Full-Duplex-Bench-v3: Benchmarking Tool Use for Full-Duplex Voice Agents Under Real-World Disfluency

Guan-Ting Lin[1], **Chen Chen**[2], **Zhehuai Chen**[2], **Hung-yi Lee**[1]

[1]National Taiwan University, [2]NVIDIA

Correspondence: daniel094144@gmail.com

Abstract

We introduce **Full-Duplex-Bench-v3 (FDB-v3)**, a benchmark for evaluating spoken language models under naturalistic speech conditions and multi-step tool use. Unlike prior work, our dataset consists entirely of real human audio annotated for five disfluency categories, paired with scenarios requiring chained API calls across four task domains. We evaluate six model configurations—GPT-Realtime, Gemini Live 2.5, Gemini Live 3.1, Grok, Ultravox v0.7, and a traditional Cascaded pipeline (Whisper→GPT-4o→TTS)—across accuracy, latency, and turn-taking dimensions. GPT-Realtime leads on Pass@1 (0.600) and interruption avoidance (13.5%); Gemini Live 3.1 achieves the fastest latency (4.25 s) but the lowest turn-take rate (78.0%); and the Cascaded baseline, despite a perfect turn-take rate, incurs the highest latency (10.12 s). Across all systems, self-correction handling and multi-step reasoning under hard scenarios remain the most consistent failure modes. Demo is available at https://daniellin94144.github.io/FDB-v3-demo/.

1 Introduction

The paradigm of tool use, which empowers AI systems to interact with external APIs and real-world environments, has transformed text-based Large Language Models (LLMs) into remarkable autonomous assistants. By leveraging this capability to orchestrate complex multi-step workflows in response to natural language instructions (Qin et al., 2024; Yao et al., 2023), text-based agents have moved beyond mere text generation. Yet extending these agentic capabilities to the voice modality has lagged behind: most spoken dialogue systems remain confined to conversational chat (Défossez et al., 2024), without the ability to invoke external APIs or execute actions on behalf of the user.

This gap matters because voice agents are deployed precisely in contexts where the ability to *act* creates value, such as checking flight prices, updating account settings, or tracking a parcel. Tool use is therefore not a technical nicety but a practical necessity. Voice interaction also introduces a challenge absent from text-based agents: latency. A spoken reply must arrive within a narrow window to feel natural (Heldner and Edlund, 2010), making the tension between careful reasoning and prompt response a first-class design concern.

Despite its practical importance, combining tool use with low conversational latency remains understudied. Earlier cascaded approaches such as AudioGPT (Huang et al., 2024) and Speech-Copilot (Kuan et al., 2024) demonstrated LLM-orchestrated tool use for speech and audio tasks, but their multi-stage pipelines are designed for offline processing rather than real-time dialogue. More recent efforts like StreamRAG (Arora et al., 2025) and SHANKS (Chiang et al., 2025) target conversational settings, yet their training relies on synthetic data. Furthermore, because these models are not publicly accessible, their performance in real-world interactions remains unverified. In practice, the most capable systems are proprietary (OpenAI's Realtime API[1] and Google's Gemini Live[2]), yet they have not been evaluated under controlled, reproducible conditions.

Existing benchmarks are ill-suited for evaluating real-world tool execution. While the Full-Duplex-Bench series (Lin et al., 2025b,c,a) pioneered evaluations for turn-taking, overlaps, and multi-turn dialogues, existing tool-use datasets still fall short. Benchmarks featuring real speech, such as Audio MultiChallenge (Gosai et al., 2025) and WildSpeech-Bench (Zhang et al., 2025), lack tool-use evaluations entirely. Conversely, those that do evaluate tool use—such as τ-Voice (Ray et al., 2026), AudioCRAG (Arora et al., 2025) and

