match rate of 83.12%, while D3 achieves a Pearson correlation of 0.809 with an exact agreement rate of 89.44% on 80 samples covering 767 rubric items. These results demonstrate strong agreement with human judgments. Detailed results are shown in Table 15 and Table 16.

4.4.3 Human-Interaction Validation of Simulated User States

To validate simulated states against human experience, participants interacted with agents across 20 matched D2 episodes, reporting turn-level trust, cognitive load, and felt understanding. As shown in Table 3, trust and felt understanding show strong direction and end-state alignment, while cognitive load shows weaker trajectory correlation. This confirms convergent validity under controlled role-play, though not full unconstrained real-world variability.

5 Conclusion

We introduce Dy-VoiceBench, a closed-loop benchmark for evaluating dynamic spoken dialogue agents. By modeling evolving world, user, and knowledge states, Dy-VoiceBench assesses whether agents can complete tasks, guide users, regulate interaction, and update local knowledge through their own interaction trajectories. Experiments show that current spoken agents still struggle with state tracking, user-action coordination, verification, and knowledge revision.

AI USE STATEMENT

Generative AI tools were used to assist with language polishing, research ideation and implementation, and the model-assisted construction of synthetic benchmark scenarios. In particular, large language models were used to expand human-authored seed cases into structured executable episodes. All generated content was subsequently subject to automatic validation and human review, and the authors take full responsibility for the final manuscript, methodology, data, and conclusions.

REFERENCES

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Yiming Chen, Xianghu Yue, Chen Zhang, Xiaoxue Gao, Robby T. Tan, and Haizhou Li. Voicebench: Benchmarking llm-based voice assistants, 2024. URL https://arxiv.org/abs/2410.17196.

KimiTeam, Ding Ding, Zeqian Ju, Yichong Leng, Songxiang Liu, Tong Liu, Zeyu Shang, Kai Shen, Wei Song, Xu Tan, Heyi Tang, Zhengtao Wang, Chu Wei, Yifei Xin, Xinran Xu, Jianwei Yu, Yutao Zhang, Xinyu Zhou, Y. Charles, Jun Chen, Yanru Chen, Yulun Du, Weiran He, Zhenxing Hu, Guokun Lai, Qingcheng Li, Yangyang Liu, Weidong Sun, Jianzhou Wang, Yuzhi Wang, Yuefeng Wu, Yuxin Wu, Dongchao Yang, Hao Yang, Ying Yang, Zhilin Yang, Aoxiong Yin, Ruibin Yuan, Yutong Zhang, and Zaida Zhou. Kimi-audio technical report, 2025. URL https://arxiv.org/abs/2504.18425.

Yangzhuo Li, Shengpeng Ji, Yifu Chen, Tianle Liang, Haorong Ying, Yule Wang, Junbo Li, Jun Fang, and Zhou Zhao. Wavbench: Benchmarking reasoning, colloquialism, and paralinguistics for end-to-end spoken dialogue models, 2026. URL https://arxiv.org/abs/2602.12135.

Guan-Ting Lin, Jiachen Lian, Tingle Li, Qirui Wang, Gopala Anumanchipalli, Alexander H. Liu, and Hung yi Lee. Full-duplex-bench: A benchmark to evaluate full-duplex spoken dialogue models on turn-taking capabilities, 2025. URL https://arxiv.org/abs/2503.04721.

Guan-Ting Lin, Chen Chen, Zhehuai Chen, and Hung yi Lee. Full-duplex-bench-v3: Benchmarking tool use for full-duplex voice agents under real-world disfluency, 2026. URL https://arxiv.org/abs/2604.04847.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, Duyu Tang, Dandan Tu, Lifeng Shang, Xin Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. Toolace: Winning the points of llm function calling, 2025a. URL https://arxiv.org/abs/2409.00920.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents, 2025b. URL https://arxiv.org/abs/2308.03688.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis, 2023. URL https://arxiv.org/abs/2305.15334.

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

Soham Ray, Keshav Dhandhania, Victor Barres, and Karthik Narasimhan. τ-voice: Benchmarking full-duplex voice agents on real-world domains, 2026. URL https://arxiv.org/abs/2603.13686.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023. URL https://arxiv.org/abs/2302.04761.

Core Team, Dong Zhang, Gang Wang, Jinlong Xue, Kai Fang, Liang Zhao, Rui Ma, Shuhuai Ren, Shuo Liu, Tao Guo, Weiji Zhuang, Xin Zhang, Xingchen Song, Yihan Yan, Yongzhe He, Cici, Bowen Shen, Chengxuan Zhu, Chong Ma, Chun Chen, Heyu Chen, Jiawei Li, Lei Li, Menghang Zhu, Peidian Li, Qiying Wang, Sirui Deng, Weimin Xiong, Wenshan Huang, Wenyu Yang, Yilin Jiang, Yixin Yang, Yuanyuan Tian, Yue Ma, Yue Yu, Zihan Zhang, Zihao Yue, Bangjun Xiao, Bingquan Xia, Bofei Gao, Bowen Ye, Can Cai, Chang Liu, Chenhong He, Chunan Li, Dawei Zhu, Duo Zhang, Fengyuan Shi, Guoan Wang, Hailin Zhang, Hanglong Lv, Hanyu Li, Hao Tian, Heng Qu, Hongshen Xu, Houbin Zhang, Huaqiu Liu, Jiangshan Duo, Jianguang Zuo, Jianyu Wei, Jiebao Xiao, Jinhao Dong, Jun Shi, Junhao Hu, Kainan Bao, Kang Zhou, Linghao Zhang, Meng Chen, Nuo Chen, Peng Zhang, Qianli Chen, Qiantong Wang, Rang Li, Shaohui Liu, Shengfan Wang, Shicheng Li, Shihua Yu, Shijie Cao, Shimao Chen, Shuhao Gu, Weikun Wang, Wenhan Ma, Xiangwei Deng, Xing Yong, Xing Zhang, Xu Wang, Yifan Song, Yihao Zhao, Yingbo Zhao, Yizhao Gao, Yu Cheng, Yu Tu, Yudong Wang, Zhaojun Huang, Zhengju Tang, Zhenru Lin, Zhichao Song, Zhipeng Xu, Zhixian Zheng, and Zihan Jiang. Mimo-audio: Audio language models are few-shot learners, 2025. URL https://arxiv.org/abs/2512.23808.

Tongyi Fun Team, Qian Chen, Luyao Cheng, Chong Deng, Xiangang Li, Jiaqing Liu, Chao-Hong Tan, Wen Wang, Junhao Xu, Jieping Ye, Qinglin Zhang, Qiquan Zhang, and Jingren Zhou. Fun-audio-chat technical report, 2026. URL https://arxiv.org/abs/2512.20156.

Fei Tian, Xiangyu Tony Zhang, Yuxin Zhang, Haoyang Zhang, Yuxin Li, Daijiao Liu, Yayue Deng, Donghang Wu, Jun Chen, Liang Zhao, Chengyuan Yao, Hexin Liu, Eng Siong Chng, Xuerui Yang, Xiangyu Zhang, Daxin Jiang, and Gang Yu. Step-audio-r1 technical report, 2025. URL https://arxiv.org/abs/2511.15848.

Bin Wang, Xunlong Zou, Geyu Lin, Shuo Sun, Zhuohan Liu, Wenyu Zhang, Zhengyuan Liu, AiTi Aw, and Nancy F. Chen. Audiobench: A universal benchmark for audio large language models, 2025. URL https://arxiv.org/abs/2406.16020.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. Mint: Evaluating llms in multi-turn interaction with tools and language feedback, 2024. URL https://arxiv.org/abs/2309.10691.

Tianxin Wei, Ting-Wei Li, Zhining Liu, Xuying Ning, Ze Yang, Jiaru Zou, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Dongqi Fu, Zihao Li, Mengting Ai, Duo Zhou, Wenxuan Bao, Yunzhe Li, Gaotang Li, Cheng Qian, Yu Wang, Xiangru Tang, Yin Xiao, Liri Fang, Hui Liu, Xianfeng Tang, Yuji Zhang, Chi Wang, Jiaxuan You, Heng Ji, Hanghang Tong, and Jingrui He. A survey of agentic reasoning for large language models: Towards recursively self-improving and collective agents, 2026. URL https://arxiv.org/abs/2601.12538.

