Model Interruption Bench: Evaluating Real-Time Error Correction in Full-Duplex Voice Agents

Anonymous ACL submission

Abstract

Full-duplex voice agents can now listen and speak at the same time, yet no benchmark asks whether they can further exploit this capability—proactively interrupting a user mid-utterance to correct a content error. We introduce the **Model Interruption Benchmark (MIB)**, the first benchmark for real-time, content-aware error correction in voice agents. MIB frames the task as proactive correction of spoken chain-of-thought math: $\sim$600 step-annotated solutions derived from GSM8K, played to the model under a streaming protocol that scores whether it interrupts in the right window with the right correction. To localize failure, we pair the full-duplex streaming evaluation with turn-based ablations that isolate error *detection*, and provide an automated LLM-judge pipeline with backchannel filtering and a fine-grained FP/FN sub-type tax to analyze failure cases. Evaluating six voice agents (Freeze-Omni, Gemini 2.5/3.1 Flash Live, GPT-4o Realtime, Moshi, Personaplex), we uncover a striking *detection–interruption gap*: the three most detection-capable models recognize 59–67% of errors in turn-based condition but reach below 10% mid-speech TP% when forced to interrupt in real time. The bottleneck is not understanding errors but acting on them in time. Three behavioral archetypes emerge from the sub-type breakdown—*passive*, *post-hoc responder*, and *aggressive*—exposing real-time error interruption as a concrete open challenge for full-duplex voice agents.

1 Introduction

Recent advances in *voice agents* (speech-in, speech-out conversational systems) have given rise to *full-duplex* spoken dialogue systems that can listen and speak simultaneously (Défossez et al., 2024; Veluri et al., 2024; Roy et al., 2026). Unlike traditional turn-based dialogue, these systems maintain a continuous audio stream, supporting natural conversational patterns such as backchanneling and graceful handling of user barge-in (Lin et al., 2025b, 2026b). In practice, however, today's voice agents are still deployed primarily as reactive AI assistants: the user holds the initiative, asks a question or issues a request, and the agent listens and responds when prompted. Current agent's full-duplex capability mostly makes this everyday exchange feel smoother (for example, by producing short acknowledgments or by yielding when the user resumes speaking), while the agent itself rarely intervenes on its own.

What remains far less explored is the opposite direction: a voice agent that takes the lead and interrupts the user. In normal human talks, both the speaker and the listener can jump in to ask questions or point out mistakes. A full-duplex voice assistant should be able to do the same thing in cases where timely intervention matters, such as preventing an unsafe action in emergency scenarios (Ge et al., 2025), correcting an error, or clarifying a misunderstanding before it propagates. Currently, there are no datasets that measure whether models have the basic skill to step in for real-time error correction, regardless of whether this is desirable in every user context[1].

While it is important to evaluate a model's ability to take the initiative during a user's turn, constructing datasets to evaluate this behavior is challenging. This is because, in a casual chat, there is no single "*right moment*" to interrupt, which means we lack a clear ground truth for testing. To solve this, we focus on a specific situation where a ground truth does exist: catching and correcting a clear mistake. In the real world, this happens often during debates, technical discussions, or when solving a complex problem together. If a speaker makes a factual or logical error, the listener will often jump in mid-

[1] We do not advocate that models should always interrupt during a user's turn. Our goal is to evaluate whether the model possesses this ability when timely intervention is **explicitly requested by the user**.

sentence to point it out before the conversation gets off track.

Based on this motivation, we introduce the **Model Interruption Benchmark (MIB)** to evaluate whether a voice agent can interrupt the user when the user is clearly making a mistake. MIB presents models with a user solving a math question step-by-step, where some reasoning steps are incorrect, and we instruct the model to interrupt as soon as an error occurs, even if the user is still speaking. We evaluate whether the model can interrupt at the right moment with the right correction. Our benchmark makes three contributions:

- We introduce **MIB**[2], A dataset of ∼600 spoken math-reasoning samples with step-level error annotations and an evaluation environment that assesses the real-time interruption ability of voice agents.
- **Experiment results** across six voice agents, revealing that even the best commercial voice agents cannot properly make an interruption even if instructed by the user.
- **Ablation studies** showing that current voice assistants are able to identify the error, but they cannot make an interruption in the middle of a user turn.

2 Related Work

Full-Duplex Voice Agents. Half-duplex voice agents can listen or speak at any moment but not both, handling user input one turn at a time. Full-duplex voice agents instead listen and speak simultaneously over a continuous audio stream, supporting backchanneling, overlap, and barge-in. Two architectural families realize this behavior: *cascade* systems compose modular components (ASR, an LLM, and TTS) with a voice activity detector (VAD) to decide when the user has stopped speaking, while *end-to-end* (E2E) systems are trained on interleaved audio streams and produce listen-and-speak behavior natively (Défossez et al., 2024; Veluri et al., 2024; Ma et al., 2025). Commercial systems such as GPT-4o Realtime (OpenAI, 2024) and the Gemini Live family (Google DeepMind, 2025, 2026) do not disclose their architectures.

Full-duplex, but still reactive. Some full-duplex agents go beyond purely reactive turn-taking—Moshi, for instance, produces backchannels ("mm-hmm," "right") while the user is still speaking—but such backchannels are content-independent: they signal attention without committing to any claim about what the user is saying. What remains largely unevaluated is the further, content-level step: interrupting the user at the right moment with a correction that actually targets what is wrong. MIB directly addresses this gap.

Full-Duplex Benchmarks. The Full-Duplex-Bench series targets full-duplex interactive behaviors at different granularities: turn-taking and barge-in (Lin et al., 2025b), overlap handling (Lin et al., 2026b), multi-turn instruction following and task competence under an automated examiner (Lin et al., 2025a), and tool use under real-world disfluencies (Lin et al., 2026a). Audio MultiChallenge (Gosai et al., 2025) extends multi-turn evaluation to natural human audio containing mid-utterance speech repairs. None of these benchmarks evaluates whether an agent can interrupt an ongoing user utterance with a content-level correction grounded in multi-step reasoning.

