

LTS-VoiceAgent: A Listen-Think-Speak Framework for Efficient Streaming Voice Interaction via Semantic Triggering and Incremental Reasoning

Wenhao Zou^{1*}, Yuwei Miao², Zhanyu Ma¹, Jun Xu^{1†},
Jiuchong Gao^{1†}, Jinghua Hao¹, Renqing He¹, Jingwen Xu¹

¹Meituan, ²Independent Researcher

Abstract

Real-time voice agents face a tension: end-to-end models often lack deep reasoning, while cascaded ASR–LLM–TTS pipelines achieve stronger reasoning but defer LLM inference until after ASR finalizes the transcript, creating an avoidable post–turn-end silent gap. We target this gap under complex spoken requests that admit a relatively determinate answer, and treat it as system-induced rather than communicatively motivated. Existing cascaded streaming strategies attempt to reduce it via mechanical segmentation (e.g., fixed chunks, VAD-based splitting) or speculative generation, but they frequently either break semantic units or waste computation on predictions that must be rolled back. We propose LTS-VoiceAgent, a Listen–Think–Speak framework that explicitly separates *when to think* from *how to reason incrementally*. It features a Dynamic Semantic Triggerer to detect meaningful prefixes, and a Dual-Role Stream Orchestrator that coordinates a background Thinker (for state maintenance) and a foreground Speaker (for speculative solving). This parallel design enables “thinking while listening” without committing audible content before turn end. We also introduce a Pause-and-Repair benchmark that injects natural disfluencies as a semantic-level streaming stress test. Experiments across VERA, Spoken-MQA, BigBenchAudio, and our benchmark show that LTS-VoiceAgent achieves a stronger accuracy–latency–efficiency trade-off than serial cascaded baselines and existing streaming strategies.

1 Introduction

The convergence of reasoning-capable Large Language Models (Brown et al., 2020; Achiam et al., 2023; Wei et al., 2022) and high-fidelity speech foundation models (Radford et al., 2023; Wang et al., 2023) has enabled Voice AI to evolve from

*Work done during internship at Meituan.

†Corresponding authors.

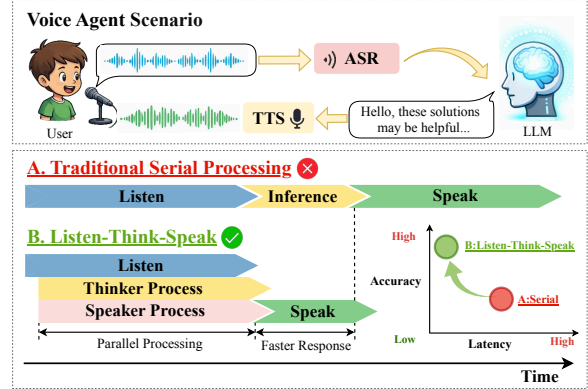


Figure 1: Traditional serial processing vs. LTS-VoiceAgent. The bottom-right inset is a conceptual illustration; quantitative results are reported in Sec. 5.

simple command-execution bots into conversational agents that can handle substantive, multi-step spoken tasks such as math tutoring and open-domain QA (Hurst et al., 2024; Huang et al., 2024). We target this setting, where the user issues a content-bearing request that admits a relatively determinate answer and the dominant source of perceived sluggishness is the post–turn-end silent gap induced by the system rather than communicatively motivated devices such as backchannels, placeholders, or deliberately delayed responses that may carry pragmatic signals (Levinson, 2016; Stivers et al., 2009; Tormala et al., 2011).

While End-to-End (E2E) models reduce latency (Fang et al., 2024; Défossez et al., 2024), they still lag behind text-only LLMs in complex reasoning, making cascaded ASR–LLM–TTS pipelines the dominant solution for substantive spoken interaction (Chen et al., 2024b; Jain et al., 2025). However, in standard cascades the LLM only begins substantial reasoning after ASR finalizes the transcript, leaving the user waiting in this avoidable silent gap (Skantze, 2021). As contrasted in Figure 1, traditional serial pipelines (Figure 1A) prioritize accuracy but suffer from this gap, whereas our LTS