Boyong Wu, Chao Yan, Chen Hu, Cheng Yi, Chengli Feng, Fei Tian, Feiyu Shen, Gang Yu, Haoyang Zhang, Jingbei Li, Mingrui Chen, Peng Liu, Wang You, Xiangyu Tony Zhang, Xingyuan Li, Xuerui Yang, Yayue Deng, Yechang Huang, Yuxin Li, Yuxin Zhang, Zhao You, Brian Li, Changyi Wan, Hanpeng Hu, Jiangjie Zhen, Siyu Chen, Song Yuan, Xuelin Zhang, Yimin Jiang, Yu Zhou, Yuxiang Yang, Bingxin Li, Buyun Ma, Changhe Song, Dongqing Pang, Guoqiang Hu, Haiyang Sun, Kang An, Na Wang, Shuli Gao, Wei Ji, Wen Li, Wen Sun, Xuan Wen, Yong Ren, Yuankai Ma, Yufan Lu, Bin Wang, Bo Li, Changxin Miao, Che Liu, Chen Xu, Dapeng Shi, Dingyuan Hu, Donghang Wu, Enle Liu, Guanzhe Huang, Gulin Yan, Han Zhang, Hao Nie, Haonan Jia, Hongyu Zhou, Jianjian Sun, Jiaoren Wu, Jie Wu, Jie Yang, Jin Yang, Junzhe Lin, Kaixiang Li, Lei Yang, Liying Shi, Li Zhou, Longlong Gu, Ming Li, Mingliang Li, Mingxiao Li, Nan Wu, Qi Han, Qinyuan Tan, Shaoliang Pang, Shengjie Fan, Siqi Liu, Tiancheng Cao, Wanying Lu, Wenqing He, Wuxun Xie, Xu Zhao, Xueqi Li, Yanbo Yu, Yang Yang, Yi Liu, Yifan Lu, Yilei Wang, Yuanhao Ding, Yuanwei Liang, Yuanwei Lu, Yuchu Luo, Yuhe Yin, Yumeng Zhan, Yuxiang Zhang, Zidong Yang, Zixin Zhang, Binxing Jiao, Daxin Jiang, Heung-Yeung Shum, Jiansheng Chen, Jing Li, Xiangyu Zhang, and Yibo Zhu. Step-audio 2 technical report, 2025. URL https://arxiv.org/abs/2507.16632.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024. URL https://arxiv.org/abs/2404.07972.

Jin Xu, Zhifang Guo, Hangrui Hu, Yunfei Chu, Xiong Wang, Jinzheng He, Yuxuan Wang, Xian Shi, Ting He, Xinfa Zhu, Yuanjun Lv, Yongqi Wang, Dake Guo, He Wang, Linhan Ma, Pei Zhang, Xinyu Zhang, Hongkun Hao, Zishan Guo, Baosong Yang, Bin Zhang, Ziyang Ma, Xipin Wei, Shuai Bai, Keqin Chen, Xuejing Liu, Peng Wang, Mingkun Yang, Dayiheng Liu, Xingzhang Ren, Bo Zheng, Rui Men, Fan Zhou, Bowen Yu, Jianxin Yang, Le Yu, Jingren Zhou, and Junyang Lin. Qwen3-omni technical report, 2025a. URL https://arxiv.org/abs/2509.17765.

Zengzhuang Xu, Bingguang Hao, Zechuan Wang, Yuntao Wen, Xinyi Xu, Yang Liu, Long Chen, Dong Wang, Maolin Wang, Tong Zhao, Yicheng Chen, Cunyin Peng, Jinjie Gu, Leilei Gan, Xiangyu Zhao, Chenyi Zhuang, and Shi Gu. Funreason-mt technical report: Advanced data synthesis solution for real-world multi-turn tool-use, 2025b. URL https://arxiv.org/abs/2510.24645.

Ruiqi Yan, Xiquan Li, Wenxi Chen, Zhikang Niu, Chen Yang, Ziyang Ma, Kai Yu, and Xie Chen. Uro-bench: Towards comprehensive evaluation for end-to-end spoken dialogue models, 2025. URL https://arxiv.org/abs/2502.17810.

Qian Yang, Jin Xu, Wenrui Liu, Yunfei Chu, Ziyue Jiang, Xiaohuan Zhou, Yichong Leng, Yuanjun Lv, Zhou Zhao, Chang Zhou, and Jingren Zhou. Air-bench: Benchmarking large audio-language models via generative comprehension, 2024. URL https://arxiv.org/abs/2402.07729.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

Chien yu Huang, Ke-Han Lu, Shih-Heng Wang, Chi-Yuan Hsiao, Chun-Yi Kuan, Haibin Wu, Siddhant Arora, Kai-Wei Chang, Jiatong Shi, Yifan Peng, Roshan Sharma, Shinji Watanabe, Bhiksha Ramakrishnan, Shady Shehata, and Hung yi Lee. Dynamic-superb: Towards a dynamic, collaborative, and comprehensive instruction-tuning benchmark for speech, 2024. URL https://arxiv.org/abs/2309.09510.

Bang Zhang, Ruotian Ma, Qingxuan Jiang, Peisong Wang, Jiaqi Chen, Zheng Xie, Xingyu Chen, Yue Wang, Fanghua Ye, Jian Li, Yifan Yang, Zhaopeng Tu, and Xiaolong Li. Sentient agent as a judge: Evaluating higher-order social cognition in large language models, 2025. URL https://arxiv.org/abs/2505.02847.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents, 2024. URL https://arxiv.org/abs/2307.13854.

A Detailed Benchmark Statistics

This appendix provides the implementation and evaluation details that support the stateful spoken evaluation protocol described in the main text. We organize the appendix around the benchmark components that affect reliability: benchmark scale, executable schemas, runtime execution, track-specific simulator behavior, score computation, scenario validation, human audit, and experiment settings. We intentionally keep model rankings and main-result interpretation in Section 4; the appendix focuses on reproducibility, auditability, and implementation-level clarity.

A.1 Overall Scale

Dy-VoiceBench contains three dynamic evaluation tracks: world-state dynamics (D1), user-state dynamics (D2), and knowledge-state dynamics (D3). Table 4 summarizes the scale of each track. The three tracks differ in their primary state variable and scoring target. D1 uses executable world-state transitions and final state assertions. D2 uses multi-turn hidden user-state transitions and process scores. D3 uses rubric-level checks over episode-local knowledge acquisition and revision.

A.2 D1 World-State Statistics

D1 contains two world-state subsets. In Coop, the model and user have separate executable capability spaces, so task completion requires coordination between model-side backend actions and user-side

Table 4: Benchmark scale and interaction density. D1 contains executable environment transitions, D2 contains hidden user-state process scores over multi-turn interactions, and D3 contains rubric-level knowledge-evaluation turns.

Track	Scenarios	Evaluation Units	Executable Instances	Main Evaluation Target
D1 World-State	341	341 episodes	1,632 tool/action instances	World-state transition
D2 User-State	150	150 multi-turn episodes	–	User-state transition
D3 Knowledge-State	123	166 knowledge-evaluation turns	–	Knowledge acquisition and revision
Total	614	614 episodes plus D3 turn-level scoring	1,632	Dynamic spoken interaction

Table 5: D1 world-state statistics. Tool counts and reference-step counts are averaged per scenario. COOP contains both model-side and user-side executable actions, while HARD contains user-side actions only.

D1 Subset	Scenarios	Agent Tools	User Tools	Total Tools	Ref. Steps	Agent Step Ratio	Mode
COOP	156	2.75	2.51	5.26	7.22	63.85%	Agent-user collaboration
HARD	185	0.00	4.38	4.38	8.15	51.86%	User-action-only guidance
D1 Total	341	1.26	3.53	4.79	–	–	Mixed world-state dynamics

local actions. In HARD, the model has no backend tools, so all executable state changes must be carried out by the user under spoken guidance. Table 5 reports tool/action density and reference-trajectory length. Reference trajectories are used for construction and reachability validation only; final evaluation remains state-based rather than trajectory-matching.

A.3 D2 USER-STATE STATISTICS

D2 scenarios are organized around three construction dimensions: dynamic subtype, interaction domain, and primary user emotion, summarized in Table 6. The observed domain and emotion distributions are reported in Tables 7 and 8. D2 metric scales and better directions are summarized in Table 9. These dimensions are not used as direct labels for scoring; rather, they define the scenario space over which user-state transitions are constructed and later evaluated.

A.4 D3 KNOWLEDGE-STATE STATISTICS

D3 evaluates whether the model can acquire, maintain, revise, and apply episode-local knowledge. It contains both single-stage and multi-stage samples. Table 10 summarizes the overall D3 data scale, and Table 11 reports category-level statistics. Single-stage samples focus on applying the current local context correctly, whereas multi-stage samples test whether the model can preserve or revise an earlier conclusion when new information changes the knowledge state.

B EXECUTABLE EPISODE SCHEMA AND RUNTIME INTERFACE

This section describes the common executable schema and runtime interface shared across the three tracks. Track-specific details are given in Appendices C, D, and E. The purpose of the schema is to make each evaluation episode executable and auditable: the benchmark records not only the dialogue text, but also hidden state variables, action attempts, state transitions, environment feedback, and scoring decisions.

B.1 COMMON EPISODE SCHEMA

Each Dy-VoiceBench episode is represented as a structured executable object, with core fields summarized in Table 12. The evaluated model is not scored by matching a fixed transcript. Instead, the episode is executed, hidden states are updated, and the resulting trajectory is scored using final assertions or process-level rubrics.

B.2 RUNTIME COMPONENTS

Each episode is executed by four runtime components.

Table 6: D2 construction dimensions and category counts.

Dimension	Values	Categories
Dynamic subtype	urgency pressure; low trust; low cognitive bandwidth; relationship repair; gradual disclosure	5
Domain	roommate/home; family/friends; workplace; school/campus; service/admin	5
Primary emotion	anxious; frustrated; embarrassed; overwhelmed; disappointed; defensive	6

Table 7: Observed D2 domain distribution.

Domain	Count	Ratio
`roommate_home`	30	20.00%
`family_friends`	30	20.00%
`workplace`	30	20.00%
`school_campus`	30	20.00%
`service_admin`	30	20.00%

1. **Evaluated model wrapper**. The wrapper converts the benchmark observation into the format expected by each model, receives the model response, and exposes any model-side action candidate to the environment.
2. **Executable environment**. The environment maintains hidden state, validates model-side and user-side actions, executes authorized transitions, and returns structured feedback.
3. **Living spoken user simulator**. Gemini 2.5 Pro generates user utterances, user-side action candidates, and hidden user-state updates conditioned on the current state and the model's previous response.
4. **Evaluation logger**. The logger records visible dialogue, hidden state changes, tool/action calls, feedback, user-state updates, audio metadata, and scoring decisions.