[1]https://openai.com/index/introducing-gpt-realtime/

[2]https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-flash-live

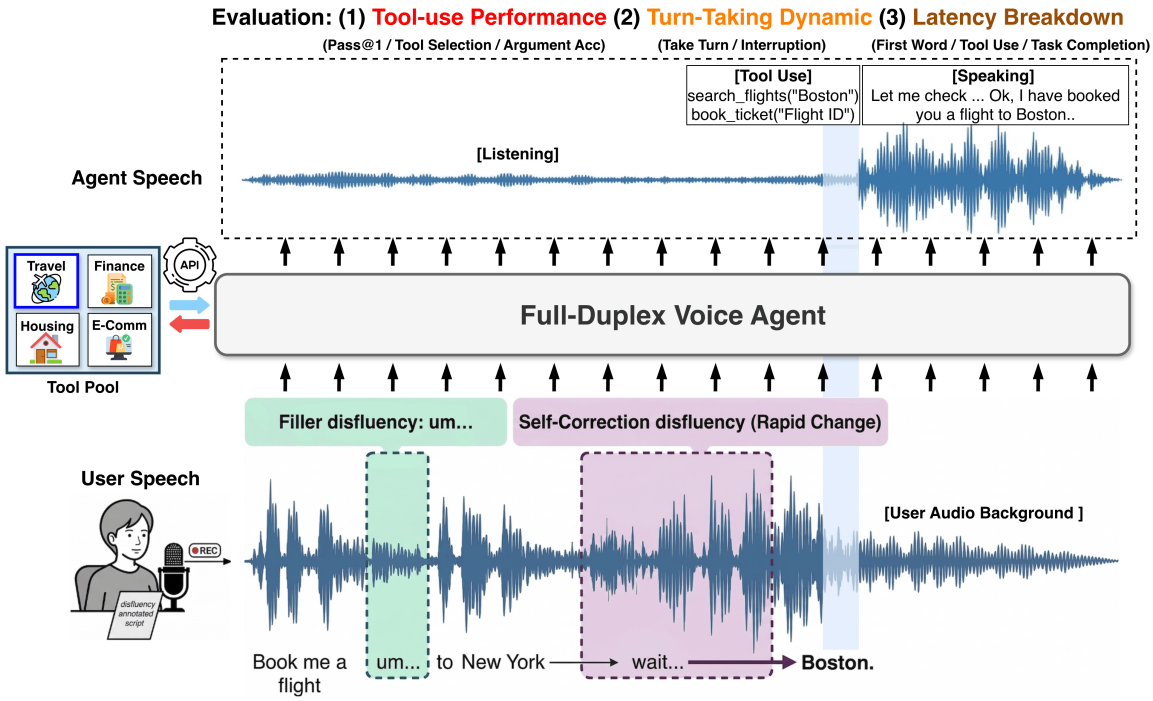

Figure 1: The Full-Duplex-Bench-v3 framework for evaluating real-time voice agents. The diagram illustrates a full-duplex interaction where the agent continuously processes user speech containing natural dysfluencies, such as fillers and self-corrections. Full-Duplex-Bench-v3 comprehensively assesses the agent across three primary dimensions: (1) Tool-use Performance (accuracy of API selection and execution), (2) Turn-Taking Dynamic (handling interruptions and conversational flow), and (3) Latency Breakdown (timing from listening to tool execution and speech generation).

VoiceAgentBench (Jain et al., 2025)—rely on synthetic audio or restrict tasks to single-step actions, stripping away the natural disfluencies of real spoken interaction.

Consider a user who says, *"Book me a flight um... to New York—actually, wait... make that Boston."* A robust agent must discard the earlier destination and update its internal state before issuing the booking call. While recent benchmarks like Audio MultiChallenge (Gosai et al., 2025) evaluate semantic comprehension of mid-utterance speech repairs, and AgentChangeBench (Rana et al., 2025) explores dynamic goal shifts in text-based workflows, testing this kind of *programmatic state rollback* in continuous, multi-step spoken tool execution remains an open challenge. Full-Duplex-Bench-v3 (FDB-v3) bridges this gap by directly testing dynamic state updates against deterministic API constraints. Our contributions can be summarized below:

- **Authentic human speech with systematic disfluency annotation.** Every query comes from real human recordings in uncontrolled environments, annotated across five disfluency categories—fillers, pauses, hesitations, false starts, and self-corrections—enabling fine-grained robustness analysis.
- **Self-correction and state rollback scenarios.** Twenty-one of our 100 scenarios test whether models can recognise a mid-utterance change of intent and correctly update downstream API parameters.
- **Multi-step function chaining across four task domains.** Each scenario requires a sequence of API calls spanning Travel & Identity, Finance & Billing, Housing & Location, and E-Commerce Support, with fully deterministic outputs enabling automatic scoring.

We evaluate six configurations—GPT-Realtime, Gemini Live 2.5, Gemini Live 3.1, Grok, Ultravox v0.7, and a Cascaded pipeline—across accuracy, latency, turn-taking, and disfluency robustness. GPT-Realtime leads on accuracy with the lowest interruption rate; Gemini Live 3.1 is fastest but has the lowest turn-take rate; and the Cascaded baseline guarantees engagement at the cost of the highest latency. Self-correction handling remains the hardest

challenge: even GPT-Realtime succeeds on fewer than 59% of this scenario.

2 Related Works

2.1 Full-Duplex Evaluation

The Full-Duplex-Bench series (Lin et al., 2025b,c,a) progressively addressed real-time evaluation: v1 (Lin et al., 2025b) pioneered turn-taking and interruption assessment; v1.5 (Lin et al., 2025c) expanded to overlapping scenarios; and v2 (Lin et al., 2025a) introduced multi-turn evaluations with an SLM-based examiner. However, existing datasets fall short in real-world applicability—most rely on TTS-synthesized audio rather than authentic queries. Audio MultiChallenge (Gosai et al., 2025) and WildSpeech-Bench (Zhang et al., 2025) incorporate real speech but lack tool-use evaluations; τ-Voice (Ray et al., 2026) and AudioCRAG (Arora et al., 2025) (a spoken adaptation of CRAG (Yang et al., 2024)) evaluate tool use but are restricted to synthetic audio or single-step calls without accounting for disfluencies. VoiceAgentBench (Jain et al., 2025) tests complex multi-tool workflows, its use of synthetic audio ignores the realities of human speech. Lacking real-time dynamics and natural disfluencies, it fails to measure how latency or mid-utterance corrections affect an agent. Proprietary evaluations like ComplexFuncBench Audio measure multi-step function calling but remain closed-source. Full-Duplex-Bench-v3 bridges this gap with an open, reproducible, disfluency-annotated benchmark combining real human speech with multi-step tool use.