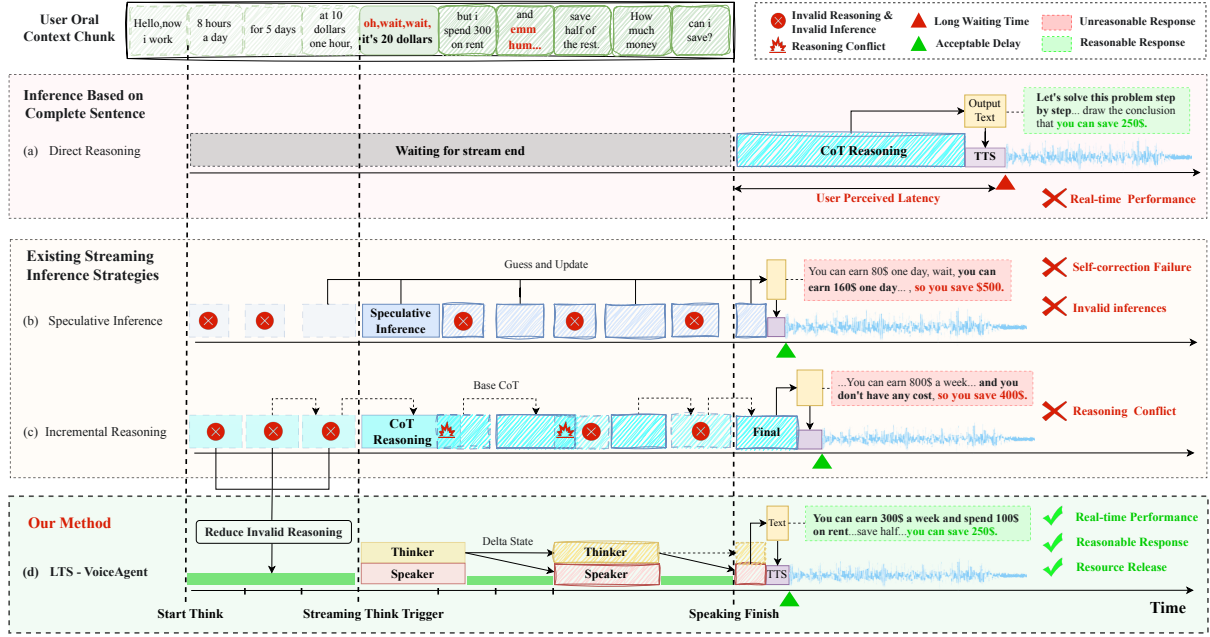


Figure 2: Different reasoning paradigms under streaming spoken input. Panels (a)–(d) illustrate, respectively: (a) serial cascade with post–turn–end waiting; (b) speculative pre-generation (Li and Grover, 2025); (c) chunk- or VAD-triggered incremental reasoning (Chen et al., 2024a); and (d) our LTS-VoiceAgent. The depicted “post–turn–end waiting time” is a *qualitative* positioning of system-induced delay; concrete delay magnitudes (sub-second to tens of seconds depending on backbone and reasoning mode) are reported in Table 1.

framework (Figure 1B) shifts understanding and reasoning preparation into the listening phase. Further mechanism analysis in Figure 2(a) reveals the root cause: serial systems are architecturally bound to silence while the user speaks. Existing streaming strategies introduce parallelism, but still suffer from inefficient speculation and fragmented reasoning. Guessing the user’s intent before turn end (Li and Grover, 2025) can reduce waiting, but as Figure 2(b) shows, intent shifts (e.g., “oh, wait...”) force rollbacks and waste computation. Chunk- or VAD-triggered reasoning (Figure 2(c)) often fires on semantically incomplete fragments; since VAD relies on acoustic silence rather than semantic completeness (Chen et al., 2024a), hesitations can still trigger invalid reasoning and state conflicts.

To address these challenges, we propose LTS-VoiceAgent, a **Listen–Think–Speak** framework for deciding *when* to think and *how* to reason incrementally. As shown in Figure 2(d), LTS-VoiceAgent uses a *Dynamic Semantic Trigger* to filter non-informative fragments and activates reasoning only at semantically sufficient moments. A Stream Orchestrator then maintains a continuous reasoning chain: a background *Thinker* refines the state, while a foreground *Speaker* prepares candidate responses from the evolved context. This enables reuse and

self-correction during pauses, reducing post–turn–end delay. Our framing builds on incremental dialogue processing (Schlangen and Skantze, 2011; DeVault et al., 2009; Kahardipraja et al., 2023), but targets system-level waiting in autoregressive LLM cascades.

In summary, our contributions are as follows:

- *Dynamic Semantic Trigger.* We propose a lightweight module that identifies valid semantic boundaries within the continuous stream. By detecting when the input context is semantically sufficient, it triggers reasoning at appropriate commit points, reducing redundant computation caused by premature execution.
- *Dual-Role Stream Orchestrator.* We design a collaborative inference architecture that leverages batch processing to parallelize understanding and generation. By coordinating a Thinker for robust state tracking (input sanitization and planning) and a Speaker for speculative execution, this mechanism helps the model handle “pause-and-repair” behaviors and intent drifts effectively while achieving sub-second response latency.
- *Pause-and-Repair Benchmark.* To evaluate

streaming agents under controlled spoken-style perturbations, we construct a dataset based on GSM8K and MMLU-Pro, enriched with hesitations, self-corrections, and varying speaking rates as a semantic-level stress test. Experiments show that LTS-VoiceAgent improves the latency–quality–efficiency trade-off while reducing inference costs compared to strong baselines.

2 Related Work

We focus on three lines most relevant to LTS-VoiceAgent; broader connections to incremental dialogue processing and simultaneous translation are in Appendix B.

2.1 End-to-End Omni-Modal Models

End-to-end models remove intermediate text transcription by mapping audio directly to semantic representations. Prior work explores simultaneous decoding (Ren et al., 2020), parallel speech decoding (Fang et al., 2024, 2025), speech–LLM alignment (Wang et al., 2024a), and speech-in/speech-out modeling (Zhang et al., 2024; Long et al., 2025; Shi et al., 2025). Yet these omni-modal models still lag behind text LLMs in complex reasoning and controllable instruction following.

2.2 Advanced Cascaded Architectures: Component Acceleration

Cascaded ASR–LLM–TTS systems remain attractive for reasoning and controllability. Recent work accelerates individual components, including ASR speculative decoding (Wei et al., 2025b), streaming TTS (Du et al., 2024; Shikhar et al., 2025), and Audio-LLM front-ends (Li et al., 2024). However, most efforts still optimize components within a serial pipeline where LLM reasoning starts only after a complete transcript.

2.3 Streaming Inference and Speculative Strategies

Streaming inference aims to break serial waiting through input-time speculation (Li and Grover, 2025), clause/sentence segmentation and context reuse (Chen et al., 2024a; Likhomanenko et al., 2025; Ethiraj et al., 2025), or CoT-style reasoning/acceleration (Goel et al., 2025; Woo et al., 2025; Cheng and Van Durme, 2024; Xu et al., 2025b; Hou et al., 2025). These approaches either roll back invalid guesses, rely on mechanical segmentation, or assume static complete inputs,

leaving semantic triggering and state reuse under streaming revisions underexplored.