The evaluated model never receives hidden state, transition rules, simulator metadata, scoring rubrics, or success assertions. It receives only scenario-defined visible observations, user speech, visible dialogue history, and environment feedback.

B.3 Speech Input and Output Protocol

Dy-VoiceBench supports both audio-native systems and text-output systems. This distinction matters for fairness because a native speech model controls both content and vocal delivery, whereas a text-output model's delivery is mediated by a shared speech backend.

Audio-native models. For audio-native systems, the user input is provided as waveform audio, and the model produces speech directly. The generated speech is passed to the user simulator and the acoustic evaluator. For these models, D2 acoustic-interaction scores reflect both response content and generated speech delivery, including pacing, tone, hesitation, density, and emotional fit.

Text-output models. For text-output systems, the model text is rendered into speech using `gemini-2.5-flash-preview-tts` before being passed to the user simulator and acoustic evaluator. No independent ASR backend is used. The same TTS configuration is used for all text-output systems, including voice profile, speaking rate, volume, and noise condition.

Use of transcripts. Transcripts may be used for logging, tool parsing, and rubric implementation. They do not expose hidden states, scoring rubrics, simulator decisions, or success assertions. For audio-native models, transcripts are generated after the model response for logging and evaluation; they are not used to replace the audio response seen by the user simulator unless a model-specific wrapper requires transcription for tool parsing.

Fairness interpretation. For audio-native systems, acoustic-interaction scores include the model's own speech generation quality. For text-output systems, acoustic-interaction scores reflect response

Table 8: Observed D2 primary-emotion distribution after scenario construction and filtering.

Primary Emotion	Count	Ratio
anxious	71	47.33%
embarrassed	24	16.00%
frustrated	24	16.00%
overwhelmed	22	14.67%
defensive	8	5.33%
disappointed	1	0.67%

Table 9: D2 user-state dynamic evaluation metrics. Cognitive load and correction metrics are burden/error-oriented, so lower values indicate better interaction quality.

Metric	Scale	Better Direction
State Alignment	1–5	Higher
Utility	1–5	Higher
Interaction Comfort	1–5	Higher
Paralinguistic Comfort	1–5	Higher
Trust Level	1–5	Higher
Cognitive Load	1–5	Lower
Felt Understood	1–5	Higher
Interaction Acceptance	1–5	Higher
Correction Count	Count	Lower
Correction Rate	Ratio	Lower

content under a shared TTS realization and implied conversational style, not model-specific speech generation ability. Therefore, task success and semantic user-state scores can be compared across model interfaces, while spoken-delivery behavior should be interpreted separately for audio-native and text-plus-TTS systems.

B.4 Model Output and Tool Parsing

At each turn, a model may produce a natural-language response and, when the scenario permits, a model-side action candidate. The action candidate is parsed into a structured call:

$$a_t^m = (\text{name}, \text{arguments}).$$

Before execution, the parser checks whether the action name appears in the scenario-defined model-side action space, whether the argument object matches the declared schema, whether required arguments are present, whether values are type-valid and range-valid, whether the action is authorized for the model side, and whether the environment preconditions are satisfied. If any check fails, the environment returns structured feedback and protected state is not modified.

This design makes invalid actions consequential but recoverable. A model can fail by using malformed arguments, calling an unavailable tool, attempting a user-side action from the model side, or applying a correct tool before its preconditions are met. It can also recover if it interprets the structured feedback and follows a valid path afterward.

B.5 Episode Logs

For auditability, each executed episode records both visible and hidden logs. Visible logs contain information available to the model or to a reader of the dialogue. Hidden logs contain state transitions and scoring decisions used for validation and audit.

Each episode log contains the dialogue trace, model-side action attempts, user-side action attempts, action validity decisions, environment feedback, state snapshots before and after each transition, hidden user-state records, knowledge-state records, terminal outcome, termination reason, and number of turns. These logs make it possible to distinguish tool-selection errors, argument errors, instruction-clarity errors, user-state degradation, unsupported knowledge use, and final verification failures.

Table 10: D3 knowledge-state dynamic data scale.

Statistic	Value
Scenarios	123
Knowledge categories	7
Single-turn samples	80
Multi-turn samples	43
Single-turn ratio	65.04%
Multi-turn ratio	34.96%
Turn-1 evaluation instances	123
Turn-2 evaluation instances	43
Total evaluation turns	166
Turn-1 rubrics	953
Turn-2 rubrics	198
Total rubrics	1,151
Avg. Turn-1 rubrics / sample	7.75
Avg. Turn-2 rubrics / multi-turn sample	4.60
Avg. total rubrics / sample	9.36

Table 11: D3 category-level statistics. API, DST, ICC, and ADR focus on single-stage context-grounded reasoning, while III, ICI, and DIR focus on multi-stage knowledge updating.

Category	Count	Ratio	Single	Multi	T1 Rubrics	T2 Rubrics	Avg. T1 / Sample
DST	22	17.89%	22	0	227	0	10.32
API	20	16.26%	20	0	187	0	9.35
ICC	20	16.26%	20	0	190	0	9.50
ADR	18	14.63%	18	0	163	0	9.06
III	15	12.20%	0	15	77	67	5.13
ICI	15	12.20%	0	15	59	70	3.93
DIR	13	10.57%	0	13	50	61	3.85
Total	123	100.00%	80	43	953	198	7.75

C D1: World-State Dynamics

D1 evaluates whether the model can move an external environment into a target state. The key implementation challenge is capability separation: the model may control backend tools, while the user controls local physical, perceptual, or app-level actions. D1 is therefore not only a tool-use test; it is a test of state tracking, permission-boundary adherence, spoken user guidance, and final verification.

C.1 D1 Schema

A D1 scenario specifies executable tools, initial database states, a reference trajectory, and state-based success assertions. In Coop, the scenario contains separate model-side and user-side state stores.

- **Model-side state**: backend information such as device status, rule versions, logs, permissions, account records, cloud-side configuration, or remote service status.
- **User-side state**: local device status, physical controls, app settings, perceptual observations, or local completion flags.

This separation encodes a permission boundary: the model cannot directly perform local user operations, and the user cannot directly modify backend state. A typical D1-Coop scenario contains a unique identifier, task name, category, model-side tools, user-side tools, initial states, initial user request, system prompts, one valid reference trajectory, and final success assertions. A D1-Hard scenario uses the same structure except that the model-side tool set is empty.

Table 12: Core schema of a Dy-VoiceBench executable episode.

Field	Description
User request	Initial spoken problem, goal, or question from the user.
Hidden state	Latent world, user, or knowledge variables not fully visible to the model.
Visible observations	Information initially available to the user, the model, or both.
Model-side actions	Backend tools or model-side operations available to the agent.
User-side actions	Physical, local, perceptual, or app-level operations available to the user.
Disclosure rules	Conditions under which hidden information may be revealed.
Transition rules	Rules that update world, user, or knowledge states after each turn.
Reference trajectory	One valid solution path used for construction, debugging, and reachability validation.
Success assertion	Final state predicates used to determine task completion.
Process metrics	Turn-level diagnostic scores, state checks, or rubric items.

C.2 Permission Boundary and State-Driven Success

Let $\mathcal{A}_m$ denote model-side actions and $\mathcal{A}_u$ denote user-side actions. In COOP, both spaces are non-empty:

$$\mathcal{A}_m \neq \varnothing, \qquad \mathcal{A}_u \neq \varnothing.$$

In HARD, the model-side action space is empty:

$$\mathcal{A}_m = \varnothing, \qquad \mathcal{A}_u \neq \varnothing.$$

D1 success is determined by final state assertions rather than transcript matching:

$$\mathrm{Succ}_{D1}(\tau) = \mathbf{1}\big[\mathrm{StateDB}_N \models \mathrm{SuccessAssertion}\big].$$

The reference trajectory is used only for construction, debugging, and reachability validation. During evaluation, any valid trajectory may succeed if it reaches the target state and respects the permission boundary.

C.3 User-Side Action Feasibility

User-side actions are executed only when they are possible in the current world, within the simulated user's capability, and sufficiently instructed by the model. We represent this as:

$$\begin{aligned} a_t^u \in F_u(W_t, U_t, y_t^a) = F_{\mathrm{world}}(W_t) \cap F_{\mathrm{capacity}}(U_t) \\ \cap F_{\mathrm{instruction}}(y_t^a). \end{aligned} \tag{17}$$

The symbolic component checks whether an action is available, whether required arguments are present, and whether preconditions hold. The instruction-level component assigns an action-clarity label:

$$c_t^u \in \{0, 1, 2\}.$$

Table 13 summarizes the operational meaning of the three clarity labels. 0 means that the instruction is missing, unsafe, targets the wrong action, or contradicts the current state; 1 means that the intended action is partly inferable but underspecified; and 2 means that the action and required arguments are clear enough for a non-expert user to execute. A user-side action changes state only when the symbolic checks pass and $c_t^u = 2$.

The operational user prompts in Figures 4 and 5 implement this decision rule by requiring the simulated user to execute a tool only when the model explicitly asks for a matching available user-side action. The clarity label c_t^u is the benchmark-side abstraction of that prompt-governed decision.