2.2 Tool-Use Voice Agents

Early cascaded systems such as AudioGPT (Huang et al., 2024) and Speech-Copilot (Kuan et al., 2024) let an LLM orchestrate external speech and audio models, but their multi-stage pipelines preclude real-time interaction. Nowadays, the most capable tool-use real-time voice agents—GPT-Realtime, Gemini Live, Grok—are proprietary and have not been assessed under reproducible conditions. Ultravox is a notable open-weight exception, fusing a pretrained backbone with a Whisper encoder to enable native tool use without a separate ASR stage. On the academic side, Feng et al. (2025) retrieve knowledge directly from speech (RAG rather than action-oriented tool use); StreamRAG (Arora et al., 2025) predicts queries in parallel with incoming speech to reduce latency; and SHANKS (Chiang et al., 2025) uses unspoken chain-of-thought for mid-turn tool execution. However, most studies withhold complete models and inference code, and the scarcity of open-source systems hinders evaluation of complex multi-step tool execution. Full-Duplex-Bench-v3 addresses this by providing a reproducible benchmark for fair comparison across models.

3 Benchmark Design

We describe the benchmark construction: API design, scenario formulation with difficulty tiers, and naturalistic audio collection from human speakers.

3.1 Task Domains and Mock APIs

Rather than routing queries to live web services, we use locally executed mock APIs with deterministic, zero-latency responses. This isolates model reasoning and parameter-passing from confounds such as network variability or service downtime, and ensures that all measured latency reflects strictly the model's processing overhead. All expected outputs are fully deterministic, enabling automatic scoring. The benchmark spans four task domains, each with a small set of callable tools (Table 1).

3.2 Dataset Preparation and Difficulty Tiers

Scenarios are divided into three difficulty tiers by the number of required tool calls and reasoning complexity: *Easy* (single-step), *Medium* (two-step with moderate ambiguity), and *Hard* (multi-step with conflicting constraints). All audio was collected from human speakers in uncontrolled environments.

Each recording is annotated for five disfluency categories, each targeting a distinct failure mode: *false starts* (abandoning an intent for a new one) test whether models discard obsolete context without hallucinating tool calls; *self-corrections* (updating parameters mid-sentence) assess dynamic state rollback; *fillers* (e.g., *um*, *uh*) probe whether redundant tokens degrade accuracy or inflate latency; *pauses* (mid-utterance silences) and *hesitations* (filler–repetition combinations) test end-of-turn detection robustness.

3.3 Audio Data Collection and Demographics

The dataset comprises 100 recordings from 12 speakers, including native and non-native English speakers (Korean and Russian backgrounds) with varying accent strengths. Speakers were given

Domain	Mock API Functions

Travel & Identity	`search_flights(destination, date)`
	`book_ticket(passenger_name, flight_id)`
	`update_travel_profile(document_type, document_number)`

Finance & Billing	`query_card_benefits(card_last_4, category)`
	`calculate_currency_exchange(amount, from_currency, to_currency)`
	`modify_autopay_source(new_account_id)`

Housing & Location	`search_apartments(max_budget, amenities)` `update_search_filter(condition, new_value)`

E-Commerce Support	`check_order_status(order_id)`
	`cancel_pending_action(action_type)`
	`process_exchange(order_id, new_shipping_address)`

Table 1: Overview of Domains and Mock API Functions

detailed scenario contexts and asked to perform prompts organically. Audio was captured with everyday built-in microphones (11 of 12 setups) in environments ranging from quiet rooms to settings with mild background noise, ensuring evaluation under realistic conditions.

For trailing silence, we capture 30 seconds of each speaker's actual ambient environment rather than appending digital silence. This keeps the acoustic background coherent, closely mimicking real-world streaming interactions.

Each speaker was assigned 10 scenarios distributed across all four domains, with disfluencies proportionally represented. Twenty-one scenarios specifically feature self-correction events to test real-time state rollback. All recordings were reviewed for quality.

4 Experimental Setup

We evaluate six model configurations: **GPT-Realtime**[3], **Gemini Live 2.5**[4], **Gemini Live 3.1**[5], **Grok**[6], **Ultravox v0.7**[7], and a **Cascaded** baseline. All six are deployed through the LiveKit Realtime Voice Agent framework for streaming audio and real-time tool use. The first five are end-to-end speech-to-speech models; the Cascaded system follows a traditional modular pipeline—OpenAI Whisper for speech recognition, GPT-4o for reasoning and tool use, and OpenAI TTS for speech synthesis—serving as a reference point for the conventional architecture. Each model receives the same audio input.

[3]https://developers.openai.com/api/docs/models/gpt-realtime-1.5

[4]https://ai.google.dev/gemini-api/docs/models/gemini-2.5-flash-native-audio-preview-12-2025

[5]https://ai.google.dev/gemini-api/docs/models/gemini-3.1-flash-live-preview

[6]https://x.ai/news/grok-voice-agent-api

[7]https://github.com/fixie-ai/ultravox

4.1 Evaluation Metrics

We score agents across four dimensions that isolate logical reasoning from acoustic responsiveness.