In summary, LTS-VoiceAgent reduces avoidable post–turn-end waiting in cascaded voice agents by replacing mechanical segmentation with semantic triggering and coordinating background state maintenance (*Thinker*) with foreground speculative generation (*Speaker*).

3 Methodology

This section details LTS-VoiceAgent, a unified framework designed to address the challenges of *when* to think and *how* to reason incrementally in streaming voice interactions. As illustrated in Figure 3, the system consists of two core components: (1) a *Dynamic Semantic Trigger*, which detects reasoning-worthy semantic boundaries in real-time streaming text; and (2) a *Dual-Role Stream Orchestrator*, which coordinates a background *Thinker* and a foreground *Speaker* to reduce post–turn-end response delay through asynchronous state updates.

3.1 Overview

In standard cascaded voice interaction, the text stream output by the ASR, denoted as $S = \{s_1, s_2, \dots, s_t\}$, grows continuously. Traditional strategies often fall into two extremes: either relying on acoustic VAD, which leads to frequent interruptions by meaningless hesitations (e.g., “um...”), or employing aggressive speculative generation that results in computational waste. LTS-VoiceAgent proposes a Listen–Think–Speak paradigm based on semantic awareness.

The system workflow is as follows. Text chunks incrementally output by the ASR are first processed by the Semantic Trigger. This module estimates the semantic saturation probability P_{trigger} of the current prefix $S_{1:t}$. Only when $P_{\text{trigger}} > \tau$ does the system activate the LLM for reasoning. Consequently, the model performs two types of reasoning: (1) *thinking while listening*, where the model updates internal states during the user’s speech; and (2) *direct responding*, where the model commits an answer. To address overlap and correction issues in streaming input, the system continuously maintains a reasoning state table that preserves user intent and intermediate deductions. Once the user finishes speaking, the final response is generated based on this accumulated state table.

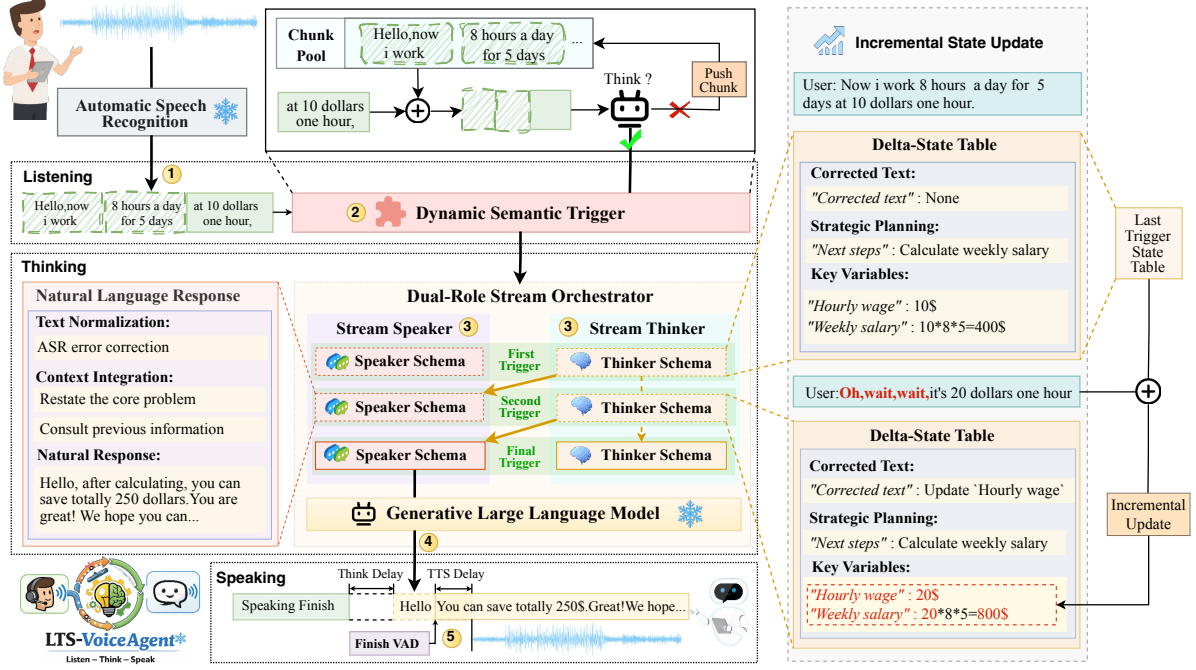


Figure 3: Overview of LTS-VoiceAgent.

3.2 Dynamic Semantic Trigger

The core objective of the trigger is to allocate computational costs to moments characterized by a “significant information increment” and “sufficient time for reasoning,” rather than mere acoustic silence. We model this as a streaming binary classification task.

3.2.1 Data Synthesis via Semantic Alignment

Due to the scarcity of large-scale annotated “semantic breakpoints,” we designed an LLM-assisted synthetic supervision pipeline. We select raw texts D_{raw} from representative reasoning-intensive corpora to serve as the seed data. The pipeline is intentionally *data-agnostic*: by swapping the seed corpus to in-domain text or dialogue data (e.g., short voice commands, slot-filling scripts, or open-domain chitchat) and rerunning the same automated annotation, the lightweight DistilBERT trigger can be re-fine-tuned for new domains without labeling the resulting prefix instances by hand.