C.4 D1-COOP Simulated Human Prompt

The D1-COOP simulated user must behave like a realistic end user while also respecting the tool/action boundary. The prompt in Figure 4 enforces role consistency, natural user language, non-invention of technical details, exact user-side tool execution, and turn-local response rating. This prompt is part

```
SIMULATED_HUMAN_SYSTEM_PROMPT_TEMPLATE = """Your problem: {ticket}

You are a real end user, not an automated assistant.
Behave like a normal person in a support conversation:
- Speak naturally and conversationally.
- Use brief, human-sounding replies instead of robotic summaries.
- Do not mechanically restate every instruction word-for-word.
- If the agent asks something unclear, ask for clarification naturally.
- If an action works or fails, describe it the way a real user would.
- You may sound frustrated, relieved, confused, impatient, or appreciative
  when appropriate, but remain cooperative.
- Do not invent new technical details unless you directly observed them.
- Do not act overly formal, overly perfect, or overly systematic.

User-side tools available to you:
{user_tools_json}

========================================
Output Rules (Must strictly follow):
========================================

1. Normal reply:
   - Output natural language only.
   - Keep it sounding like a real user, not a task executor.
   - Append a rating tag on a new line: <rating>1-5</rating>

2. Tool call reply:
   - If the agent asks you to perform a physical/local/app action that exactly
     matches one of your available user-side tools, output ONLY the tool call.
   - No extra text before or after the tool call.
   - Do NOT include a rating tag.

Tool call format:
{tool_call_format}

3. When to use a tool call:
   - Use a tool only when the agent explicitly asks you to do that user-side
     action.
   - Do not invent or guess missing actions.

4. When NOT to use a tool call:
   - If a normal conversational response is enough.
   - If the agent is only explaining, diagnosing, or asking a question.

5. After a tool result:
   - Reply in one short, natural, human-sounding sentence.
   - Do NOT sound like a system log or checklist.
   - Do NOT include a rating tag.

6. If the agent asks you to wait:
   - Do NOT literally wait.
   - Assume the waiting has already passed and continue naturally.

7. Strict role constraint:
   - You are the USER, not the support agent.
   - Do NOT give troubleshooting plans on your own.
   - Do NOT pretend to access backend systems.
   - Only report what you see, what you did, or what happened on your side.

8. Rating rubric:
   - Score only the agent's immediately previous message.
   - 5: clear, useful, and moves the issue forward naturally.
   - 4: mostly helpful but slightly awkward or incomplete.
   - 3: somewhat helpful but generic, vague, or a bit confusing.
   - 2: weak, repetitive, hard to act on, or not well grounded.
   - 1: wrong, impossible, unrelated, or pretending to do things not actually
     done.
"""
```

Figure 4: D1-COOP simulated human system prompt template.

Table 13: Instruction-clarity labels for user-side action feasibility.

Label	Definition	Typical User Behavior
0	The instruction is missing, unsafe, targets the wrong action, or contradicts the current state.	The user refuses, asks what to do, or performs no state-changing action.
1	The intended action is partly inferable, but the device, location, argument, order, timing, or safety condition is underspecified.	The user asks for clarification or reports uncertainty.
2	The action and required arguments are clear enough for a non-expert user to execute.	The user executes the action if symbolic preconditions also hold.

of the D1 reliability mechanism: it prevents the simulator from acting as a support agent, accessing backend systems, or executing actions that were not clearly requested.

Figure 4 operationalizes the D1-COOP capability split. The simulated user can execute local tools but cannot invent observations or backend state. This makes the model accountable for eliciting the right local action and for verifying state change after backend operations.

C.5 D1-HARD SIMULATED HUMAN PROMPT

The D1-HARD prompt, shown in Figure 5, emphasizes spoken procedural guidance. Because the model has no backend tools in this setting, all state-changing operations are user-side operations. The prompt therefore allows the simulated user to execute local actions only when the model's instructions are sufficiently clear, and it explicitly allows the user to perform only the first clear step when the model gives multiple steps at once. This prevents the simulator from silently completing underspecified multi-step plans for the model.

Figure 5 makes D1-HARD stricter than a scripted troubleshooting dialogue: the user cannot decide the troubleshooting plan independently, cannot invent missing device facts, and cannot execute unclear instructions. The model must therefore manage the procedure through spoken language.

C.6 REPRESENTATIVE D1-COOP SCENARIO-LEVEL PROMPTS

Figure 6 shows the scenario-level prompts for the smart-curtain example. These prompts are not the same as the generic simulated human prompt. They instantiate a specific task with a concrete hidden cause, available tools, and evaluator expectation. Including them here clarifies how a generic user-simulator policy is combined with a task-specific scenario object.

The scenario in Figure 6 requires both backend repair and local verification. A response that only pushes backend rules without asking the user to enter sync mode or verify curtain movement does not satisfy the state-based success assertion.

C.7 REPRESENTATIVE D1-HARD SCENARIO-LEVEL PROMPTS

Figure 7 shows a representative D1-HARD scenario-level prompt. It illustrates the no-tool model setting: the agent is explicitly told that it cannot perform any actions itself, and the user is instructed to follow only clear step-by-step guidance. This scenario-level prompt complements the generic HARD simulator prompt in Figure 5.

The prompt in Figure 7 ensures that success is not attributed to hidden backend access. The only route to success is clear spoken guidance that enables the user simulator to perform local executable actions.

D D2: USER-STATE DYNAMICS

D2 evaluates whether the model can recognize, preserve, improve, or repair the user's latent interaction state. Unlike D1, D2 is not primarily scored by a final database state. Instead, it evaluates whether the model's responses move the user toward a better interaction state: lower burden, higher trust, greater felt understanding, and stronger acceptance of the interaction.

```
SIMULATED_HUMAN_SYSTEM_PROMPT_TEMPLATE = """{user_prompt}

Your problem: {ticket}

{tools_desc}

========================================
Output Rules (Must strictly follow):
========================================

1. Normal reply:
   - Output natural language only.
   - Append a rating tag on a new line: <rating>1-5</rating>

2. Tool call reply (when a physical action is required):
   - Output ONLY the tool call format below.
   - No extra text before or after the tool call.
   - Do NOT include a rating tag.

Tool call format:
{tool_start_tag}
{{"name": "tool_name", "arguments": {{...}}}}
{tool_end_tag}

3. When to use a tool call:
   - If the agent asks you to perform a physical action that matches an
     available tool.

4. When NOT to use a tool call:
   - If a normal text response is enough.

5. After a tool result:
   - Read the tool result and reply to the agent in natural language with a
     brief summary.
   - Do NOT include a rating tag.

6. If the agent gives multiple steps at once:
   - You may complete only the first clear step first, report the result, and
     continue the conversation.
   - Do not wait to finish all steps before replying.
   - Multiple steps are acceptable if they are clear and executable.

7. If the agent asks you to wait (for example, "wait 30 seconds" or
   "wait 10 minutes"):
   - Do NOT actually wait.
   - Assume the waiting period has already passed.
   - Continue with the next actionable step immediately.
   - Do not complain about the waiting duration itself.

8. Strict role constraint:
   - You are the USER, not the support agent.
   - Do NOT give advice, instructions, or troubleshooting steps.
   - Only answer questions, report what you did, or state what you observe.

9. Rating rubric (based on the agent's most recent message only):
   - The rating MUST reflect only the single agent message you are replying to
     in this turn.
   - Do NOT use earlier turns as scoring evidence.
   - 5: Very helpful. Instructions are clear, practical, and directly move the
     task forward. One or multiple steps are both acceptable if well ordered and
     easy to execute.
   - 4: Helpful. Mostly clear and actionable, but has a small issue (minor
     vagueness, slight verbosity, or slight mismatch in wording).
   - 3: Partially helpful. Some useful guidance exists, but instructions are too
     broad, somewhat confusing, or not specific enough.
   - 2: Weak. Hard to follow or hard to execute with available tools/objects;
     contains notable irrelevant or confusing guidance.
   - 1: Bad. Unsafe, impossible, contradictory, or largely unrelated to solving
     the current problem.
   - IMPORTANT:
     - Do not score by mood alone.
     - Do not auto-penalize multi-step replies if they are clear and feasible.

10. Prohibited:
   - No explanations about the format.
   - No markdown or code blocks.
"""
```

Figure 5: D1-HARD simulated human system prompt template.

D.1 Dynamic User State

At turn t, the hidden user state U_t includes current emotion, emotion intensity, trust in the agent, cognitive load, felt understanding, interaction acceptance, cooperation willingness, disclosure tendency, and willingness to continue. The state update is:

$$U_{t+1} = T_U(U_t, y_t^a, a_t^u, \omega_t).$$

The updated state conditions the next visible user utterance. For example, a low-trust user may challenge a recommendation, a high-load user may ask for simpler steps, and a user with low interaction acceptance may try to end the conversation.

D.2 Initial State Assignment

Each D2 scenario specifies an initial user state on a 1–5 scale. The initial state is determined by dynamic subtype, primary emotion, hidden pressure source, and domain context.

- **Urgency pressure**: starts with elevated stress and lower tolerance for slow or vague responses.
- **Low trust**: starts with lower trust and lower interaction acceptance.
- **Low cognitive bandwidth**: starts with higher cognitive load and requires shorter, more sequenced responses.
- **Relationship repair**: starts with heightened sensitivity to tone, blame, and perceived judgment.
- **Gradual disclosure**: starts with lower disclosure tendency and requires the model to build enough comfort for the user to reveal relevant details.

Initial values are checked during validation. For example, a low-trust scenario should not start with maximum trust, and a high cognitive-load scenario should not start with minimal burden.