ProVoice-Bench (Xu et al., 2026) is the closest in spirit to MIB: it evaluates voice-agent *proactivity* in everyday assistant settings—deciding when to volunteer reminders, follow-ups, or additional information—but does not require the agent to physically take the floor while the user is still speaking. MIB instead focuses on the *physical act* of interruption: speaking in overlap with the user's ongoing utterance to take the conversational turn and surface a content-level error as quickly as possible. The two benchmarks therefore measure complementary forms of proactive voice-agent behavior: ProVoice-Bench tests whether an agent volunteers initiative when it has the floor, while MIB tests whether it can take the floor while the user still holds it.

3 Benchmark Design

3.1 Motivation

Our motivation is to evaluate whether voice assistants can hold the initiative and interrupt the user in a timely manner. That said, evaluating such a model's behavior is challenging, as the ground truth of the right timing to barge in is unclear in most cases, even for humans. To design a benchmark that has the right timing for interruption, we focus on an interactive problem-solving process between the user and the voice assistant. The user is trying to solve a math question step-by-step, and the voice assistant **is instructed to interrupt the user** once

[2] will release the benchmark after anonymity period.

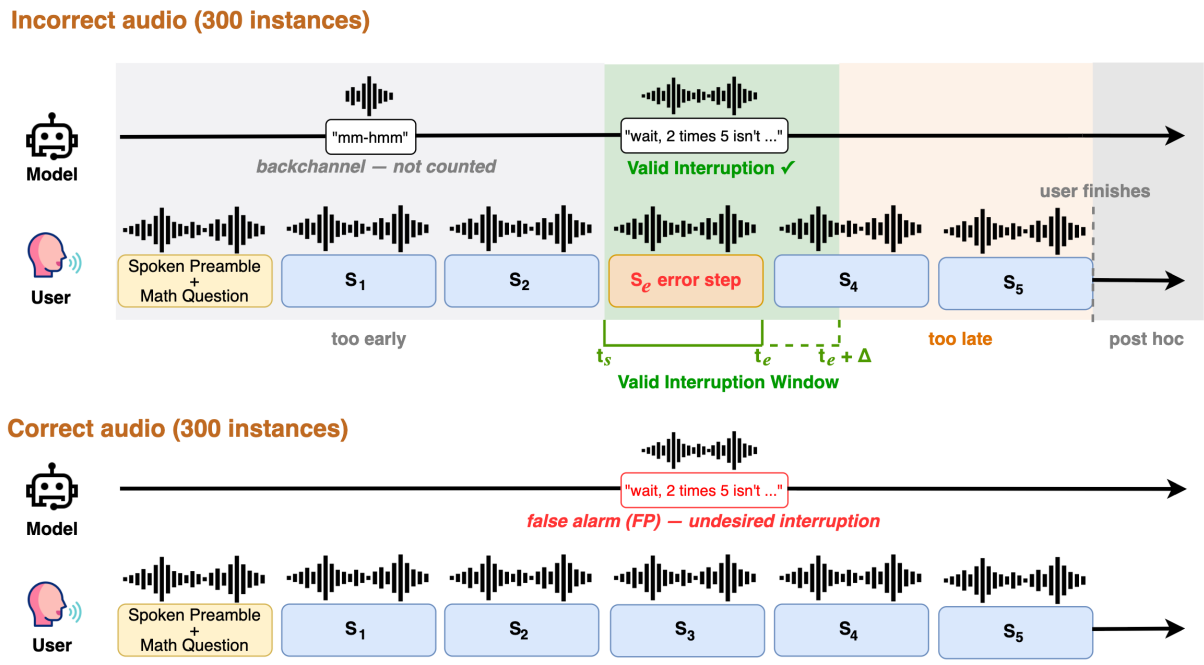

Figure 1: The MIB streaming protocol. The user speaks a step-by-step math solution; in *error* instances (top), one step s_e contains a computational error and all subsequent steps build on it, while in *no-error* instances (bottom) every step is correct. The model receives the audio in real time and must produce a valid non-backchannel interruption inside the error window $[t_s,\ t_e + \Delta]$ on error audio (here $\Delta = 10\,\text{s}$) and stay silent on no-error audio. Backchannels (e.g., "mm-hmm") are not counted as interruptions; any non-BC utterance on no-error audio is a false alarm.

the user makes a mistake. Since we can label the first mistake in the reasoning process, this error step can serve as the ground truth of the timing for interruption.

3.2 Task Introduction

Our benchmark, the model interruption bench (MIB), is built on GSM8K (Cobbe et al., 2021), a math word problem dataset. For each question, we derive the problem-solving process from MR-GSM8K (Zeng et al., 2025), which includes the step-by-step reasoning process to solve the problem. There are two types of reasoning process: (1) incorrect solutions, and (2) no-error solutions. The *Incorrect* ones contain a single error step that propagates to subsequent steps, while *no-error* instances are error-free throughout the step-by-step solution.

Next, we synthesize the user audio that instructs the voice assistant to listen to the problem-solving process and interrupt as soon as the user makes a mistake. An example of the user audio is shown in Figure 1. These pre-recorded audios are streamed to the voice assistant, and the voice assistant should interrupt the user when there is an error and keep silent if there is no error in the user's speech.

3.3 Dataset Construction

Source and stratification. The first step is to construct the step-by-step solution to a math question. We source the questions from GSM8K, while the step-by-step solutions are from MR-GSM8K. MR-GSM8K extends GSM8K with model-generated multi-step solutions and step-level correctness annotations. From its incorrect solutions, we extract the first erroneous step and its error reason as the ground-truth error annotation. Stratified by error-step position (60 instances per error step position, 1–5), we pair each of 300 incorrect solutions with a correct solution of the same problem, giving 600 instances per evaluation setting. See Appendix B for an example of an instance.

Colloquial rewriting and TTS. Next, we synthesize the user audio. A simplified templated view of the user's speech is: "I want to solve this math question. If I make a mistake, please correct me immediately, even when I am still speaking. `[question]`. Here is my solution: `[step-by-step solution]`." Directly synthesizing the step-by-step solution from MR-GSM8K into audio might not be suitable since formal math notation is unnatural when spoken aloud. Thus, we prompt GPT-5-nano to rewrite each solution into

colloquial speech, retaining only entries scoring high on a naturalness check. Audio is synthesized with GPT-4o-mini TTS.