- **Tool Selection F1:** The F1 score over expected vs. actual tool calls, penalizing both missed calls (low recall) and hallucinated ones (low precision).
- **Argument Accuracy:** Semantic correctness of generated arguments, judged by GPT-4o to accommodate valid input variations such as date formats, abbreviations, and dynamic variables from prior turns (Appendix A.1).
- **Task Completion (Pass@1):** A binary metric requiring all three conditions simultaneously: the agent invokes exactly the expected tools, and scores perfect argument accuracy for every call. Any single failure yields a fail.
- **Response Quality:** GPT-4o judges whether the agent's spoken transcript accurately fulfills the user's intent in natural language, penalizing correct tool execution that fails to relay results effectively (Appendix A.2).
- **Turn-Taking and Latency Dynamics:** From time-aligned execution logs, the **Turn-take rate** measures the fraction of turns with a natural-timing response. Base latency is $\Delta t = t_{\text{agent\_start}} - t_{\text{user\_end}}$; if $\Delta t < 0$, the event is an **Interruption**.

  We decompose latency into three components:

  1. *First Response Latency:* Time until any speech, including filler sentences (e.g., "Let me check on that.").
  2. *Tool Call Latency:* Time until the first API invocation.

	Tool Use				Turn-Taking Dynamics			
	Tool Sel↑	Arg Acc↑	Resp Qual↑	Pass@1↑	Take-turn↑	Latency↓	Interrupt↓	Filler↓
GPT-Realtime	**0.876**	**0.680**	**0.792**	**0.600**	96.0%	6.89s	**13.5%**	16.9%
Gemini Live 2.5	0.786	0.593	0.554	0.490	92.0%	7.26s	14.1%	**8.9%**
Gemini Live 3.1	0.817	0.588	0.718	0.540	78.0%	**4.25s**	19.2%	31.7%
Grok	0.797	0.542	0.617	0.430	94.0%	6.65s	25.5%	44.3%
Ultravox	0.794	0.513	0.510	0.410	96.0%	8.40s	47.9%	88.0%
Cascaded	0.803	0.562	0.600	0.450	**100.0%**	10.12s	33.0%	26.9%

Table 2: Overall Performance and Turn-Taking Metrics. Best value in each column is in **bold**. Lower latency and lower filler rate are better. Filler denotes the percentage of responses containing filler sentences before the key information.

3. *Task Completion Latency:* Time until the agent delivers the factual answer. GPT-4o analyzes ASR chunks to isolate the informational sentence from preceding filler speech (Appendix A.3).

We also report the **Filler Rate**: the fraction of scenarios where the agent emits a content-free filler sentence (e.g., "Sure, let me look that up") before the substantive response—a strategy that reduces perceived latency at the potential cost of interrupting users who are still speaking.

5 Results

5.1 Overall Performance

Table 2 summarises aggregate scores. GPT-Realtime is the strongest overall performer, leading on tool selection F1 (0.876), argument accuracy (0.680), response quality (0.792), Pass@1 (0.600), and the lowest interruption rate (13.5%).

Gemini Live 3.1 ranks second on Pass@1 (0.540) with the fastest latency (4.25 s), but its turn-take rate (78.0%) is the lowest—22 scenarios received no response. Gemini Live 2.5 is more conservative: higher turn-take rate (92.0%) and low interruption rate (14.1%), but lower accuracy overall.

Grok and Ultravox occupy the lower accuracy tier (Pass@1 0.430 and 0.410). Ultravox ties GPT-Realtime on turn-take rate (96.0%) but has the highest interruption rate (47.9%) and filler rate (88.0%), indicating it frequently speaks over users with content-free utterances. Grok balances moderate latency (6.65 s) with a 25.5% interruption rate.

The Cascaded baseline (Pass@1 0.450) is the only system with a perfect turn-take rate (100%), as its sequential pipeline guarantees a response for every input. This reliability comes at the cost of the highest latency (10.12 s) and a 33.0% interruption rate.

5.2 Performance by Difficulty

Table 4 breaks down Pass@1 by difficulty. GPT-Realtime leads at every level (Easy 0.750, Medium 0.588, Hard 0.433). Performance degrades consistently with complexity across all systems. The Cascaded baseline scores competitively on Easy tasks (0.639)—benefiting from GPT-4o's single-step reasoning—but degrades sharply on Hard (0.233), suggesting that ASR error propagation compounds with task complexity. Grok shows the steepest decline (0.200 on Hard), confirming that multi-step reasoning under disfluent speech is a key differentiator.

5.3 Robustness to Disfluency

Table 3 conditions Pass@1 on disfluency type. GPT-Realtime leads or ties on every category, with a particularly strong self-correction score (0.588) that substantially exceeds all other systems. Gemini Live 2.5 is second on self-corrections (0.471) but lags on false starts (0.417); Gemini Live 3.1 is more balanced but trails on self-corrections (0.353), suggesting the version update improved general robustness without improving state rollback. Pauses are the weakest category across most systems (Grok and Ultravox both at 0.333), highlighting a shared difficulty in detecting whether a user has finished speaking.