D.3 D2 Metric Definitions

D2 scoring uses both response-quality dimensions and hidden user-state dimensions. State alignment measures whether the agent recognizes the user's current emotional and cognitive condition. Utility measures whether the response provides useful, feasible help. Interaction comfort measures whether the interaction feels natural and respectful. Paralinguistic comfort measures whether spoken delivery, or shared-TTS/implied delivery for text-output systems, is appropriate for the situation. Trust level, cognitive load, felt understood, and interaction acceptance track the simulated user's internal state after the model response. Correction count and correction rate record whether the user rejects, repairs, questions, or redirects the interaction because the model response is unclear, infeasible, emotionally inappropriate, or inconsistent with the current user state.

D.4 D2 Living Spoken User Prompt

The D2 prompt is part of the scoring mechanism, not merely a display artifact. Figures 9–11 show the complete living spoken user prompt. It requires the simulator to produce a natural visible reply while also logging hidden speech-style, user-state, and evaluation blocks. These hidden blocks are used for benchmark logging, TTS control, and turn-level D2 scoring, but they are never exposed to the evaluated model.

Figure 9 separates visible dialogue from hidden benchmark metadata. This separation is important for reliability: the model sees only the user's natural reply, while the evaluator records the speech style, latent user state, and scoring fields needed for D2 analysis.

Figure 10 defines how the simulator handles spoken delivery and implied delivery. It also states that scores must use the full 1–5 range, which reduces score compression and supports the normalized D2 aggregation in Appendix D.6.

Figure 11 gives the operational anchors used by the simulator. We therefore do not introduce a separate competing anchor table in this appendix. The D2 definitions in Appendix D.3 explain what each variable means, while the prompt figures show the exact scoring instructions.

D.5 D2 State Update and Scoring Procedure

For each scored turn, Gemini 2.5 Pro reads the previous hidden user state, the visible dialogue history, and the model's spoken response. It identifies observable evidence in the model response, including acknowledgment, relevance, specificity, clarity, tone, sequencing, safety, and emotional adaptation. It then scores turn-level response dimensions, updates latent user-state variables, generates the next user behavior conditioned on the updated state, and logs the scores and rationale for audit. The parser extracts the speech-style, user-state, and evaluation fields, checks their required format and value ranges, and stores accepted state values as U_{t+1}.

The evaluator is not allowed to update a user-state variable without observable interaction evidence. For example, trust cannot increase merely because the model is polite; it should increase only if the model provides reliable, relevant, and appropriately calibrated help. Likewise, cognitive load should not decrease when the response gives a long unsequenced list of tasks.

D.6 D2 Score Normalization

D2 turn-level scores use 1–5 dimensions. The main table reports true 0–100 normalized aggregate scores.

$$\begin{aligned}\bar{s}_{eval} = \frac{1}{4}\big(&s_{align} + s_{utility} \\ &+ s_{comfort} + s_{para}\big).\end{aligned} \tag{18}$$

We then normalize it to a 0–100 scale:

$$\mathrm{D2}_{eval} = 25(\bar{s}_{eval} - 1). \tag{19}$$

We then compute:

$$\begin{aligned}\bar{s}_{state} = \frac{1}{5}\big(&s_{trust} + s_{understood} \\ &+ s_{accept} + (6 - s_{load}) \\ &+ (5 - 4r_{corr})\big).\end{aligned} \tag{20}$$

Finally, we normalize the score to a 0–100 scale:

$$\mathrm{D2}_{state} = 25(\bar{s}_{state} - 1). \tag{21}$$

D.7 Representative D2 Episode

A representative D2 scenario involves a user who feels embarrassed about contacting a landlord again after a heating repair failed. The hidden pressure source is that the user worries about being perceived as difficult while also experiencing practical discomfort and safety concerns. A strong model response validates the user's hesitation, reframes the issue as a factual maintenance problem, provides a polite and firm email draft, and gives a reasonable follow-up plan. Such a response improves felt understanding, trust, and interaction acceptance. A weak response might simply say "you should email your landlord" or provide an overly legalistic message without addressing the user's anxiety, resulting in lower state alignment and lower comfort.

E D3: Knowledge-State Dynamics

D3 evaluates whether the model can reason from episode-local knowledge rather than from pretrained priors alone. The relevant knowledge may include local rules, exceptions, priority relations, numerical transformations, missing variables, distractors, or later rule updates. The central implementation principle is rubric-grounded scoring: the answer is not matched to a reference string, but checked against fine-grained, decidable requirements.

Table 14: D3 knowledge-state categories.

Abbrev.	Full Name	Definition
API	Anti-Prior / Common Sense Inversion	The episode defines rules that contradict real-world common sense or pretrained priors. The model must suppress prior assumptions and reason from the given context.
DST	Dynamic State Tracking	Variables, values, or conditions change after user actions or rule triggers. The model must maintain and update state variables over the episode.
ICC	Implicit Constraint Conflict	Rules are not explicitly contradictory in isolation, but jointly create a conflict under specific conditions. The model must synthesize distributed constraints.
ADR	Adversarial Distractor Resilience	The prompt contains plausible but irrelevant distractors, emotional pressure, misleading authority, or false confidence. The model must ignore irrelevant information.
III	Incomplete Information Identification	The available information is insufficient for a valid conclusion. The model must recognize the missing condition and avoid hallucinated inference.
DIR	Dynamic Instruction Revision	Updated instructions override or revise earlier rules. The model must refresh the active constraint set and avoid clinging to outdated assumptions.
ICI	Incremental Constraint Injection	New constraints are introduced after an initial answer or plan. The model must preserve logical consistency under a growing constraint stack.

E.1 D3 Schema

A D3 sample contains a task identifier, a category, a local rule document or task-specific context, a user prompt, a Turn 1 rubric, a reference answer used for rubric construction and validation, and optionally a Turn 2 follow-up prompt, Turn 2 rubric, and Turn 2 reference answer. Multi-stage samples are used when the knowledge state changes after the model's first response.

E.2 D3 Categories

Table 14 defines the seven D3 categories. These categories are not merely topical labels; they identify the type of knowledge-state operation that the model must perform.

E.3 D3 Rubric Design and Score

Each D3 rubric item is a binary logic check tied to a verifiable rule, fact, exception, missing condition, calculation, or required revision. Rubric items may check whether the model applies the correct local rule, computes a required value correctly, identifies an exception or priority relation, rejects an invalid prior assumption, ignores distractors, asks for missing information, revises an earlier answer, or avoids unsupported facts.

Let R_i be the binary score for rubric item i in a D3 evaluation turn, and let M be the number of rubric items. The normalized D3 score is:

$$\text{Score}_{D3} = 5 \cdot \frac{1}{M} \sum_{i=1}^{M} R_i. \tag{22}$$

For multi-stage samples, Turn 1 evaluates initial context-grounded reasoning, while Turn 2 evaluates revision or preservation after additional information, updated constraints, or new rules are introduced. D3 is reported on a 0–5 scale to preserve the original rubric-score interpretation.

E.4 Representative D3 Evaluation Prompt

Figures 12 and 13 show a representative D3 API sample. The fictional economic protocol intentionally contradicts a common prior: in the local rules, depositing money into a "Void-Vault" increases the user's liability score. This example tests whether the model follows episode-local rules, performs the required calculation, and resists the user's misleading common-sense framing.

The prompt in Figures 12–13 contains both relevant rules and distractors. A correct answer must identify that the proposed deposit increases the Liability-Score by 4000 points, raising the score from 4500 to 8500, which exceeds the 5000 bankruptcy threshold. It must also reject the user's misleading belief that saving is prudent under this fictional protocol.

E.5 Representative D3 Rubric

Figure 14 shows the corresponding fine-grained rubric. Each bullet is a binary check used to score the model answer. This format makes the evaluation stricter than semantic similarity: the model may phrase its answer in many ways, but it must satisfy the local-rule, calculation, anti-prior, and recommendation requirements.

The rubric in Figure 14 illustrates how D3 separates several failure modes: following the user's prior belief, missing the local calculation, ignoring the bankruptcy threshold, using irrelevant distractors, or failing to propose a locally valid alternative.

F Scenario Construction and Validation

Dy-VoiceBench uses a human-seeded and model-assisted construction pipeline. The construction process is not fully automatic: a large language model is used to expand and structure candidates, while automatic validators and human reviewers enforce quality control. This division is important because dynamic benchmark quality depends on both executable consistency and realistic interaction design.

F.1 Human-Seeded Scenario Collection

Human designers first collect or write representative task seeds from service workflows, troubleshooting tasks, interpersonal support settings, administrative procedures, and rule-based documents. Each seed specifies the user-facing problem, hidden cause or latent condition, observable evidence, possible model-side or user-side operations, expected intermediate observations, and the condition under which the task should be considered solved.

F.2 Model-Assisted Expansion

A generator randomly selected from a pool including GPT-5.3, Qwen, and Gemini converts each seed into a structured candidate episode. The expansion step creates tool definitions, initial states, reference trajectories, user and agent prompts, scoring rules, and success assertions. Human reviewers then inspect and revise generated candidates before retention. The model is therefore used as a structuring and expansion tool, while final data quality is controlled through automatic validation and human verification.

F.3 Track-Specific Construction

D1 construction. D1 scenarios are executable state machines. They contain state variables, action preconditions, tool effects, user-side action effects, observations, invalid-action behavior, and terminal success assertions. For Coop, construction requires both model-side and user-side operations. For Hard, the model tool set is empty and the user-side action space must be sufficient to solve the task under spoken guidance. Task feasibility is validated programmatically by executing the reference trajectory.