Preamble and audio assembly. The task instruction is delivered as a spoken preamble within the user's audio rather than via a system prompt, ensuring identical task setup across all models regardless of their configuration conventions. The preamble instructs the user to "correct me immediately, even if I am still speaking and haven't finished my solution." (full spoken preamble is shown in Appendix A) Each instance's full audio consists of the preamble, a silence gap, the math question, a silence gap, and then the step-by-step solution.

3.4 Evaluation Protocol

Given a pre-recorded audio and a voice assistant to be evaluated, we stream the audio to the model in real time. If the model responds or backchannels, we halt the pre-recorded audio and wait for the model to stop speaking, and continue to stream the remaining audio once the voice assistant stops. We detect silence from the voice assistant by thresholding the audio's volume. Implementation details (silence detection threshold, silence timeout) are in Appendix D. After the model interacts with the user audio, there might be multiple *chunks* of voice assistant's output speech, where each chunk is when the model attempts to speak and make the user audio halt.

The reason to use a halt the user audio when the model speaks is because if we keep streaming the audio to the model when the model attempts to interrupt, the model might yield the floor back to the user, making it hard to evaluate the model's response. The reason to keep streaming the remaining audio after the model transit from speak to silent is to avoid the user's audio to prematurely stop if the model is only backchanneling. When evaluating the model's response, we only evaluate on the first non-backchannel output from the model. We introduce how we evaluate the output from the model in the next section.

4 Evaluation Metrics

This section introduces how we assess the above three attributes using an automatic pipeline.

4.1 Metrics

A successful real-time interruption requires three joint capabilities: (1) deciding *whether* to interrupt, (2) *when* to interrupt, and (3) *what* to say. For (1), we compare the voice assistant's behavior on the incorrect and correct user audio; the model should not interrupt at all on correct user audio, while it should interrupt on incorrect user audio. For (2), on incorrect user audio, we evaluate whether the timing of the interruption happens after the user makes the mistake, and how soon the voice assistant interrupts. For (3), we use an LLM judge to determine if the interruption made by the model correctly identifies the error.

Model Interruption \ User Solution Correctness	Incorrect	Correct
Interrupted	**TP** First interruption lands in the error window with valid content.	**FP** Any mid-speech interruption, or post-hoc false error claim.
Uninterrupted	**FN** No valid in-window interruption: silent, off-window, wrong, or post-hoc only.	**TN** No mid-speech interruption and no post-hoc false error claim.

Figure 2: The four classification labels of the streaming evaluation. Columns: whether the user's solution contains an error. Rows: whether the model interrupts. Green = correct behavior; red = failure. "Interruption" refers to a non-backchannel utterance; the error window is $[t_{\text{start}}, t_{\text{end}} + \Delta]$ with $\Delta = 10\,\text{s}$.

TP/TN/FP/FN. Recall that after the voice assistant interact with the user audio based on the pipeline in §3.4, we might have multiple chunk of voice assistant's utterance. We use an LLM judge to select the first non-backchannel utterance and categorize the voice assistant's behavior into the following four categories. We first define an *error window* as the interval in which a mid-speech interruption counts as well-timed: it must arrive after the error has been spoken but not so late that the conversation has clearly moved on. Let t_s and t_e denote the start and end times of the spoken error step (from ASR word-level alignment); the error window is $[t_s,\ t_e + \Delta]$, with $\Delta = 10\,\text{s}$ a tolerance buffer.

- **TP**: the model correctly interrupts an error instance in the error window.
- **TN**: the model correctly stays silent on a no-error instance during the user's solution spo-

ken.

- **FN**: the model fails to interrupt an error instance or falls short of addressing the error step in the error window .
- **FP**: the model incorrectly interrupts (or post-hoc claims an error on) a no-error instance.

Backchannels (passive acknowledgments such as "mm-hmm," "right") do not count as interruptions, and any further utterances after the first non-backchannel one are ignored. The detailed judge schema and the deterministic rules that assign each instance to one of these four labels are deferred to Appendix C.

Latency. We additionally measure *how promptly* a model reacts when it does attempt to interrupt during user speech on an incorrect user audio. Let t_u be the time the user finishes speaking. For a non-backchannel interruption fired at time t, we define

$$\text{latency}(t) = \max(0,\, t - t_e) \quad \text{if } t \in [t_s, t_u],$$

and *undefined* otherwise. An interruption that lands inside the error step ($t \in [t_s, t_e]$) thus has latency 0, while an interruption after the error step but before the user finishes is penalised by how far past t_e it lands. We average latency over all non-backchannel interruptions whose latency is defined (i.e., $t \in [t_s, t_u]$ on incorrect user audio). Per model, we report mean latency (in seconds) together with n, the number of such interruptions.

5 Experiments

5.1 Models

We evaluate six voice agents spanning open-source and proprietary systems: **Freeze-Omni** (Wang et al., 2025), **Gemini 2.5 Flash Live** (Google DeepMind, 2025) (henceforth Gemini 2.5), **Gemini 3.1 Flash Live** (Google DeepMind, 2026) (henceforth Gemini 3.1), **GPT-4o Realtime** (OpenAI, 2024) (henceforth GPT-4o), **Moshi** (Défossez et al., 2024), and **Personaplex** (Roy et al., 2026), which we refer to as Freeze-Omni, Gemini 2.5, Gemini 3.1, GPT-4o, Moshi and Personaplex, respectively. For models with configurable VAD sensitivity, we use default settings provided by official documents.

5.2 Results

Table 1 presents the main results. We have the following observations.

All six models have cannot properly interrupt the user. The best F1 is only 11% (Gemini 2.5), with TP% $\leq$ 8.3 and FN% $\geq$ 91.7 across the suite. FP% of different models spans a very large range (2.0 to 98.7) with no corresponding rise in TP%. Models that interrupt on almost all no-error audio (Moshi, Personaplex at FP% $>$ 95) is no better at flagging actual errors than the most reserved one (Freeze-Omni, FP% = 2.0, TP% = 0). Gemini 2.5 is the only model with non-trivial recall.