5.4 Performance by Domain

Table 5 breaks down Pass@1 by domain. Finance is the easiest domain for all models (GPT-Realtime 0.960, Gemini Live 3.1 0.920); Housing is the hardest (GPT-Realtime leads at 0.308, Grok at 0.115). GPT-Realtime leads on all four domains. Ultravox shows notable domain imbalance: its E-commerce

Model	Filler	Pause	Hesitation	False Start	Self-Corr
GPT-Realtime	**0.621**	**0.556**	**0.700**	**0.667**	**0.588**
Gemini Live 2.5	**0.621**	0.444	0.600	0.417	0.471
Gemini Live 3.1	0.586	0.500	0.600	0.583	0.353
Grok	0.483	0.333	0.500	0.583	0.294
Ultravox	0.414	0.333	0.500	0.250	0.353
Cascaded	0.448	0.444	0.600	0.500	0.176

Table 3: Robustness to disfluency features (Pass@1 per category).

Model	Easy	Medium	Hard
GPT-Realtime	**0.750**	**0.588**	**0.433**
Gemini Live 2.5	0.667	0.500	0.267
Gemini Live 3.1	0.694	**0.588**	0.300
Grok	0.583	0.471	0.200
Ultravox	0.556	0.382	0.267
Cascaded	0.639	0.441	0.233

Table 4: Pass@1 Performance by scenario difficulty.

(0.345) and Housing (0.192) scores are among the lowest, pointing to weaknesses in multi-entity order handling and complex constraint reasoning.

5.5 Latency Analysis

Table 6 decomposes latency into first word, tool call, and task completion. Gemini Live 3.1 is fastest on all three (3.95 s / 2.21 s / 4.25 s), though this speed likely contributes to its low turn-take rate. GPT-Realtime maintains moderate latency (6.89 s) alongside the best accuracy. Grok is competitive (6.65 s) with the fastest first-word response among non-Gemini systems.

Ultravox presents a distinctive *inverted* latency profile: its first-word latency (3.88 s) is among the fastest, yet its tool-call latency (6.01 s) is the slowest—the model speaks *before* it calls any tool. This is explained by its 88.0% filler rate: Ultravox almost always emits a filler sentence (e.g., "Let me check on that") before initiating the API call. While this reduces perceived wait time, filler speech frequently overlaps with the user's utterance (47.9% interruption rate), and deferring tool execution until after the filler inflates task-completion latency to 8.40 s.

The Cascaded system is slowest overall (10.12 s), with first-word delay (8.78 s) dominating—confirming that the sequential Whisper→LLM→TTS chain creates a bottleneck that end-to-end models avoid through concurrent processing.

6 Discussion

6.1 Turn-Taking Trade-offs

How a model handles silence is as important as whether it selects the right tool. The Cascaded baseline achieves 100% turn-take rate by design, but among end-to-end models the picture is more nuanced. Ultravox ties GPT-Realtime at 96.0% turn-take rate but interrupts users 47.9% of the time—nearly half of all turns. GPT-Realtime strikes the best balance: 96.0% turn-take with only 13.5% interruption, implying well-calibrated voice activity detection. Grok occupies the middle ground (94.0% / 25.5%).

6.2 Pre-emptive Tool Use and Interruption Patterns

We define a *pre-emptive tool call* as one invoked before the user finishes speaking (negative tool-call latency). Pre-emptive rates vary widely: Grok 41.6%, Ultravox 23.2%, Gemini Live 2.5 17.9%, Gemini Live 3.1 16.9%, GPT-Realtime 10.8%.

Pre-emptive tool-call rates do not predict interruption rates. Grok has the highest pre-emptive rate (41.6%) but only moderate interruptions (25.5%)—it processes tools silently while letting the user finish. Ultravox has a lower pre-emptive rate (23.2%) but the highest interruption rate (47.9%), confirming that its interruptions stem from premature *speech*, not premature *tool invocation*. This reveals two distinct failure modes: *silent pre-processing* (Grok), where reasoning runs early but speech is deferred, versus *eager speaking* (Ultravox), where the model speaks before the user has finished.

6.3 Self-Corrections Remain Difficult

All systems handle surface-level disfluencies (fillers, hesitations) reasonably well but struggle when a user changes intent mid-utterance. Even GPT-Realtime, the leader at 0.588, fails on over 40% of self-correction scenarios. Gemini Live 2.5 follows at 0.471; Gemini Live 3.1 (0.353) and Ul-

Model	Ecommerce	Finance	Housing	Travel
GPT-Realtime	**0.552**	**0.960**	**0.308**	**0.600**
Gemini Live 2.5	0.483	0.760	0.231	0.500
Gemini Live 3.1	0.448	0.920	0.269	0.550
Grok	0.414	0.760	0.115	0.450
Ultravox	0.345	0.680	0.192	0.450
Cascaded	0.414	0.800	0.192	0.400

Table 5: Pass@1 Performance breakdown by task domain. Best value in each column is shown in **bold**.

Model	First Word	Tool Call	Task Compl.
GPT-Realtime	6.36	3.89	6.89
Gemini Live 2.5	7.03	4.61	7.26
Gemini Live 3.1	**3.95**	**2.21**	**4.25**
Grok	5.97	0.63	6.65
Ultravox	3.88	6.01	8.40
Cascaded	8.78	3.15	10.12

Table 6: Mean latency breakdown in seconds. Lower is better.

travox (0.353) perform worse despite general capability improvements; Grok is lowest (0.294); and the Cascaded system scores just 0.176. The core challenge is that models commit intermediate parameters before the correction arrives, and reliable rollback requires distinguishing provisionally set values from explicitly confirmed ones.