We employ GPT-4o as a “linguistic analysis expert” to insert special trigger tokens “[T]” into the text. The insertion rules adhere to semantic completeness principles: tokens appear only at the boundaries of clauses, constraints, or complete intent expressions, specifically where no massive information influx is expected immediately, while strictly skipping disfluent hesitation markers. Following automated annotation, the authors manually

verified the dataset to ensure the validity of the semantic boundaries. Subsequently, we simulate the ASR transmission process at varying speaking rates, slicing the verified text containing “[T]” into streaming prefix sequences $x_{1:k}$. For each prefix, if its end crosses a “[T]” token, it is labeled as a positive instance $y = 1$; otherwise, it is negative $y = 0$. This process constructs a binary classification dataset $D_{trigger}$ optimized for streaming interaction.

3.2.2 Trigger Inference

We employ a lightweight DistilBERT as the backbone for the trigger. During inference, the trigger encodes the accumulated transcript from the ASR and outputs a confidence score. To prevent jitter, we implement a dual-filtering mechanism: (1) *threshold gating*, where triggering occurs only when the score exceeds τ ; and (2) *deduplication suppression*, where if the current transcript is identical to that at the last trigger (indicating a non-steady ASR state), triggering is suppressed. Together with the orchestrator-level concurrency control (Sec. 3.3.3), this design ensures that the system consumes reasoning resources only when the user expresses new valid information.

3.3 Dual-Role Stream Orchestrator

To reconcile the conflict between deep semantic understanding and sub-second responsiveness in

streaming interaction, we propose the *Dual-Role Stream Orchestrator*. Responding within a single inference pass often leads to frequent interruptions and semantic fragmentation (or “thought discontinuity”), even with trigger controls. Therefore, unlike traditional approaches that couple understanding and generation into a single serial process, we leverage the batch processing capabilities of the inference engine to concurrently schedule two complementary roles: the *Thinker* and the *Speaker*. Through an asynchronous communication mechanism, the orchestrator enables the Speaker to perform speculative solving before VAD finalizes the turn, while utilizing the global state maintained by the Thinker in the background to guide the logical direction of the generation.

3.3.1 Thinker: Stream Analysis & State Maintenance

The Thinker functions as a background analysis module dedicated to maintaining a structured understanding of user intent without directly participating in response generation. Addressing the noise inherent in streaming ASR, the Thinker first performs *Input Sanitization*, leveraging context to correct phonetic errors in the transcript (e.g., correcting “sign” to “sine”). Subsequently, it evaluates semantic completeness, extracts *Key Variables*, and formulates a dynamic *Plan*. Its compact JSON *State Snapshot* contains `corrected_text` (internal sanitized candidate, with raw transcript retained), `key_variables` (task-relevant entities/parameters), and `plan` (logical steps for subsequent reasoning).

3.3.2 Speaker: Speculative Execution

The Speaker acts as the foreground executor responsible for generating the final user-facing response. To overcome the latency bottleneck caused by serial waiting, the Speaker adopts an *Answer-First* strategy similar to PredGen (Li and Grover, 2025): reasoning is triggered immediately upon activation, without waiting for the user’s speech to fully cease. Its inference process follows a “Restate–Consult–Solve” paradigm: first, the Speaker briefly restates the core problem for implicit self-verification (*Restate & Verify*); next, it consults the decision plan in the shared memory (*Consult Plan*)—if the plan is clear, the Speaker follows its steps; otherwise, it constructs logic based on the raw context. Finally, it outputs the specific response. The full prompt templates for the Thinker and Speaker are

provided in Appendix E. Crucially, the Speaker’s output is not committed to TTS playback before the orchestrator confirms turn end (Sec. 3.3.3), so any rollback induced by subsequent ASR updates remains an internal event rather than an audible glitch.

3.3.3 Orchestrator: Asynchronous Coordination Mechanism

The orchestrator manages the asynchronous collaboration and interruption handling between the Thinker and the Speaker. Regarding communication, the system employs a *State Injection* mechanism: the plan and `key_variables` committed by the Thinker at the *previous* trigger of the same turn are injected into the *current* Speaker’s context as prior knowledge. Because both roles are launched concurrently from that committed snapshot, this design enables the Speaker to “stand on the shoulders of the Thinker,” benefiting from deep planning without incurring the latency penalty of waiting for the Thinker’s real-time inference. Regarding interruption handling, when a substantial semantic shift is detected in the user’s input stream, the orchestrator immediately terminates the current Speaker process but preserves the valid state generated by the Thinker. This state is then merged into the next context, achieving efficient “thought rollback” and state updates.

Concurrency control and state synchronization.

To ensure stability under high-noise ASR fluctuations, the orchestrator enforces two explicit policies. (1) *Latest-context-first*: whenever the Speaker is (re-)activated, it always receives the latest complete ASR transcript together with the most recent committed Thinker snapshot, so any stale partial hypothesis is replaced rather than blended; a Thinker result that completes after a newer snapshot has been committed is likewise discarded. (2) *Preemptive termination*: if the trigger fires again while a Speaker process is still running—typically because of a user self-correction or an ASR revision—the orchestrator immediately terminates the in-flight Speaker, retains the valid Thinker state, and starts a fresh Thinker–Speaker cycle on the updated transcript. Combined with the input-side filtering in Sec. 3.2.2, these two policies are designed to ensure that, even under rapid ASR jitter, the output reaching TTS is computed from the most recent stable context, thereby avoiding audible rollbacks in our current evaluation. We provide the corresponding

pseudo-code in Appendix D.