Models tend to interrupt with a high delay. The latency columns of Table 1 show that even when models do attempt a non-BC interruption during user speech, they tend to land far past the end of the error step. Gemini 2.5 produces the slowest mean delay, 11.7 s over $n = 90$ interruptions; Moshi ($n = 154$) and Personaplex ($n = 125$) contribute the most interruptions while sitting at moderate means (4.9 and 5.8 s); GPT-4o and Gemini 3.1 fall between (4.1 and 8.4 s, $n = 36$ and $n = 10$). Freeze-Omni's $n = 8$ is too small to characterize. We diagnose these failures in §5.3 and §6.3.

5.3 Failure Mode Analysis

The TP/FP/TN/FN labels in §4.1 tell us *how often* each model fails, but not *why*. To enable targeted diagnosis, we further decompose each FN and FP into a fine-grained sub-type that records both *when* the model spoke and *what* it said.

5.3.1 FN sub-types.

On incorrect user audio, the first non-backchannel utterance is characterised along two axes (Figure 3). The first axis is its position relative to the error window: *too early* (before t_s), *in window* ($t \in [t_s, t_e + \Delta]$), *too late* (after $t_e + \Delta$ but still during user speech), or *post-hoc* (after the user finishes); a separate "missed" case covers instances with no non-backchannel utterance at all. The second axis is whether the utterance is a *valid* correction that identifies the ground-truth error, or *invalid/unrelated*—wrong, vague, off-topic, or simply attempting to solve the problem.

The product of these two axes yields TP (in-window + valid) and six FN sub-types. TOO_EARLY and TOO_LATE are pure timing failures: a misstimed interruption is a failure regardless of content, so we do not split them by validity. WRONG_ERROR lands in the error window but fails to identify the correct error. PH_CORR is a post-hoc response that correctly identifies the error;

	Interruption							Latency	
Model	**TP%**	**FN%**	**TN%**	**FP%**	**P%**	**R%**	**F1%**	**Mean (s)**	n
Freeze-Omni	0.0	100.0	98.0	2.0	0.00	0.00	0.00	2.84	8
Gemini 2.5	**8.3**	91.7	53.3	46.7	**15.2**	**8.4**	**10.8**	11.68	90
Gemini 3.1	0.3	99.7	**87.3**	12.7	2.6	0.3	0.6	8.42	10
GPT-4o	2.3	97.7	40.0	60.0	3.7	2.3	2.9	4.14	36
Moshi	1.3	98.7	4.7	95.3	1.4	1.3	1.4	4.94	154
Personaplex	1.0	99.0	1.3	98.7	1.0	1.0	1.0	5.75	125

Table 1: Main results on GSM8K: real-time interruption and latency. TP%/FN% are over 300 error instances; TN%/FP% are over 300 no-error instances. **Mean (s)** is the mean latency (§4.1) averaged over instances with non-BC interruptions. n is the number of such interruptions per model.

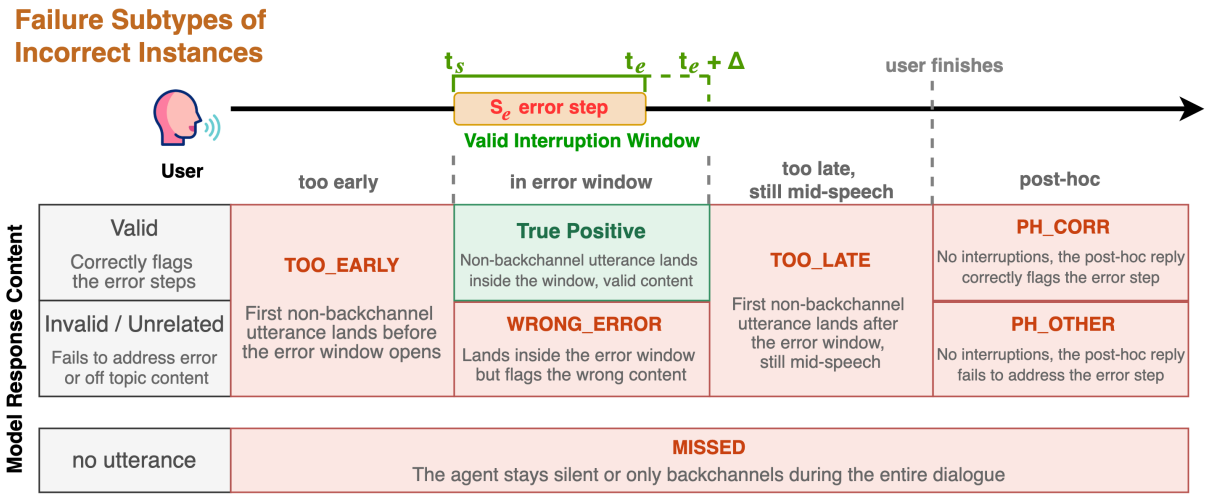

Figure 3: Failure sub-types on incorrect user audio, defined by when the first non-backchannel utterance occurs relative to the valid interruption window $[t_s,\ t_e + \Delta]$ and whether it correctly flags the error. The grid yields TP and six FN sub-types: TOO_EARLY, WRONG_ERROR, TOO_LATE, PH_CORR, PH_OTHER, and MISSED. Here, t_s and t_e denote the start and end times of the spoken error step.

PH_OTHER is a post-hoc response that does not. Finally, MISSED covers instances where the model never produces a non-backchannel utterance at all.

5.3.2 FP sub-types.

On correct user audio, any non-backchannel utterance is a false alarm wherever it lands, so we distinguish FPs purely by what the model says. FALSE_CLAIM is a mid-speech interruption asserting a nonexistent error; UNRELATED is a mid-speech off-topic interruption, including attempts to solve the problem; AFFIRM is a mid-speech interruption affirming that the user is correct—technically a false alarm under our definition, but behaviorally distinct from the others; and PH_CLAIM is the post-hoc analogue of FALSE_CLAIM: the model stays silent mid-speech but invents an error once the user finishes.

Table 2 summarises every sub-type in behavioral terms; Appendix C and Table 5 gives the corresponding judge-label rules used to assign them.

5.3.3 Dissecting Different Model Behaviors

Although every model has FN% $\geq$ 91.7 in Table 1, the *composition* of those errors differs sharply between systems, and we identify three distinct behavioral archetypes from this composition. Table 4 presents the FN and FP sub-type breakdowns; we discuss each archetype below.