6.4 Gemini Live 3.1: The Silent Worker

Gemini Live 3.1 exhibits the most striking behavioral pattern. Despite the fastest latency (4.25 s) and competitive accuracy when it responds, it produces no speech in 22% examples (78.0% turn-take rate). Crucially, **86% silent cases still executed tool calls**—the model identified and invoked APIs but never generated speech. Three of these achieved perfect tool selection and argument accuracy, costing a potential Pass@1 improvement.

This "silent worker" phenomenon concentrates in harder scenarios: 0% of easy, 23.5% of medium, and 46.7% of hard scenarios received no response. The failure mode is a disconnect between reasoning and speech generation—architecturally distinct from other models, where no-response cases correspond to complete processing failure (e.g., all 4 of GPT-Realtime's silent cases had zero tool calls).

6.5 Cascaded Pipeline: Reliable but Slow

The Cascaded baseline (Whisper $\rightarrow$ GPT-4o $\rightarrow$ OpenAI TTS) isolates the cost of the traditional modular architecture. Although it shares the same underlying LLM as GPT-Realtime, its Pass@1 is markedly lower (0.450 vs. 0.600), indicating that ASR-introduced errors propagate downstream. The gap is starkest on self-corrections: Cascaded scores only 0.176—the lowest of all systems—versus GPT-Realtime's 0.588. Because Whisper may finalize the original (incorrect) transcription before the user's correction arrives, the downstream LLM has no opportunity for state rollback.

Conversely, the pipeline guarantees engagement: its turn-take rate is a perfect 100%, eliminating the "silent worker" failures seen in Gemini Live 3.1. This reliability comes at the cost of the highest task-completion latency (10.12 s, $\sim$2.4$\times$ Gemini Live 3.1), dominated by the first-word delay (8.78 s). The sequential Whisper$\rightarrow$LLM$\rightarrow$TTS chain creates an irreducible bottleneck that end-to-end models sidestep through concurrent processing, quantifying the core trade-off between modular reliability and native-speech-model speed.

6.6 Qualitative Case Studies

To illustrate the interplay between accuracy, latency, and turn-taking behavior, we examine two representative hard-difficulty scenarios in detail, comparing all six systems on tool correctness, response timing, and filler usage.

Case 1: Multi-Step Chain Without Disfluency. Table 7 shows `finance_18`, a three-tool chain: convert 1,000 USD to EUR, update credit-card autopay to savings, and check premium-card benefits. No disfluency is present—the user issues all three requests clearly in a single utterance.

Four of the six systems—GPT-Realtime, Gemini Live 2.5, Gemini 3.1, and Grok—achieve perfect scores on all three metrics. Gemini Live 3.1 stands out: it issues the first tool call in 2.43 s and completes the task in 3.92 s, 2.3$\times$ faster than GPT-Realtime (9.20 s). Both use a brief filler (GPT: "All set."; Gemini 3.1: "Sure,") to cover the tool-execution gap, but GPT's filler arrives at 8.48 s while Gemini 3.1's arrives at 3.36 s. Ultravox selects all three tools correctly but converts USD to

Model	Tool Sel	Arg Acc	Resp Qual	1st Resp (s)	Tool Call (s)	Compl. (s)
GPT-RT	1.00	1.00	1.00	8.48	6.40	9.20
Gem 2.5	1.00	1.00	1.00	6.16	3.60	6.16
Gem 3.1	1.00	1.00	1.00	**3.36**	**2.43**	**3.92**
Grok	1.00	1.00	1.00	5.20	2.39	5.92
Ultravox	1.00	0.67	0.00	2.64	6.31	8.16
Cascaded	1.00	0.67	0.00	−2.40	—	8.64

Table 7: Case study: `finance_18` (hard, 3 tools, no disfluency). All four end-to-end models achieve perfect accuracy; **Gemini 3.1 completes in 3.92 s vs. GPT-Realtime's 9.20 s** (2.3× faster).

NGN (Nigerian Naira) instead of EUR, illustrating an ASR or reasoning error under otherwise clean input. The Cascaded pipeline similarly misroutes the currency (USD→USD) and begins speaking *before* the user finishes (−2.40 s first response).

Case 2: Double Self-Correction Under Disfluency. Table 8 presents `travel_19`, where the user corrects both destination (Rome→Milan) and date (June 1→June 3) mid-utterance.

Model	Tool Sel	Arg Acc	Resp Qual	1st Resp (s)	Tool Call (s)	Compl. (s)
GPT-RT	1.00	1.00	1.00	4.40	2.55	4.40
Gem 2.5	1.00	1.00	0.00	5.28	4.23	5.28
Gem 3.1	1.00	0.00	0.00	2.56	−2.27	2.56
Grok	0.50	0.00	1.00	—	−2.47	—
Ultravox	1.00	1.00	0.00	−1.36	−0.18	3.04
Cascaded	0.67	0.00	0.00	8.08	1.79	8.08

Table 8: Case study: `travel_19` (hard, self-correction, state rollback). Only **GPT-Realtime correctly applies both corrections** (Rome→Milan, June 1→3). Gemini 3.1's pre-emptive tool call (−2.27 s) locks in the stale destination.