4 Dataset: A Pause-and-Repair Benchmark

To bridge the gap between static text benchmarks (e.g., GSM8K (Cobbe et al., 2021), MMLU (Hendrycks et al., 2021)) and real-world spoken interaction, and to verify the robustness of our framework in noisy scenarios, we constructed a synthetic streaming speech dataset that targets semantic-/logical-level streaming behavior, where fragmented updates from disfluencies and self-corrections can disrupt streaming reasoning even after correct ASR transcription. We discuss the acoustic gap between synthetic disfluencies and fully spontaneous speech in the Limitations section.

Scenario-based contextualization. We use GSM8K and MMLU-Pro (Wang et al., 2024b) as source data and convert abstract exam questions into concrete, first-person help-seeking scenarios (e.g., homework tutoring). This keeps each audio input semantically self-contained and requires end-to-end reasoning from the speech stream.

Hybrid spontaneous simulation. To stress-test disfluent, revision-heavy inputs, we use a Hybrid Linguistic Perturbation pipeline (Appendix L) to simulate text-/semantic-level “thinking-while-speaking” behavior.

Instead of simple noise injection, this dataset simultaneously integrates two complex speech characteristics:

- *Filler word injection.* We introduce a hesitation mechanism that dynamically controls the density of fillers (e.g., “um,” “uh”) and discourse markers based on text length, avoiding semantic collapse caused by over-padding.
- *Self-correction patterns.* Addressing dynamic intent shifts, we design four self-correction paradigms, including entity correction and intent reset. We use instruction tuning to ensure that these correction behaviors align with human cognitive habits.

Finally, all text variants are converted into waveform data via the open-source TTS component CosyVoice 2 (Du et al., 2024), with explicit style instructions (e.g., “note pauses, control speaking rate”) so that inserted tokens are encouraged to be

rendered with pauses and speaking-rate variation rather than as plain lexical pronunciations.

5 Experiments

In this section, we evaluate LTS-VoiceAgent in terms of reasoning quality, response latency, and inference efficiency.

5.1 Experimental Setup

5.1.1 Datasets

We evaluate LTS-VoiceAgent on three representative public benchmarks and our synthetic dataset. In contrast to closely related setups that approximate streaming ASR by feeding fixed text chunks or injecting heuristic noise to mimic recognition errors and revisions, we adopt an audio-only evaluation pipeline that exposes the system to realistic ASR behavior, including misrecognitions, non-uniform speaking rates, and related artifacts. Specifically, we use VERA (AIME and GPQA-Diamond) for challenging math and science reasoning (Lin et al., 2025), Spoken-MQA for multi-step spoken math QA (Wei et al., 2025a), BigBenchAudio for diverse open-domain spoken tasks (Suzgun et al., 2022), and our Pause-and-Repair Benchmark (Sec. 4) featuring extensive hesitations and self-corrections.

5.1.2 Baselines

We compare LTS-VoiceAgent against three streaming paradigms: *Serial Reasoning* (No-think / Think), which runs the LLM only after ASR completes and provides an accuracy upper bound; *PredGen*, which triggers chunk-based speculative generation on fixed text segments (Li and Grover, 2025); and *LTS-VAD*, which augments PredGen with voice activity detection so that reasoning is triggered mainly during silences. In addition, we compare against an end-to-end speech model, Qwen2.5-Omni (Xu et al., 2025a), to quantify how cascaded pipelines differ from end-to-end modeling in the accuracy–latency trade-off.

5.1.3 Implementation & Metrics

We control for ASR variability by using a unified internal streaming ASR API across all experiments, and we fix the LLM backbone to Qwen3-8B. All experiments are executed on a single NVIDIA A100 GPU, and all baselines are evaluated under the same hardware and ASR setup for fairness. We implement LTS-VoiceAgent with an asynchronous dual-stream inference stack based on vLLM (Kwon

Method	VERA			Spoken-MQA			BigBench			Our Bench		
	Acc(%)	Lat	TTFS	Acc(%)	Lat	TTFS	Acc(%)	Lat	TTFS	Acc(%)	Lat	TTFS
Serial (No-think)	4.36	1205	4782	75.53	904	3557	54.95	990	3166	58.45	663	1764
Serial (Think, upper bound)	18.54	151k	212k	83.95	37k	43k	73.77	76k	84k	76.34	51k	61k
PredGen (No-think)	4.34	247	298	67.87	230	281	54.5	213	264	55.84	251	303
PredGen (Think)	8.69	47k	106k	83.52	23k	25k	71.6	33k	46k	70.51	43k	48k
LTS-VAD (No-think)	3.98	244	380	75.14	213	266	55.2	206	258	56.41	244	294
LTS-VAD (Think)	8.69	47k	103k	83.52	24k	26k	71.5	33k	45k	72.36	29k	38k
LTS-VoiceAgent (Ours)	6.88	230	415	78.52	207	332	71.9	209	338	62.57	236	336
Qwen2.5-Omni (E2E)	1.44	339.9	985.95	59.89	139.44	634.68	43.1	195.13	332.92	50.25	245.17	449.9

Table 1: Overall performance (Accuracy, Latency, and TTFS) across three public benchmarks and our proposed benchmark. Latency and TTFS are measured in milliseconds (ms). Serial (Think) provides an upper bound with full context; bold marks the best among real-time methods. Large values are rounded to 'k' for compactness.

et al., 2023). The streaming ASR transcript is processed at a 200 ms granularity (16 kHz, mono PCM; 3,200 samples per chunk). For the Dynamic Semantic Trigger, we use a lightweight DistilBERT classifier (Sanh et al., 2019) with maximum input length 512 and threshold gating at $\tau = 0.65$, together with deduplication suppression that only prevents repeated triggers on identical transcripts under ASR jitter.