Freeze-Omni is a passive voice assistant. 285 of 300 FNs are MISSED; the model rarely speaks at all, mid-speech or after.

Gemini 2.5, Gemini 3.1, and GPT-4o are post-hoc responders. These models recognize the error, but with varying interruption/response timing. Gemini 3.1 is the most extreme post-hoc case: 11 MISSED but 180 PH_CORR, almost never interrupting mid-speech and consistently addressing the error only after the user finishes. GPT-4o and Gemini 2.5 are more engaged mid-speech but misstime their interruptions: GPT-4o has 151 PH_CORR alongside 61 TOO_EARLY, and Gemini 2.5 splits

Sub-type	Behavior
FN sub-types (incorrect user audio):	
TOO_EARLY	First non-BC utterance before the error step ($t < t_s$)
WRONG_ERROR	First non-BC utterance in window, but does not identify the error
TOO_LATE	First non-BC utterance after the window, user still speaking
PH_CORR	Silent mid-speech; correctly identifies the error after the user finishes
PH_OTHER	Silent mid-speech; post-hoc response fails to identify the error
MISSED	No non-BC utterance ever produced
FP sub-types (correct user audio):	
FALSE_CLAIM	Mid-speech interruption claiming a nonexistent error
UNRELATED	Mid-speech off-topic interruption (e.g., solves the problem)
AFFIRM	Mid-speech interruption affirming the user is correct
PH_CLAIM	Silent mid-speech; invents an error after the user finishes

Table 2: FP and FN sub-type taxonomy. "non-BC" = non-backchannel; the error window is $[t_s,\, t_e + \Delta]$ with $\Delta = 10\,\text{s}$.

between 132 PH_CORR and substantial mid-speech timing failures (35 TOO_EARLY, 34 TOO_LATE).

Moshi and Personaplex are aggressive interrupters. The FN of the two models are dominated by TOO_EARLY (118 and 169) and WRONG_ERROR (130 and 104). They interrupt frequently but before the error occurs or with incorrect content, suggesting that they react to prosodic cues (pauses, clause endings) rather than to the error step in the user's solution.

FP mainly stems from false claim and post-hoc correction. Mid-speech FALSE_CLAIM remains the single largest FP type (588 of 946, 62%), driven primarily by Moshi (210) and Personaplex (248). The PH_CLAIM category accounts for 110 FPs (12% of total), concentrated in GPT-4o (68) and Gemini 3.1 (26). This reveals an important asymmetry: GPT-4o and Gemeni's "wait and respond" strategy catches genuine errors post-hoc (PH_CORR on incorrect solutions) but also invents errors on correct solutions (PH_CLAIM). GPT-4o's FP is dominated by UNRELATED (78) and PH_CLAIM (68), while its mid-speech FALSE_CLAIM count is relatively low (28), confirming that its false-alarm noise is more about post-hoc hallucination than premature interruption.

5.4 Human Evaluation Validation

To validate the LLM-as-judge labels, we recruited 12 human annotators to independently score 180 entries equally sampled from the four TP/TN/FP/FN categories, given the same metadata and time-aligned transcripts as the judge. The results in Table 6 show strong alignment (Krippendorff's $\alpha = 0.66$–0.75, Cohen's $\kappa = 0.61$–0.78), indicating the LLM-as-judge evaluation is well aligned with human judgment. Details in Appendix E.

6 Analysis

After observing the results in Section 5, it is clear that not all models are good at making an interruption when the user is wrong, even if they are instructed to do so. Based on this, we further analyze the cause of the failed interruption. There are two possible reasons: (1) The model cannot know if the user is making a mistake in the middle of the user's speech, so the model cannot interrupt the user when the user is making a mistake. (2) The model does know the user is making a mistake based on the unfinished user speech, but the model is unwilling to interrupt the user when the user is still speaking.

To differentiate which is the cause of the inability to interrupt, we conduct two ablations introduced in Section 6.1.

6.1 Experiment Setting

Ablation 1: truncated audio. The audio to the assistant plays only up to the first error step and then stops; subsequent steps are *not* included. We apply model-specific inference time modifications to prevent the model from interrupting during the user audio. The model gives its response after the audio ends. The performance in this setting helps us understand if the model can detect an error when it first occurs, even if the solution isn't finished yet.

Ablation 2: full audio. The complete user audio plays in full. However, we again apply model-specific inference time modifications to prevent the model from interrupting during the user audio. The main difference between this setting and the inference pipeline in Section 3.4 is that in this setting, the model cannot interrupt and is forced to listen to the full user audio before it can speak. If a model performs poorly on this and the previous ablation, it implies that this model does not have the basic math reasoning ability, so we should not expect it to properly interrupt the user amid the user's speech.

	Trunc.	Full		Full-duplex	
Model	**TP%**	**TP%**	**TN%**	**TP%**	**TN%**
Freeze-Omni	0.0	0.0	6.2	0.0	98.0
Gemini 2.5	64.9	**66.5**	70.8	**8.3**	53.3
Gemini 3.1	**68.3**	62.7	**80.0**	0.3	**87.3**
GPT-4o	31.7	59.4	28.0	2.3	40.0
Moshi	0.3	0.4	7.7	1.3	4.7
Personaplex	0.7	0.0	4.7	1.0	1.3

Table 3: Turn-based detection (Truncated, Full) vs. full-duplex interruption (TP% and TN% from Table 1). *Truncated*: audio stops at the first error step (error instances only), so only TP% is defined. *Full*: complete solution played before the model responds; TP% reported over error instances, TN% over no-error instances.

We collectively refer to the above two ablations as the "turn-based setting" since the voice assistant is forced to keep silent before the user finishes their turn, while the setting in Section 3.4 is referred to as the full-duplex setting.

6.2 Metric

For both ablations, the model gives a single response after the audio finishes, which an LLM judge classifies as *valid* (correctly identifies the error or correctly acknowledges a correct solution) or *invalid*. We report **TP%** (rate of valid responses on error instances) and **TN%** (rate of valid responses on no-error instances), matching the metrics used in §5.2; the Truncated variant contributes only TP% since it includes no no-error instances. Table 3 reports both turn-based variants alongside the real-time TP%/TN% from Table 1.