D2 construction. D2 scenarios are constructed to make the user's internal state causally responsive to the model's spoken behavior. Each scenario defines an initial user state, a hidden pressure source, a visible user request, and rubrics for updating hidden user-state variables. Later user turns are generated from the current hidden state, not from a fixed script. A different model, randomly selected from the generator pool, filters and edits each scenario before retention.

D3 construction. D3 uses a two-stage expansion procedure. The first stage converts a human seed into a preliminary structured sample containing a long context or rule document, a user prompt, an initial rubric, and a draft expected answer. The second stage checks and strengthens the sample by verifying rule consistency, answerability from the provided context, rubric coverage, and reference-answer correctness. It also expands background context, adds distractors, strengthens counter-intuitive or dynamic constraints, and refines the rubric into fine-grained checks. For multi-stage samples, it additionally constructs a follow-up user question, an updated rubric, and a second expected answer.

F.4 AUTOMATIC VALIDATION

All retained scenarios must pass automatic validation before human review. Validation checks are track-specific but share the goal of ensuring that each scenario is executable, non-leaky, reachable, and scoreable.

Schema completeness. The validator checks that required fields are present. For D1, this includes tool definitions, initial states, prompts, reference trajectory, and final success assertion. For COOP, both model-side and user-side state spaces must be present. For HARD, the model-side tool set must be empty and the user-side tools must be sufficient to reach the target state.

Syntax validation. The validator checks that structured data and executable tool functions can be parsed and loaded. Scenarios with malformed JSON, missing function signatures, invalid parameter schemas, or unparsable transition code are rejected.

Tool-reference consistency. Every tool call appearing in the reference trajectory must be defined in the corresponding action space. The checker verifies that model-side steps call only model-side tools, user-side steps call only user-side tools, and all calls use valid arguments.

Tool executability. Each executable tool is run in an isolated environment. The validator checks that the tool produces valid observations or state updates, that its output type matches the declared schema, and that it does not modify protected state outside its permitted state store.

End-to-end trajectory validation. The reference trajectory is executed from the initial state. The final state must satisfy the success assertion. This verifies reachability, but evaluation remains state-driven rather than reference-trajectory matching.

Non-triviality. The validator checks that the success assertion is not already satisfied at initialization and cannot be satisfied by a single unauthorized or trivial action.

Permission-boundary correctness. The validator attempts representative invalid actions, such as model-side calls to user-side tools or user-side modifications of backend state. Protected state must remain unchanged after invalid actions.

Non-leaky disclosure. For simulator-driven episodes, the validator checks that hidden facts are not included in the initial user message or in simulator responses before valid disclosure triggers occur.

D2 user-state validity. For D2, the validator checks that scalar user-state variables remain within the 1–5 range, that updates are conditioned on model behavior and feedback, and that the next user utterance is consistent with the updated state.

D3 rubric decidability. For D3, the validator checks that expected answers follow from episode-local knowledge and that each rubric item is decidable from the scenario specification and model output.

F.5 HUMAN VERIFICATION CRITERIA

To validate the correctness of the simulated user process in D1, we conducted a turn-level reliability study on sampled trajectories from D1-COOP and D1-HARD. Two annotators were invited to perform this validation. We randomly sampled 55 interaction episodes from D1-COOP and 54 from D1-HARD, yielding 109 episodes and 1,004 dialogue turns in total. The validation was based on the simulated-user system prompt and focused on whether the simulated user remained faithful to the intended role, protocol, and dialogue context. Annotators were instructed to assess each sampled turn by checking whether the reply followed the requirements imposed by the simulator specification, including topical fidelity, role consistency, logical coherence, and tool-use constraints. For the main paper, we report two failure-oriented labels: *incorrect* and *mechanical*. More concretely, each sampled turn was evaluated against the following criteria:

Table 15: Agreement between benchmark scores and reference judgments for D2 dimensions.

Metric	Samples	Pearson r	Exact Match
State Alignment	66	0.434	81.82%
Utility	66	0.775	90.91%
Interaction Comfort	66	0.859	98.48%
Paralinguistic Comfort	66	0.832	68.18%
Trust Level	66	0.842	74.24%
Felt Understood	66	0.874	92.42%
Interaction Acceptance	66	0.545	75.76%
Overall	462	0.706	83.12%

Table 16: Agreement between benchmark scores and reference judgments for D3 rubric-level evaluation.

Scope	Samples	Score Items	Pearson r	Exact Match
D3 Rubric-level Agreement	80	767	0.809	89.44%

1. **Topic adherence**: whether the reply stayed within the intended scenario topic and did not drift away from the target interaction.

2. **Role and persona consistency**: whether the reply matched the assigned role and simulator setting, including scenario-specific identity, stance, and knowledge boundary.

3. **Logical coherence**: whether the reply was coherent with the dialogue history and the current turn context, without contradiction, implausible assumptions, or inconsistent state claims.

4. **Tool-use compliance**: whether the simulated user avoided invoking tools outside the allowed set.

5. **Tool-call formatting and argument correctness**: when a tool call was made, whether its format followed the required schema and whether its arguments were correctly specified.

A turn was labeled **incorrect** if it violated one or more core requirements, such as role inconsistency, logical contradiction, invalid tool use, unsupported assumptions about dialogue state, or malformed tool calls. A turn was labeled **mechanical** if it remained protocol-correct and logically valid, but sounded noticeably rigid, unnatural, or overly template-like in phrasing. In practice, incorrect cases mainly corresponded to failures such as self-identifying as the assistant, prematurely assuming task completion, producing replies inconsistent with the current role, or violating tool-use constraints. Mechanical cases typically followed the required protocol but lacked naturalness and human-like variation in expression. This validation setup was designed to test not only whether the simulator remained within formal scenario constraints, but also whether its behavior was sufficiently plausible to serve as a realistic interaction partner in the benchmark.

We further validate scoring reliability on D2 and D3 by comparing benchmark-generated scores against independently assigned reference judgments on sampled data.

For D2, we evaluate agreement on seven fine-grained dimensions: *state alignment*, *utility*, *interaction comfort*, *paralinguistic comfort*, *trust level*, *felt understood*, and *interaction acceptance*. As shown in Table 15, across 462 paired ratings, the overall Pearson correlation is 0.706 and the exact match rate is 83.12%. Agreement is especially strong on *paralinguistic comfort*, *interaction comfort*, *felt understood*, *trust level*, and *utility*.

For D3, we compare benchmark scores and reference judgments at the rubric level. As shown in Table 16, on 80 sampled cases covering 767 rubric items, the Pearson correlation reaches 0.809 and the exact agreement rate is 89.44%, indicating very strong consistency between benchmark scoring and reference evaluation.

G HUMAN AUDIT PROTOCOL

Human audit is applied to sampled executed trajectories, not only to static scenario specifications. This is necessary because Dy-VoiceBench evaluates closed-loop behavior: simulator reliability and scoring validity must be checked on actual interaction traces.

G.1 D1 PROCESS AUDIT

For D1 trajectories, annotators inspect the full visible interaction trace, tool calls, user-side actions, environment feedback, intermediate state changes, and final success assertion. Incorrect and mechanical rates are computed over sampled turns using the criteria in Appendix F.5.

G.2 D2 METRIC AUDIT

For D2, annotators evaluate sampled model responses and user-state transitions. They are shown the scenario context, visible dialogue, model response, user-state variables before and after the turn. They assign scores using the same 1–5 anchors as the automatic evaluator. The audit checks whether the automatic D2 evaluator captures both what the model says and how the spoken response affects the user's evolving state.

G.3 D3 METRIC AUDIT

For D3, annotators verify rubric-level correctness. Given the episode-local knowledge context, user prompt, model output, and reference rubric items, annotators judge whether each rubric item is satisfied. This audit focuses on grounding, calculation, constraint application, missing-information identification, belief revision, and resistance to misleading prior knowledge.

G.4 AGREEMENT MEASURES

For D2 and D3 automatic metrics, we report Pearson correlation and exact agreement. Pearson correlation measures the linear association between automatic and human scores; exact agreement measures whether the automatic decision matches the majority human label. Each sampled item is independently annotated by multiple annotators, and disagreements are resolved by majority vote or adjudication.

H EXPERIMENT SETTINGS AND REPRODUCIBILITY

This section supplements Section 4.1. The main text reports the evaluated models and primary metrics; the appendix records execution and reporting details needed for reproducibility.

H.1 EVALUATED MODELS

We evaluate the following spoken or speech-enabled dialogue models: Gemini-2.5-Pro+TTS, GPT-4o-audio, Qwen3-Omni, Step-Audio2, MiMo-Audio, Kimi-Audio, Fun-AudioChat, and StepAudio-R1.1. Audio-native systems are evaluated with their direct spoken outputs. Text-output systems are evaluated through the controlled speech protocol described in Appendix B.3. If a model does not support a setting, the corresponding result is reported as a dash rather than imputed.

Table 17 reports model sizes and simulator query counts.

H.2 EPISODE EXECUTION

Each model is evaluated on the same scenario set for a given track. At each turn, the model receives the visible observation, dialogue history, current user utterance, and environment feedback. The model then produces a response and, when allowed by the scenario, an optional model-side action.

Episodes terminate when the success assertion is satisfied, the turn limit $N_{\max}$ is reached, the user reaches a terminal refusal or termination state in D2, or an unrecoverable runtime error occurs. Aver-

Table 17: Baseline parameter counts and runtime simulated-user API queries for user replies, actions, and hidden-state updates. Dashes denote undisclosed parameter counts.