Trigger implementation details. We instantiated the training set with 1,000 samples each from GSM8K and MMLU-Pro. The streaming decomposition (Sec. 3.2.1) expanded these texts into cumulative prefixes, creating a large-scale sample pool. After applying 1:1 negative downsampling to balance the dataset, we obtained 100k samples to fine-tune the DistilBERT classifier (learning rate $5e-5$). We evaluate the trigger’s effectiveness implicitly through end-to-end system metrics rather than standalone classification accuracy, acknowledging the non-unique nature of semantic boundaries in spoken language.

Evaluation metrics. We evaluate *quality* via task accuracy, *latency* via streaming latency L (time from turn end to the first emitted text token) and Time-to-First-Sentence (TTFS, time from turn end to the first complete sentence at which TTS can begin fluid playback), and *efficiency* via the Number of Forward-pass Evaluations (NFE, the number of Speaker invocations per turn) and the Number of Interruption Times (NIT, how many of those invocations are preemptively terminated by a subsequent semantic shift). The *interruption rate* is $IR = NIT/NFE$; lower values indicate more precise trigger timing. Formal definitions are provided in Appendix C. Unless otherwise noted, both the

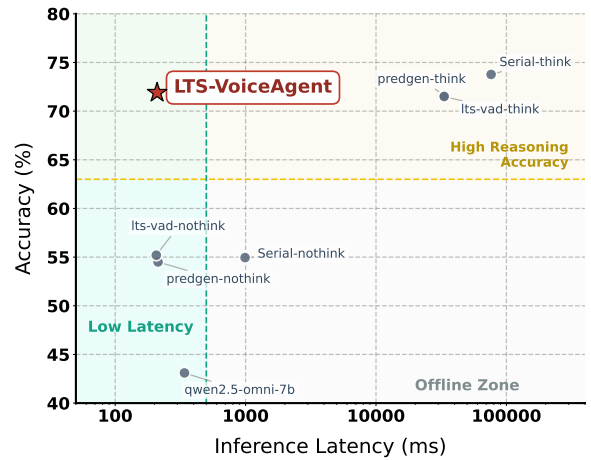


Figure 4: Performance (Accuracy) vs. Latency trade-off analysis.

Speaker and Thinker streams use greedy decoding with temperature 0 and a maximum of 4,096 new tokens.

5.2 Experimental Results

Table 1 presents comprehensive results on the three public datasets, covering task accuracy, streaming latency, and TTFS. These metrics capture complementary aspects of spoken interaction: accuracy reflects whether the final response is correct, while latency and TTFS quantify how quickly the system can begin delivering usable output after turn end. To better visualize this quality–latency trade-off, we plot accuracy against latency for BigBenchAudio by converting the corresponding results into a 2-D scatter plot, as shown in Figure 4.

LTS-VoiceAgent improves the quality–latency trade-off. We compare methods under an interactive-latency budget rather than collapsing quality and latency into a single weighted score.

Method	VERA			Spoken-MQA			BigBench			Our Bench		
	NFE↓	NIT↓	Rate(%)	NFE↓	NIT↓	Rate(%)	NFE↓	NIT↓	Rate(%)	NFE↓	NIT↓	Rate(%)
PredGen (No-think)	176.83	175.11	99.03	59.39	58.14	97.90	80.73	79.38	98.33	172.46	156.06	90.05
PredGen (Think)	176.83	175.83	99.43	59.39	58.39	98.31	80.73	79.73	98.76	172.46	164.34	95.29
LTS-VAD (No-think)	10.94	8.04	73.49	3.42	1.82	53.21	6.28	4.68	74.52	8.23	3.42	41.53
LTS-VAD (Think)	10.94	9.80	89.58	3.42	2.39	69.88	6.29	5.26	83.62	8.23	6.22	75.49
LTS-VoiceAgent (Ours)	2.37	0.13	5.46	1.96	0.16	8.16	2.05	0.20	9.76	2.02	0.19	9.4

Table 2: Efficiency analysis comparing Number of Forward-pass Evaluations (NFE), Number of Interruption Times (NIT), and Interruption Rate across all datasets. Lower values (↓) indicate better efficiency and fewer unnecessary interruptions. **Bold** indicates the best performance.

Table 1 shows that LTS-VoiceAgent matches streaming-level latency while achieving markedly higher accuracy than chunk- and VAD-based baselines across all benchmarks. Its TTFS remains sub-second, with only a small increase over the most aggressive chunking variants. *Serial (Think)* yields the highest accuracy with full context, but incurs TTFS in the tens to hundreds of seconds, making it impractical for interactive agents. LTS-VoiceAgent narrows this quality gap while retaining real-time responsiveness in practice. We provide contextual comparisons with commercial voice LLMs in Appendix F, and qualitative case analyses of streaming revisions and ASR artifacts in Appendix K.

Efficiency. Table 2 reports NFE, NIT, and the interruption rate. PredGen is highly wasteful: it performs frequent speculative evaluations and almost all of them are interrupted, indicating pervasive rollback/recompute. LTS-VAD reduces the evaluation count but still incurs substantial interruptions, with rates ranging from 41% to 90% depending on the dataset and mode. In contrast, LTS-VoiceAgent consistently achieves the lowest overhead and keeps interruptions near zero, resulting in single-digit interruption rates across all benchmarks. Trigger inference is lightweight, with an average runtime of about 5 ms. We further report audio-side naturalness and audible rollback checks in Appendix G, and subjective overhearer ratings in Appendix H.