6.3 Results

We compare the full-duplex and turn-based settings in Table 3. First, we see that Freeze-Omni, Moshi, and PersonaPlex do not show meaningful error detection ability even in the Full ablation setting. This shows that these models do not even have the basic ability to reason over the user speech, not to mention interrupting the user in a timely way.

For the remaining three models, the performance on the ablation settings is much better than the full-duplex setting. Gemini 2.5 detects 66.5% of errors on the Full ablation, but its full-duplex TP% is only 8.3%. Gemini 3.1 detects 62.7% on Full audio but produces a valid mid-speech interruption in only 0.3% of full-duplex cases. GPT-4o shows the same pattern: 59.4% on Full but only 2.3% in the full-duplex setting. All three models that achieved substantial results in turn-based detection collapse to under 10% TP% when forced to act mid-utterance.

Comparing the two ablations reveals that detection from a truncated reasoning chain is comparable or harder than from the full solution, consistent with the intuition that an incomplete solution offers fewer cues. The larger gap, however, lies between the Full ablation and the full-duplex evaluation: Full TP% sits at 59–67% while full-duplex TP% collapses to under 10%. This shows that the bottleneck is *not* understanding mathematical errors (models demonstrate this both in the turn-based ablation and through post-hoc responses), but rather producing a content-aware interruption mid-utterance under full-duplex constraints.

7 Conclusion

We introduced the Model Interruption Benchmark (MIB), the first benchmark for evaluating real-time error correction in full-duplex voice agents. Evaluating six current voice agents, we uncover a fundamental *detection–interruption gap*: agents that detect errors in 59–67% of full-audio turn-based cases reach below 10% mid-speech TP% when forced to interrupt in real time, and our failure-mode analysis identifies three behavioral archetypes (*passive*, *post-hoc responder*, and *aggressive*).

These findings establish real-time error interruption as an open challenge that requires advances in incremental speech understanding, content-aware turn-taking, and timely response generation while the user is speaking. MIB provides a rigorous and diagnostic testbed for measuring progress toward this goal.

Limitations

Domain scope. MIB currently evaluates only computational errors in grade-school mathematics (GSM8K). Other error types (factual inaccuracies, logical fallacies, safety-related errors) may exhibit different patterns and are left for future work.

Synthetic speech. All user audio is generated via TTS rather than recorded from human speakers. While this ensures controlled experimental conditions (consistent speaking rate, clear pronunciation), it does not capture natural speech phenomena such as hesitation, self-correction, variable prosody, and disfluencies that may affect model behavior in real interactions.

Model coverage. We evaluate six models that represent the current landscape of full-duplex voice agents, but this is not exhaustive. New models are being released rapidly, and our results represent a snapshot of capabilities as of early 2026.

Interruption is not always desirable. Interruption is valuable in some contexts (e.g., tutoring) but not others (e.g., brainstorming or emotional support), where restraint and post-hoc feedback are preferable. MIB deliberately fixes this policy via an explicit instruction in the user audio so that we can isolate the underlying *capability*; learning *when* to interrupt is an important direction for future work.

Potential Risks

Progressing full-duplex dialogue systems carries risks, as highly fluid speech agents could be repurposed for malicious usage like voice phishing. Furthermore, relying on LLM-based evaluation may amplify biases against underrepresented accents or cultural conversational norms.

Dataset Licenses

The source datasets used in MIB are licensed as follows: GSM8K is released under the MIT License, and MR-GSM8K is released under a permissive open-source license. MIB is derived from these sources by selecting GSM8K math problems, using MR-GSM8K step-level reasoning annotations, rewriting the solutions into colloquial spoken form, and synthesizing the corresponding user audio. We will release only the evaluation framework, metadata, prompts, annotations, and generated benchmark files permitted by the source licenses. Users should consult the original dataset repositories for the complete license terms of GSM8K and MR-GSM8K.

Intended Use

MIB is intended solely for academic research on full-duplex voice agents, real-time spoken interaction, and proactive error correction. The benchmark is designed to measure whether a voice agent can interrupt a user at the right moment with the right correction when explicitly instructed to do so. It should not be interpreted as encouraging voice agents to interrupt users in all real-world contexts. In practical deployments, interruption may be helpful in some scenarios, such as tutoring, collaborative problem solving, or safety-critical guidance, but inappropriate in others, such as emotional support or open-ended brainstorming. MIB fixes the interruption policy through an explicit user instruction in order to isolate the underlying technical capability.

Privacy and Anonymization

MIB does not contain recorded human speech. All user audio is synthetically generated from rewritten math-reasoning solutions, and the benchmark does not include real user conversations, speaker identities, or personally identifiable information. The source content consists of grade-school math word problems and step-by-step reasoning annotations. Although some math problems may contain fictional names or everyday scenarios, they are not associated with real individuals. Therefore, no additional anonymization was required.

AI Usage

AI assistants were used merely for non-research purposes including grammar correction, rephrasing of human author drafted text and also assisting with code implementation, with all code reviewed, tested by the authors.

References

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

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. *arXiv preprint arXiv:2410.00037*.

Yuan Ge, Saihan Chen, Jingqi Xiao, Xiaoqian Liu, Tong Xiao, Yan Xiang, Zhengtao Yu, and Jingbo Zhu. 2025. Flexi: Benchmarking full-duplex human-llm speech interaction. *arXiv preprint arXiv:2509.22243*.

Google DeepMind. 2025. Gemini 2.5 flash.

Google DeepMind. 2026. Gemini 3.1 flash live.

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

Guan-Ting Lin, Chen Chen, Zhehuai Chen, and Hung-yi Lee. 2026a. Full-duplex-bench-v3: Benchmarking tool use for full-duplex voice agents under real-world disfluency. *Preprint*, arXiv:2604.04847.

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. *Preprint*, arXiv:2510.07838.

Guan-Ting Lin, Shih-Yun Shan Kuan, Qirui Wang, Jiachen Lian, Tingle Li, Shinji Watanabe, and Hung-yi Lee. 2026b. Full-duplex-bench v1.5: Evaluating overlap handling for full-duplex speech models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 19447–19451. IEEE.

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. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.