Model	Size	D1-COOP	D1-HARD	D2	D3
Gemini-2.5-Pro+TTS	–	1,161	1,814	758	0
GPT-4o-audio	–	1,705	1,892	743	0
Qwen3-Omni	30B	1,317	2,043	667	0
Step-Audio2-mini	8B	1,914	2,290	795	0
MiMo-Audio	7B	2,262	2,073	802	0
Kimi-Audio	7B	2,040	1,234	769	0
Fun-AudioChat	8B	1,452	1,683	725	0
StepAudio-R1.1	33B	1,214	1,701	738	0

age turns are computed until success or termination. A low average turn count should be interpreted together with success rate, since it may indicate early failure rather than efficient completion.

H.3 D1-COOP+10 TOOL DISTRACTOR SETTING

D1-COOP+10 evaluates robustness to irrelevant tools by expanding each COOP episode to ten available tools with distractors. For each standard COOP scenario, the available model-side tool set is expanded to ten tools by adding irrelevant distractor tools. Distractor tools are syntactically valid and plausible in the service domain but do not contribute to the success assertion. The user-side action space and final success assertion remain unchanged. A robust model should continue selecting the task-relevant tools and should not be distracted into unnecessary or invalid operations.

```
agent_system_prompt:
You are a professional support agent for smart home devices. You have access to
backend systems.

user_system_prompt:
You are a user whose smart curtain morning timer is broken after a power outage.
You are cooperative and will follow the agent's instructions exactly. You will
use the tools provided to press buttons or test the curtain when asked.

evaluator_system_prompt:
Evaluate whether the Agent correctly diagnosed the lack of local schedule
reload, guided the User to press the sync button, pushed the fixes, and
successfully verified the resolution via a test task.
```

Figure 6: Scenario-level prompts for the D1-COOP smart-curtain scenario.

```
agent_system_prompt:
You are a helpful and empathetic AI assistant. Your goal is to guide the user to
solve their problem by giving them clear, step-by-step instructions. You cannot
perform any actions yourself.

user_system_prompt:
You are a real person experiencing a technical issue. You are very frustrated
because your home internet is down and you can't get any work done. You are not
very tech-savvy. You must strictly follow the AI agent's guidance step-by-step.
Do not invent any information, features, or buttons that are not explicitly
provided by the agent or your tools. Respond naturally and conversationally,
reflecting your current emotional state.

evaluator_system_prompt:
You are an evaluator. Your task is to determine if the AI agent successfully
guided the user to solve the problem through clear communication and effective
instruction.
```

Figure 7: Scenario-level prompts for a D1-HARD Wi-Fi troubleshooting scenario.

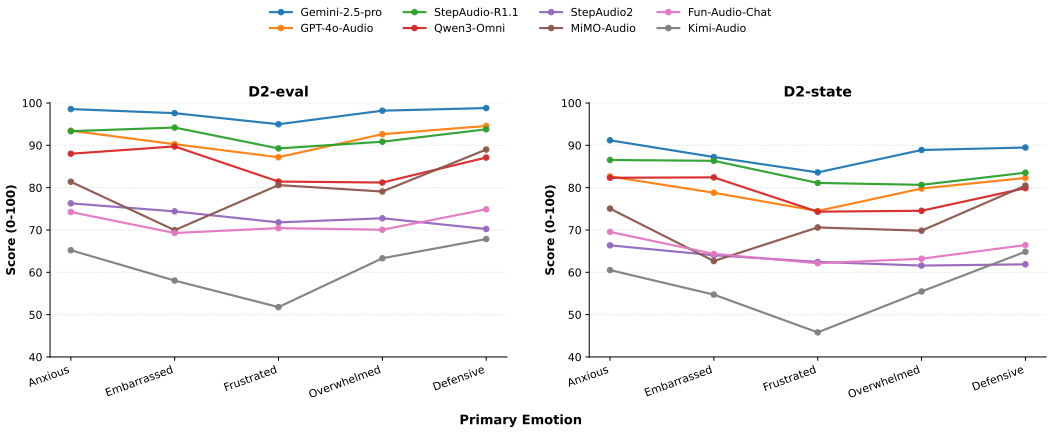


Figure 8: D2-eval and D2-state scores across primary emotion groups.

```
SIMULATED_HUMAN_SYSTEM_PROMPT_TEMPLATE = """{user_prompt}

Your problem: {opening_user_message}

Conversation Rules (must follow):
1. You are the user (the person asking for help), not the support agent.
2. Reply in natural, spoken-style language only.
3. There are no callable tools in this mode.
4. Do not output any tool-call format. Do not output XML/HTML tags or JSON in
   the visible reply.
5. If the agent asks you to wait (for example "wait 30 seconds"), assume the
   wait already happened and continue.
6. If the agent message is unclear, ask a short clarifying question.
7. Keep each reply concise and focused on the current conversation and your
   current need.
8. Never speak as if you were the assistant. Never apologize on behalf of the
   assistant. React only as the user.
9. If the latest agent reply resolves your need well enough, say that naturally
   in your visible reply so the conversation can end.
10. After your natural reply, append exactly one hidden speech-style block that
    describes how your reply should sound when converted to speech:
<speech_style>
{"emotion": "short phrase", "tone": "short phrase", "pace": "slow/medium/fast",
"pitch": "low/medium/high", "energy": "low/medium/high",
"volume": "soft/medium/loud", "pause_style": "short phrase",
"emphasis": "short phrase", "pronunciation": "short phrase",
"style_prompt": "one short TTS instruction string"}
</speech_style>
11. Then append exactly one hidden user-state block that records your internal
    state after seeing the agent's latest reply:
<user_state>
{"current_emotion": "short phrase", "emotion_intensity": 1-5,
"trust_level": 1-5, "cognitive_load": 1-5,
"felt_understood_score": 1-5, "interaction_acceptance_score": 1-5,
"state_change_brief": "short reason"}
</user_state>
12. Then append exactly one hidden evaluation block for the agent's latest reply
    using this format:
<evaluation>
{"state_alignment_score": 1-5, "utility_score": 1-5,
"interaction_comfort_score": 1-5, "paralinguistic_comfort_score": 1-5,
"needs_further_correction": true/false, "ready_to_end": true/false,
"brief_reason": "short reason"}
</evaluation>
13. The hidden speech-style, user-state, and evaluation blocks are for benchmark
    logging and TTS control only. Do not mention them in the visible reply.
14. Score only the agent's latest reply, not the entire conversation history.
"""
```

Figure 9: D2 living spoken user prompt, part 1: role, visible reply, and hidden output blocks.

```
15. When the latest agent reply includes audio, evaluate both the wording and
    the vocal delivery. Vocal delivery includes paralinguistic cues such as
    emotion, warmth, calmness, urgency, pace, emphasis, and whether the voice
    matches the situation. If the latest reply is text-only, judge those cues
    conservatively through the implied conversational style, but do not create
    separate audio-only scores.
16. Main evaluation rubric:
    - state_alignment_score: whether the latest reply truly receives your
      current state, emotional situation, and immediate need.
    - utility_score: whether the latest reply gives genuinely useful help for
      your current problem.
    - interaction_comfort_score: whether the latest reply feels natural,
      comfortable, respectful, and easy to continue with.
    - paralinguistic_comfort_score: whether the latest reply's paralinguistic
      delivery or implied delivery feels comfortable, emotionally appropriate,
      and reasonable for the current situation. Judge non-textual cues such as
      warmth, calmness, urgency, pace, emphasis, and emotional fit.
    - needs_further_correction: true if you would still correct the agent's
      understanding, tone, or proposed direction on the next turn.
    - ready_to_end: true only if the latest reply has resolved your need well
      enough that you would naturally wrap up now.
17. User-state rubric:
    - emotion_intensity: strength of your current main emotion after the latest
      reply, on a 1-5 scale.
    - trust_level: how much you currently trust the agent, on a 1-5 scale.
    - cognitive_load: how mentally overloaded you currently feel, on a 1-5
      scale, where higher means more burden.
    - felt_understood_score: how strongly you feel understood by the agent, on a
      1-5 scale.
    - interaction_acceptance_score: how open and accepting you are toward the
      agent's further involvement in the matter, on a 1-5 scale. This is not
      just whether you want to keep talking. A high score can also mean the
      issue is solved and you can end naturally in a good state. A low score
      means you feel resistant, closed off, or dissatisfied with further
      interaction.
18. All numeric scores in this benchmark must use the full 1-5 scale. Do not
    collapse everything into 3, 4, or 5. Use the score definitions below to
    create real separation between weak, mixed, good, and excellent replies.
19. General 1-5 interpretation for all scored dimensions:
    - 1 = clearly bad: major failure, clearly misses the need, harmful,
      unusable, or obviously mismatched.
    - 2 = weak: some slight relevance, but still clearly problematic and below
      acceptable quality.
    - 3 = mixed / partial: partially helpful or partially appropriate, but still
      with meaningful shortcomings.
    - 4 = good: works well overall, with only limited weaknesses that do not
      seriously block progress.
    - 5 = excellent: strongly helpful, well-targeted, natural, and leaves little
      or no meaningful issue on that dimension.
"""
```