Ablation study. We ablate the *Dynamic Semantic Trigger* and the *Dual-Role Stream Orchestrator* to quantify their contributions. Table 3 summarizes the results averaged over all benchmarks, while Figure 5 visualizes the trade-offs across quality, responsiveness, reaction speed, computational efficiency, and interaction stability. Detailed per-benchmark results are reported in Appendix I. Re-

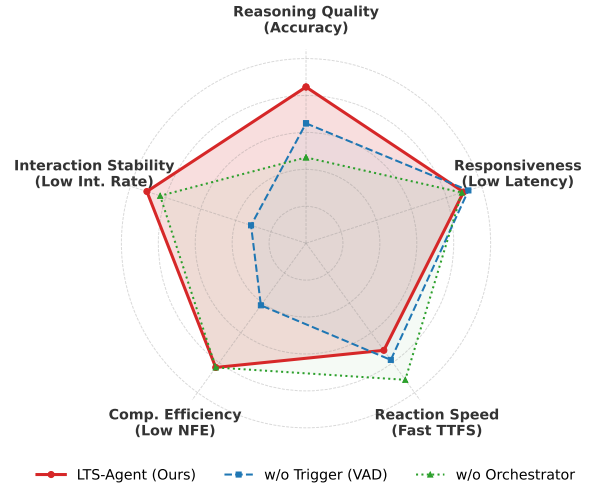


Figure 5: Ablation study on the LTS-VoiceAgent.

Metric	Ours	w/o Trigger	w/o Orch.
Acc (%)↑	55.37	52.30	49.37
Latency (ms)↓	238.38	210.75	225.58
TTFS (ms)↓	339.59	332.22	282.40
NFE↓	2.10	7.23	2.10
Int.Rate (%)↓	8.20	60.12	14.81

Table 3: Ablation results averaged over all benchmarks. Int.Rate denotes the interruption rate (NIT/NFE). Detailed per-benchmark results are in Appendix I.

moving the trigger (*w/o Trigger*) reduces accuracy and substantially increases overhead, with NFE rising from 2.10 to 7.23 and the interruption rate from 8.20% to 60.12%. Removing the orchestrator (*w/o Orchestrator*) further degrades accuracy and increases the interruption rate, despite similar NFE, indicating that orchestration is important for maintaining stable streaming reasoning. We further evaluate LTS-VoiceAgent with a larger Qwen3-32B backbone in Appendix J, and observe consistent gains in reasoning quality while preserving sub-second streaming responsiveness.

6 Conclusion

We presented LTS-VoiceAgent, which separates *when to think* from *how to reason* via dual-role orchestration. Across four benchmarks, it improves the quality–latency–efficiency trade-off over serial and streaming baselines.

Ethics Statement

This work introduces a framework for low-latency streaming voice interaction. While our primary goal is to enhance user experience in legitimate assistant scenarios, we acknowledge that improved responsiveness could potentially be misused to create more convincing automated systems for social engineering or unsolicited communication. Additionally, the "listening-while-thinking" mechanism implies continuous processing of audio streams; real-world deployments must strictly adhere to privacy protocols (e.g., local processing or explicit consent) to prevent unintended analysis of background conversations. Finally, as our framework relies on general-purpose LLMs (e.g., Qwen) for reasoning, it inherits their potential limitations regarding hallucinations and biases, necessitating safety guardrails in downstream applications.

Limitations

Our evaluation covers a limited range of backbones, languages, and acoustic conditions. Although we adopt an audio-only protocol and introduce Pause-and-Repair to stress-test hesitations and self-corrections, it should be read as a semantic-/logical-level streaming stress test rather than a full model of spontaneous speech. The Dynamic Semantic Trigger is trained mainly on math/science-style prefixes, so extreme domain shifts (e.g., short commands or open-domain chitchat) may require in-domain re-tuning. We also do not include large-scale interaction-participant studies; our latency improvements target avoidable system-induced waiting under complex spoken requests, not a universal claim that faster responses are always preferable.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Chuangtao Chen, Grace Li Zhang, Xunzhao Yin, Cheng Zhuo, Ulf Schlichtmann, and Bing Li. 2024a. Livemind: Low-latency large language models with simultaneous inference. *arXiv preprint arXiv:2406.14319*.

Yiming Chen, Xianghu Yue, Chen Zhang, Xiaoxue Gao, Robby T Tan, and Haizhou Li. 2024b. Voicebench: Benchmarking llm-based voice assistants. *arXiv preprint arXiv:2410.17196*.

Jeffrey Cheng and Benjamin Van Durme. 2024. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 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*.

David DeVault, Kenji Sagae, and David Traum. 2009. Can i finish? learning when to respond to incremental interpretation results in interactive dialogue. In *Proceedings of the SIGDIAL 2009 Conference*, pages 11–20.

Zhihao Du, Yuxuan Wang, Qian Chen, Xian Shi, Xiang Lv, Tianyu Zhao, Zhifu Gao, Yexin Yang, Changfeng Gao, Hui Wang, et al. 2024. Cosyvoice 2: Scalable streaming speech synthesis with large language models. *arXiv preprint arXiv:2412.10117*.

Vignesh Ethiraj, Ashwath David, Sidhanth Menon, and Divya Vijay. 2025. Toward low-latency end-to-end voice agents for telecommunications using streaming asr, quantized llms, and real-time tts. *arXiv preprint arXiv:2508.04721*.

Qingkai Fang, Shoutao Guo, Yan Zhou, Zhengrui Ma, Shaolei Zhang, and Yang Feng. 2024. Llama-omni: Seamless speech interaction with large language models. *arXiv preprint arXiv:2409.06666*.