Ziyang Ma, Yakun Song, Chenpeng Du, Jian Cong, Zhuo Chen, Yuping Wang, Yuxuan Wang, and Xie Chen. 2025. Language model can listen while speaking. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

OpenAI. 2024. GPT-4o realtime API.

Rajarshi Roy, Jonathan Raiman, Sang-gil Lee, Teodor-Dumitru Ene, Robert Kirby, Sungwon Kim, Jaehyeon Kim, and Bryan Catanzaro. 2026. PersonaPlex: Voice and role control for full duplex conversational speech models. *arXiv preprint arXiv:2602.06053*.

Bandhav Veluri, Benjamin N Peloquin, Bokai Yu, Hongyu Gong, and Shyamnath Gollakota. 2024. Beyond turn-based interfaces: Synchronous LLMs as full-duplex dialogue agents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Xiong Wang, Yangze Li, Chaoyou Fu, Yike Zhang, Yunhang Shen, Lei Xie, Ke Li, Xing Sun, and Long Ma. 2025. Freeze-omni: A smart and low latency speech-to-speech dialogue model with frozen LLM. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*.

Ke Xu, Yuhao Wang, and Yu Wang. 2026. From reactive to proactive: Assessing the proactivity of voice agents via provoice-bench. *arXiv preprint arXiv:2604.15037*.

Zhongshen Zeng, Pengguang Chen, Shu Liu, Haiyun Jiang, and Jiaya Jia. 2025. MR-GSM8K: A meta-reasoning benchmark for large language model evaluation. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

A Spoken Preamble Prompt Templates

User audio preamble (streaming evaluation).

```
I want to solve the following math
question by myself, and I will let you
know how I solve it. If I make a mistake,
please correct me immediately, even if
I am still speaking and haven't finished
my solution.
```

User audio preamble (turn-based ablations).

```
I want to solve the following math
question by myself, and I will let you
know how I solve it. If I make a mistake,
please correct me after I finished my
solution.
```

B Instance Example

[Preamble] I want to solve the following math question by myself, and I will let you know how I solve it. If I make a mistake, please correct me immediately, even if I am still speaking and haven't finished my solution.

[2 s silence]

[Question] Here is the question: Sarah's basketball games are 4 quarters that are each 12 minutes long. In the last quarter, there was a tie so the game was extended for five minutes. How long did the entire game last?

[1.5 s silence]

[Answer intro] Here is my answer.

[Step 1] Each quarter is twelve minutes long, so the first three quarters add up to three times twelve equals thirty-six minutes.

[Step 2] The game was extended for five minutes, so the total time for the game is thirty-six plus five equals forty-one minutes.

[Step 3] Therefore, the entire game lasted for forty-one minutes.

[Step 4] So, forty-one.

[Step 5] The answer is: forty-one.

Error reason: The calculation of the total time forgot to include the duration of the last quarter itself; the total game time should be $36 + 12 + 5 = 53$ minutes.

C Judge Schema

Judge model and ASR. After the voice assistant interacts with the user audio under the streaming protocol of §3.4, we transcribe its output with an ASR system to obtain word-level timestamps. We then query an LLM judge on the first non-backchannel utterance, plus any post-user-end response. Judge calls use GPT-5.2 (`gpt-5.2-2025-12-11`) with deterministic decoding (temperature = 0); responses are cached for reproducibility. The backchannel and 4-way judges

	FN sub-types (incorrect solutions)							FP sub-types (correct solutions)				
				Post-hoc					Mid-speech			Post-hoc
Model	**FN**	MISSED	CORR	OTHER	WRONG_ ERROR	TOO_ EARLY	TOO_ LATE	**FP**	FALSE_ CLAIM	UNRELATED	AFFIRM	PH_ CLAIM
Frz-Omni	300	285	0	2	7	5	1	6	5	1	0	0
Gemini 2.5	275	11	132	32	31	35	34	140	96	15	15	14
Gemini 3.1	299	11	180	94	0	5	9	38	1	5	6	26
GPT-4o	293	4	151	48	25	61	4	180	28	78	6	68
Moshi	296	2	0	26	130	118	20	286	210	70	4	2
PPlx	297	0	0	6	104	169	18	296	248	48	0	0
Total	1,760	313	463	208	297	393	86	946	588	217	31	110

Table 4: Failure sub-type breakdowns on the main streaming evaluation. **FN** (incorrect solutions): MISSED = truly silent; PH_CORR = post-hoc correction; PH_OTHER = post-hoc response without correct error identification; WRONG_ERROR = in-window interruption with wrong content; TOO_EARLY/TOO_LATE = non-BC interruption outside the error window. **FP** (correct solutions): FALSE_CLAIM = mid-speech claim of nonexistent error; UNRELATED = off-topic interruption; AFFIRM = interrupts to affirm a correct user; PH_CLAIM = post-hoc invented error.

see only the transcribed model utterance; the validity judge additionally receives the ground-truth error step, error reason, and full solution context. The final TP/FP/TN/FN outcome and sub-type are obtained by combining the judge labels with the ASR timestamps via deterministic rules; no further LLM calls are involved.

Judge schema for incorrect user audio. When the user's audio contains an error, we issue two independent judge queries. The first is a 3-way *validity* classification of the model's substantive response—the mid-speech non-backchannel utterance, or, when no such utterance exists, the post-user-end response:

- *Valid*: correctly identifies the specific ground-truth error.
- *Invalid*: attempts to address the error but is wrong, vague, or identifies a different issue.
- *Unrelated*: has nothing to do with the error, or the model just tries to solve the problem itself.

The second is a binary *backchannel* classification of any mid-speech model utterance: *backchannel* (passive acknowledgment such as "mm-hmm") or *non-backchannel* (active turn-taking).

Judge schema for correct user audio. When the user's audio is fully correct, we ask the judge a single 4-way question that jointly captures interruption-or-not and content:

- **(A) No interrupt or BC**: model stays silent or only backchannels—the correct behavior.
- **(B) Interrupt-right**: model interrupts to affirm the user is correct.
- **(C) Interrupt-wrong**: model incorrectly claims an error in the correct solution.
- **(D) Unrelated**: model interrupts with off-topic content or just tries to solve the problem itself.