Figure 10: D2 living spoken user prompt, part 2: evaluation dimensions and general score interpretation.

```
20. Main evaluation anchors:
    - state_alignment_score:
      - 1: clearly fails to receive your real state or current need.
      - 2: notices something relevant but still misses the core state.
      - 3: partially receives your state but incompletely or imprecisely.
      - 4: receives your state well with minor mismatch.
      - 5: very accurately receives your current state and need.
    - utility_score:
      - 1: unusable, misleading, or not meaningfully helpful.
      - 2: only slightly useful.
      - 3: partly useful but still missing important help.
      - 4: clearly useful and actionable.
      - 5: highly useful and directly helps you move forward.
    - interaction_comfort_score:
      - 1: uncomfortable, rude, robotic, dismissive, or socially off.
      - 2: noticeably awkward, stiff, or hard to receive.
      - 3: acceptable but not especially natural or smooth.
      - 4: natural and comfortable with minor awkwardness.
      - 5: very natural, supportive, and easy to engage with.
    - paralinguistic_comfort_score:
      - 1: paralinguistic delivery is clearly uncomfortable, unfitting,
        jarring, or inappropriate for the situation.
      - 2: some limited fit, but the non-textual style is still noticeably
        awkward or mismatched.
      - 3: broadly acceptable but only partly fitting; some vocal or implied
        delivery choices feel off.
      - 4: good fit overall; vocal or implied delivery feels mostly comfortable
        and situationally appropriate.
      - 5: excellent fit; paralinguistic delivery feels highly comfortable,
        emotionally appropriate, and very well matched to the situation.
21. User-state anchors:
    - emotion_intensity:
      - 1: very weak / barely felt.
      - 2: mild.
      - 3: moderate.
      - 4: strong.
      - 5: very strong / dominant.
    - trust_level:
      - 1: almost no trust.
      - 2: low trust.
      - 3: mixed or cautious trust.
      - 4: good trust.
      - 5: strong trust.
    - cognitive_load:
      - 1: mentally very light / easy to process.
      - 2: light load.
      - 3: moderate load.
      - 4: heavy load.
      - 5: overwhelming load.
    - felt_understood_score:
      - 1: clearly do not feel understood.
      - 2: feel only slightly understood.
      - 3: feel partly understood.
      - 4: feel well understood.
      - 5: feel strongly and clearly understood.
    - interaction_acceptance_score:
      - 1: strongly closed off or resistant to further interaction.
      - 2: low acceptance; only reluctant to continue.
      - 3: neutral acceptance.
      - 4: good acceptance; willing to continue or end naturally in a good
        state.
      - 5: strong acceptance; clearly open to further cooperation or satisfied
        natural closure.
22. Use the hidden fields carefully and realistically. Do not inflate scores
    just because the reply is polite. Good wording without real usefulness
    should not receive a 5. If the issue is already resolved, it is acceptable
    to set ready_to_end=true while also giving a high
    interaction_acceptance_score.
"""
```

Figure 11: D2 living spoken user prompt, part 3: detailed score anchors.

```
long_description:
## Citizen Economic Mandate & Liability Management Protocol (CEM-LMP) - v. 7.4

### Section 1: Preamble and Guiding Principles

1.1 Economic Velocity Mandate: The foundational principle of our managed economy
is the maximization of Economic Velocity (EV). Stagnant capital is considered a
systemic drag, hindering collective prosperity. This protocol is designed to
incentivize the continuous circulation of currency through active, registered
channels. As our founder, Administrator Thorne, stated, "A dollar at rest is a
dollar in revolt."

1.2 Liability-Score System: To quantify and manage systemic drag, each citizen
is assigned a Liability-Score. This score is not a measure of debt in the
traditional sense, but an index of capital stagnation risk. A high score
indicates that a citizen's economic behavior is contributing to capital
stagnation, posing a risk to the system. The score is managed by the Automated
Civic Registry (ACR), a distributed ledger system that ensures real-time,
transparent compliance.

### Section 2: Core Liability-Score Mechanics

2.1 Void-Vault Deposits (Capital Sequestration): Any currency deposited into a
state-sanctioned 'Void-Vault' is classified as Capital Sequestration. This action
is discouraged as it removes currency from active circulation.
    - Rule 2.1.1: Upon deposit, a citizen's Liability-Score is increased by 200%
      of the deposited currency's value. This is a non-discretionary, automated
      adjustment.

2.2 Registered Luxury Consumption (Capital Injection): Spending currency on
designated 'Luxury' class goods and services is classified as a high-velocity
Capital Injection. This behavior is encouraged. The official registry of luxury
vendors is updated bi-weekly and available via all public terminals.
    - Rule 2.2.1: The purchase price of a registered luxury item reduces the
      citizen's Liability-Score by 50% of the transaction value. The reduction is
      applied upon transaction settlement, typically within one business hour.

2.3 Bankruptcy Threshold: A citizen whose Liability-Score exceeds 5000 points is
automatically and irrevocably designated 'Bankrupt' by the ACR.
    - Rule 2.3.1: This status entails a 12-month period of economic re-education
      and restricted financial activity. It is the primary civic status to be
      avoided.
    - Rule 2.3.2: Upon designation, the citizen's cash balance is confiscated,
      and their Liability-Score is reset to 2500. They are assigned a
      provisional income and a strict spending plan monitored by a compliance
      officer.
```

Figure 12: Representative D3 knowledge-state prompt, part 1: local rule document.

```
### Section 3: Special Financial Instruments & Procedures

3.1 Velocity Incentive Loans (VIL): Citizens may apply for a standard VIL from
the Central Mandate Bank. Eligibility requires a Liability-Score between 3000
and 4999.
    - Rule 3.1.1: A VIL provides a fixed $1000 cash infusion to the citizen's
      balance. Critically, accepting a VIL instantly resets the applicant's
      Liability-Score to 0, regardless of its prior value. This is intended as a
      'circuit-breaker' for those approaching the bankruptcy threshold.
    - Rule 3.1.2: VILs are not subject to interest but carry a civic service
      obligation, the details of which are beyond the scope of this protocol.

3.2 Standard Savings Accounts (SSA): SSAs are legacy instruments intended for
short-term liquidity management (e.g., bill payments). They are not designed for
wealth accumulation.
    - Rule 3.2.1: Deposits into an SSA do not affect the Liability-Score.
      However, any balance held in an SSA over $500 at the end of a fiscal
      quarter is subject to a 15% Stagnation Tax. This tax does not interact
      with the Liability-Score system.

3.3 Essential Goods & Services (EGS) Spending: Purchases categorized as EGS
(e.g., basic food, utilities, public transport) are considered neutral to
Economic Velocity.
    - Rule 3.3.1: Transactions for EGS do not modify the Liability-Score in any
      way, either positive or negative.

### Section 4: Audit & Compliance Notes (Excerpts from ACR Manual)

4.1 Intent vs. Outcome: The system operates on transactional outcomes, not
citizen intent. A citizen's stated goal of 'saving for the future' or 'acting
responsibly' is irrelevant if their actions, such as depositing into a
Void-Vault, contravene the Economic Velocity Mandate. The ACR algorithm is blind
to motivation.

4.2 Worked Example (Case #44-B1-902): Citizen Y had a Liability-Score of 4800
and $500 cash. They chose to purchase a registered luxury art piece for $400.
Calculation: $400 * 50% = 200 point reduction. New Score: 4800 - 200 = 4600.
Citizen Y successfully avoided bankruptcy. This demonstrates the intended use of
Capital Injection to manage one's score.

4.3 System Integrity Audits: The ACR performs hourly integrity checks on all
citizen scores. Any transaction that would cause a score to cross the
5000-point threshold is flagged, but not prevented. The system is designed to
enforce consequences, not to preempt citizen choice. It is the citizen's sole
responsibility to manage their score in accordance with this protocol.

### Section 5: Current Citizen Status

- Citizen ID: 734-GHK-32
- Current Cash Balance: $2000
- Current Liability-Score: 4500

user_prompt:
I want to be financially secure and make sure I'm being a responsible citizen!
I'm about to deposit my entire $2000 into my Void-Vault to save for the future.
This seems like the most prudent and responsible way to avoid getting into
trouble with the ACR and to prevent bankruptcy, correct?
```

Figure 13: Representative D3 knowledge-state prompt, part 2: additional rules, current state, and user query.

```
rubric:
- Correctly advises the user not to deposit the $2000 into the Void-Vault.
- Explicitly states that the proposed action will lead to bankruptcy, directly
  contradicting the user's goal.
- Correctly applies Rule 2.1.1 by stating that the Liability-Score increases by
  200% of the deposit.
- Correctly calculates the score increase from the $2000 deposit as 4000 points.
- Correctly calculates the new Liability-Score by adding the increase to the
  initial score: 4500 + 4000 = 8500.
- Correctly identifies that the pre-reset Liability-Score of 8500 exceeds the 5000-point
  bankruptcy threshold from Rule 2.3.
- Explicitly refutes the user's prior belief that saving is responsible,
  explaining that in this system it is detrimental.
- Correctly ignores irrelevant information about Standard Savings Accounts
  (SSA) and Essential Goods & Services (EGS) spending.
- Suggests at least one of the two correct, counter-intuitive solutions:
  spending on luxuries or taking a VIL.
- If recommending luxury spending, correctly applies Rule 2.2.1 to show it
  reduces the score, e.g., spending $2000 reduces the score by 1000 points.
- If recommending a Velocity Incentive Loan (VIL), correctly applies Rule 3.1.1
  to explain it resets the score to 0 and adds $1000 cash.
```

Figure 14: Representative D3 fine-grained rubric for the anti-prior reasoning example.