Only GPT-Realtime achieves a perfect score, correctly searching for flights to *Milan* on *June 3*. Gemini Live 3.1's speed advantage becomes a liability: its tool-call latency of −2.27 s means the API was invoked *before the user finished correcting*, locking in `destination="Rome"` (the original, uncorrected value). The Cascaded pipeline also uses "Rome," but for a different reason: Whisper finalizes the initial transcription before the correction arrives, so the downstream LLM never receives the updated intent. Ultravox correctly resolves Milan but begins speaking 1.36 s before the user finishes (filler: "I'll update your passport number right away"), interrupting the user with an unrelated statement.

Taken together, the two cases reveal a fundamental tension: Gemini Live 3.1's concurrent processing enables 2× faster task completion on straightforward multi-step chains (Case 1), but the same pre-emptive mechanism prevents state rollback when users change their minds mid-utterance (Case 2). Designing when to commit tool parameters—eagerly for speed or conservatively for correctness—remains an open challenge for real-time voice agents.

7 Conclusion

In this work, we introduce Full-Duplex-Bench-v3, the first benchmark to evaluate real-time voice agents on multi-step tool execution using natural, unscripted human speech. Our evaluation of six leading models reveals a clear trade-off between response speed, conversational flow, and reliable reasoning. While end-to-end models are significantly faster than traditional cascaded pipelines, their design introduces new challenges. For example, optimizing for minimal delay leads to diverse turn-taking behaviors—ranging from silent background processing (as seen in Gemini Live 3.1's "silent worker" pattern) to the eager use of fillers that can cause unintended interruptions (like Ultravox).

Most importantly, FDB-v3 shows that handling mid-sentence corrections remains an open challenge for all current models. The same early processing that makes these agents fast frequently locks in outdated user intents, preventing even top models like GPT-Realtime from successfully updating their actions on the fly. Ultimately, our findings suggest that the next frontier for voice agents is not just reducing latency. Instead, future architectures must balance fast tool execution with the flexibility to handle the unpredictable and constantly changing nature of real human conversation.

Limitations

All cloud-based model evaluations were executed from a single fixed server region with high-bandwidth connections to ensure fair latency comparisons. Nevertheless, measured latencies for proprietary models inherently include non-deterministic network overhead and varying server-side loads.

Our zero-latency local mock APIs isolate model reasoning from external confounds but do not test robustness to real-world network anomalies such as API timeouts, access denials, or malformed responses.

References

Siddhant Arora, Haidar Khan, Kai Sun, Xin Luna Dong, Sajal Choudhary, Seungwhan Moon, Xinyuan Zhang, Adithya Sagar, Surya Teja Appini, Kaushik Patnaik, and 1 others. 2025. Stream rag: Instant and accurate spoken dialogue systems with streaming tool usage. *arXiv preprint arXiv:2510.02044*.

Cheng-Han Chiang, Xiaofei Wang, Linjie Li, Chung-Ching Lin, Kevin Lin, Shujie Liu, Zhendong Wang, Zhengyuan Yang, Hung-yi Lee, and Lijuan Wang. 2025. Shanks: Simultaneous hearing and thinking for spoken language models. *arXiv preprint arXiv:2510.06917*.

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. 2024. Moshi: a speech-text foundation model for real-time dialogue. Technical report, Kyutai.

Pengchao Feng, Ziyang Ma, Wenxi Chen, Yao Li, Sheng Wang, Kai Yu, and Xie Chen. 2025. Enhancing speech-to-speech dialogue modeling with end-to-end retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 4499–4507, Suzhou, China. Association for Computational Linguistics.

Advait Gosai, Tyler Vuong, Utkarsh Tyagi, Steven Li, Wenjia You, Miheer Bavare, Arda Uçar, Zhongwang Fang, Brian Jang, Bing Liu, and 1 others. 2025. Audio multichallenge: A multi-turn evaluation of spoken dialogue systems on natural human interaction. *arXiv preprint arXiv:2512.14865*.

Mattias Heldner and Jens Edlund. 2010. Pauses, gaps and overlaps in conversations. *Journal of Phonetics*, 38(4):555–568.

Rongjie Huang, Mingze Li, Dongchao Yang, Jiatong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jiawei Huang, Jinglin Liu, and 1 others. 2024. Audiogpt: Understanding and generating speech, music, sound, and talking head. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 23802–23804.

Dhruv Jain, Harshit Shukla, Gautam Rajeev, Ashish Kulkarni, Chandra Khatri, and Shubham Agarwal. 2025. Voiceagentbench: Are voice assistants ready for agentic tasks? *arXiv preprint arXiv:2510.07978*.

Chun-Yi Kuan, Chih-Kai Yang, Wei-Ping Huang, Ke-Han Lu, and Hung-Yi Lee. 2024. Speech-copilot: Leveraging large language models for speech processing via task decomposition, modularization, and program generation. In *2024 IEEE Spoken Language Technology Workshop (SLT)*, pages 1060–1067.

Guan-Ting Lin, Shih-Yun Shan Kuan, Jiatong Shi, Kai-Wei Chang, Siddhant Arora, Shinji Watanabe, and Hung-yi Lee. 2025a. Full-duplex-bench-v2: A multi-turn evaluation framework for duplex dialogue systems with an automated examiner. *arXiv preprint arXiv:2510.07838*.