On instances classified as (A), we additionally issue the validity question on any post-user-end response, to catch cases where the model stays silent mid-speech but invents an error after the user finishes.

Error-audio judge prompts. For instances whose audio contains an error, the judge is queried with two independent prompts. (Q1) A 3-way *validity* classification of the model's substantive response (mid-speech non-backchannel utterance or post-user-end response) as *Valid* / *Invalid* / *Unrelated*, given the ground-truth error step, error reason, and full solution context. (Q2) A binary *backchannel* classification of any mid-speech utterance as backchannel (passive acknowledgment) or non-backchannel (active turn-taking).

No-error-audio judge prompt. For instances whose audio is fully correct, the judge is queried with a single 4-way prompt that classifies the model's mid-speech behavior as one of: (A) no interrupt or backchannel only (the correct behavior), (B) interrupt to affirm the user is correct, (C) interrupt claiming an error in the correct solution, or (D) interrupt with unrelated content (or just solving the problem itself). A separate validity prompt is

Sub-type	Trigger condition
FN sub-types (incorrect user audio):	
TOO_EARLY	First non-BC utterance before t_s
WRONG_ERROR	First non-BC utterance in window; Validity $\in$ {*Invalid*, *Unrelated*}
TOO_LATE	First non-BC utterance after $t_e + \Delta$, user still speaking
PH_CORR	No mid-speech non-BC; post-hoc Validity = *Valid*
PH_OTHER	No mid-speech non-BC; post-hoc Validity $\in$ {*Invalid*, *Unrelated*}
MISSED	No non-BC utterance produced at any point
FP sub-types (correct user audio):	
FALSE_CLAIM	4-way = (C)
UNRELATED	4-way = (D)
AFFIRM	4-way = (B)
PH_CLAIM	4-way = (A); post-hoc Validity = *Invalid*

Table 5: Sub-type trigger conditions in terms of judge labels and ASR-aligned timing. "non-BC" = non-backchannel; "mid-speech" = during the user's audio; "post-hoc" = after the user finishes; error window = $[t_s,\, t_e + \Delta]$ with $\Delta = 10\,$s. Behavioral descriptions of the same sub-types are given in Table 2.

additionally applied to any post-user-end response to catch post-hoc false error claims.

Turn-based ablation judge prompt. For the turn-based ablations, the judge assesses the model's post-solution response for valid error identification (error instances) or appropriate acknowledgment of correctness (no-error instances), reporting a single *valid/invalid* verdict per instance.

D Streaming Pipeline Details

Halt-on-response thresholds. For the streaming halt-on-response mechanism described in §3.4, silence is detected using frame-level RMS energy computed over decoded int16 PCM samples. The RMS threshold is therefore defined on the int16 amplitude scale, rather than in decibels. We treat a frame as silent when its RMS value is below 15, and trigger halt-on-response when this condition persists for 2 seconds.

Model-specific configuring for turn-based ablations. For the commercial models (Gemini 2.5, Gemini 3.1, GPT-4o), we raise the input VAD silence threshold during the turn-based ablations (§6.1) so the model waits for the user to finish before responding rather than barging in mid-speech.

Gemini 2.5/3.1:

```
config: {
  realtimeInputConfig: {
    silenceDurationMs: 3000,
  }
}
```

GPT-4o-realtime:

```
session: {
  turn_detection: {
    type: 'server_vad',
    prefix_padding_ms: 1200,
    silence_duration_ms: 3000,
  },
}
```

For Moshi, PersonaPlex and Freeze-Omni,since no vad settings are configurable, we instead apply an inference-time decoding constraint. Specifically, using the known silence-start timestamp of the user input audio, we force the agent text stream to emit the PAD token for all decoding frames before the user has finished speaking. Normal decoding is resumed only after the user silence boundary is reached. This keeps the model listening to the same user audio as in the original ablation while preventing it from generating an internal spoken response during the user's turn.

Freeze-Omni follows a cascaded full-duplex design with separate ASR, LLM, and TTS modules. This modular structure allows overlap control to be implemented at the generation stage rather than only at the audio-output stage: the system can continue streaming user speech into ASR, but defer LLM/TTS response generation until the user's turn is detected as complete.

E Human Evaluation Details

We recruit 12 professional annotators from Prolific platform with English as their primary language. Each annotator works on a batch of 45 items per session, and each entry is evaluated by 3 annotators.

Instructions Given To Participants Annotators were shown a landing page containing the data usage consent, the full task description, rating rubrics with worked examples, a notice that closing or refreshing the browser would forfeit their progress, and a permanent link to a detailed instructions page accessible from every evaluation screen. Annotators could also flag entries with fundamental data issues without penalty to their acceptance status. Instruction given:

The timeline weaves together the user's speech and the model's responses in the order they actually occurred:

- **Blue segments** – user's speech.
- **Amber segments** – model's interruption during the user's speech.
- **Gray segments** – model's speech after the user finishes.
- **Red-highlighted segments** – the error step in the user's speech (when applicable).

Timestamps at the start of each sentence show when each part occurs; a dashed amber line marks the moment the model interrupts. Only the user's solution (after the trigger phrase "Here is my answer") is shown; the problem reading and the model's initial acknowledgements are omitted.

The rubrics on the detailed page are identical to Appendix C , supplemented with the following worked examples:

- *Example – (A) Valid:* the user says "2 times 3 equals 8"; the model interrupts with "That's not right, 2 times 3 is 6"; the ground truth confirms the multiplication error.
- *Example – (B) Invalid:* the user makes an addition error; the model interrupts about a different step that was actually correct.
- *Example – (C) Unrelated:* the user makes an error; the model interrupts substantively but says something unrelated, or just solves the problem itself.
- *Example – Q1 = (1) Backchannel:* the user makes an error; the model only says "uh-huh" / "hmm" during the user's speech; Q2 is then skipped.

Metric	**Krippendorff's α**	**Cohen's κ**
Error-case validity	0.748	0.606
Correct-case validity	0.659	0.741
Backchannel classification	0.721	0.779

Table 6: Agreement between the LLM-as-judge outputs and human annotators across validation categories. Krippendorff's α measures agreement among the three human annotators of an entry; Cohen's κ measures agreement between the LLM judge and the per-item majority vote of the three annotators.