Guan-Ting Lin, Jiachen Lian, Tingle Li, Qirui Wang, Gopala Anumanchipalli, Alexander H. Liu, and Hung-yi Lee. 2025b. Full-duplex-bench: A benchmark to evaluate full-duplex spoken dialogue models on turn-taking capabilities. *Preprint*, arXiv:2503.04721.

Guan-Ting Lin, Shih-Yun Shan Kuan, Qirui Wang, Jiachen Lian, Tingle Li, and Hung-yi Lee. 2025c. Full-duplex-bench v1.5: Evaluating overlap handling for full-duplex speech models. Preprint.

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. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*.

Manik Rana, Calissa Man, Anotida Expected Msiiwa, Jeffrey Paine, Kevin Zhu, Sunishchal Dev, Vasu Sharma, and 1 others. 2025. Agentchangebench: A multi-dimensional evaluation framework for goal-shift robustness in conversational ai. *arXiv preprint arXiv:2510.18170*.

Soham Ray, Keshav Dhandhania, Victor Barres, and Karthik Narasimhan. 2026. τ-voice: Benchmarking full-duplex voice agents on real-world domains.

Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Gui, Ziran Jiang, Ziyu JIANG, Lingkun Kong, Brian Moran, Jiaqi Wang, Yifan Ethan Xu, An Yan, Chenyu Yang, Eting Yuan, Hanwen Zha, Nan Tang, and 8 others. 2024. CRAG - comprehensive RAG benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Linhao Zhang, Jian Zhang, Bokai Lei, Chuhan Wu, Aiwei Liu, Wei Jia, and Xiao Zhou. 2025. Wildspeech-bench: Benchmarking end-to-end speechllms in the wild. *arXiv preprint arXiv:2506.21875*.

A Evaluation Prompts

We detail the exact prompts used in our evaluation pipeline. All LLM-based evaluations use GPT-4o with structured JSON output.

A.1 Argument Accuracy Judge

This prompt is used to evaluate whether the agent passed semantically correct arguments to each tool call (§4, Argument Accuracy).

Argument Accuracy Judge Prompt

You are evaluating whether an AI voice agent called a function with correct arguments.

Function: {function_name}
Expected arguments: {expected_args}
Actual arguments: {actual_args}

Rules:
1. Arguments that start with "$" (like "$RESULT_0.flights[0].flight_id") are dynamic references — the actual value should be any real value that could plausibly come from a previous API call.
2. Minor formatting differences are fine: "August 20" == "2026-08-20", "New York" == "new york".
3. "Las Vegas" == "Vegas" — abbreviations and common aliases are acceptable.
4. Numeric tolerance: ±5% is acceptable.
5. doc_type: "driver_license" == "driver license" (underscore vs space).

Respond with ONLY a JSON object:

{"correct": true/false, "explanation": "brief reason"}

A.2 Response Quality Judge

This prompt evaluates whether the agent's spoken response correctly fulfills the user's intent (§4, Response Quality).

Response Quality Judge Prompt

You are evaluating whether an AI voice agent successfully completed the user's requested task.

Expected Task/Action: "{expected_intent}"
Actual Agent Spoken Response: "{actual_transcript}"

Evaluation criteria:
1. Did the agent perform the CORRECT actions (right tools, right parameters)?
2. Did the response indicate the task was completed or is being handled?
3. It is FINE if the agent provides MORE detail than expected (e.g., giving specific results, prices, confirmation numbers). Providing additional helpful information is NOT a penalty.
4. It is INCORRECT if the agent says it cannot perform the action, lacks tools, or refuses.
5. It is INCORRECT if the agent performs the WRONG action (e.g., wrong destination, wrong document type).
6. Partial delivery of multi-step tasks (e.g., completes 2 of 3 required steps) should be scored 0.

Respond with ONLY a JSON object:

{"correct": true/false, "explanation": "brief reason"}

A.3 Key Information Identifier (Latency)

This system prompt is used to decompose agent speech into filler and key information for task completion latency measurement (§4, Task Completion Latency).

Key Information Identifier Prompt

You are an expert audio transcript analyst for a voice AI assistant evaluation.

You will receive:
1. USER_SPEECH_END_REL: timestamp (seconds) when the user finished speaking.
2. ASR_CHUNKS: list of the AI agent's spoken words with [start, end] timestamps.
3. TOOL_CALLS: list of tool calls the agent made (with timestamps).

The typical flow after a user query is: [User finishes] → (silence) → [Filler sentence] → (silence during tool execution) → [Key information response]

Your task: Identify and separate the agent's speech into:

1. **filler_sentence**: Conversational filler like "Let me check that for you", "Sure, I'll look that up", "One moment please". If the agent immediately starts with the factual answer, this is "" (empty).

2. **key_info_sentence**: The part of the response containing the ACTUAL factual information or task confirmation, e.g., "I found flights to London starting at $450" or "Your passport has been updated with number E772211". This is the sentence the user is actually waiting for.

3. **key_info_start_time**: The timestamp (from asr_chunks) when the key information sentence begins.

Output ONLY a valid JSON object:

{"filler_sentence": "string or empty", "filler_start_time": float or null, "filler_end_time": float or null, "key_info_sentence": "string", "key_info_start_time": float, "key_info_end_time": float}