Qingkai Fang, Yan Zhou, Shoutao Guo, Shaolei Zhang, and Yang Feng. 2025. Llama-omni2: Llm-based real-time spoken chatbot with autoregressive streaming speech synthesis. *arXiv preprint arXiv:2505.02625*.

Cecilia E. Ford and Sandra A. Thompson. 1996. *Interactional Units in Conversation: Syntactic, Intonational, and Pragmatic Resources for the Management of Turns*, page 134–184. Cambridge University Press.

- M Soledad López Gambino, Sina Zarriß, and David Schlangen. 2017. Beyond on-hold messages: Conversational time-buying in task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 241–246.
- Arushi Goel, Sreyan Ghosh, Jaehyeon Kim, Sonal Kumar, Zhifeng Kong, Sang-gil Lee, Chao-Han Huck Yang, Ramani Duraiswami, Dinesh Manocha, Rafael Valle, et al. 2025. Audio flamingo 3: Advancing audio intelligence with fully open large audio language models. *arXiv preprint arXiv:2507.08128*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
- Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*.
- Rongjie Huang, Mingze Li, Dongchao Yang, Jia-tong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jiawei Huang, Jinglin Liu, et al. 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.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.
- 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*.
- Patrick Kahardipraja, Brielen Madureira, and David Schlangen. 2023. Tapir: Learning adaptive revision for incremental natural language understanding with a two-pass model. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4173–4197.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.
- Stephen C Levinson. 2016. Turn-taking in human communication—origins and implications for language processing. *Trends in cognitive sciences*, 20(1):6–14.
- Shufan Li and Aditya Grover. 2025. Predgen: Accelerated inference of large language models through input-time speculation for real-time speech interaction. *arXiv preprint arXiv:2506.15556*.
- Yinghao Aaron Li, Xilin Jiang, Jordan Darefsky, Ge Zhu, and Nima Mesgarani. 2024. Style-talker: Finetuning audio language model and style-based text-to-speech model for fast spoken dialogue generation. *arXiv preprint arXiv:2408.11849*.
- Tatiana Likhomanenko, Luke Carlson, Richard He Bai, Zijin Gu, Han Tran, Zakaria Aldeneh, Yizhe Zhang, Ruixiang Zhang, Huangjie Zheng, and Navdeep Jaitly. 2025. Chipchat: Low-latency cascaded conversational agent in mlx. *arXiv preprint arXiv:2509.00078*.
- Yueqian Lin, Zhengmian Hu, Qinsi Wang, Yudong Liu, Hengfan Zhang, Jayakumar Subramanian, Nikos Vlassis, Hai Helen Li, and Yiran Chen. 2025. Voice evaluation of reasoning ability: Diagnosing the modality-induced performance gap. *arXiv preprint arXiv:2509.26542*.
- Zuwei Long, Yunhang Shen, Chaoyou Fu, Heting Gao, Lijiang Li, Peixian Chen, Mengdan Zhang, Hang Shao, Jian Li, Jinlong Peng, et al. 2025. Vita-audio: Fast interleaved cross-modal token generation for efficient large speech-language model. *arXiv preprint arXiv:2505.03739*.
- Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, et al. 2019. Stacl: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036.
- Maïke Paetzel, Ramesh Manuvinaurike, and David DeVault. 2015. “so, which one is it?” the effect of alternative incremental architectures in a high-performance game-playing agent. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 77–86.
- Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR.
- Yi Ren, Jinglin Liu, Xu Tan, Chen Zhang, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2020. Simulspeech: End-to-end simultaneous speech to text translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3787–3796.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.
- David Schlangen and Gabriel Skantze. 2011. A general, abstract model of incremental dialogue processing. *Dialogue & Discourse*, 2:83–111.

- Yemin Shi, Yu Shu, Siwei Dong, Guangyi Liu, Jaward Sesay, Jingwen Li, and Zhiting Hu. 2025. Voila: Voice-language foundation models for real-time autonomous interaction and voice role-play. *arXiv preprint arXiv:2505.02707*.
- Sambal Shikhar, Mohammed Irfan Kurpath, Sahal Shaji Mullappilly, Jean Lahoud, Fahad Shahbaz Khan, Rao Muhammad Anwer, Salman Khan, and Hisham Cholakkal. 2025. Llmvox: Autoregressive streaming text-to-speech model for any llm. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20481–20493.
- Gabriel Skantze. 2021. Turn-taking in conversational systems and human-robot interaction: a review. *Computer Speech & Language*, 67:101178.
- Tanya Stivers, Nicholas J Enfield, Penelope Brown, Christina Englert, Makoto Hayashi, Trine Heine-mann, Gertie Hoymann, Federico Rossano, Jan Peter De Ruiter, Kyung-Eun Yoon, et al. 2009. Universals and cultural variation in turn-taking in conversation. *Proceedings of the National Academy of Sciences*, 106(26):10587–10592.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.
- Zakary L. Tormala, Joshua J. Clarkson, and Marlone D. Henderson. 2011. Does fast or slow evaluation foster greater certainty? *Personality and Social Psychology Bulletin*, 37(3):422–434.
- Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, et al. 2023. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111*.
- Xiong Wang, Yangze Li, Chaoyou Fu, Yunhang Shen, Lei Xie, Ke Li, Xing Sun, and Long Ma. 2024a. Freeze-omni: A smart and low latency speech-to-speech dialogue model with frozen llm. *arXiv preprint arXiv:2411.00774*.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, et al. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.
- Chengwei Wei, Bin Wang, Jung-jae Kim, and Nancy F Chen. 2025a. Towards spoken mathematical reasoning: Benchmarking speech-based models over multi-faceted math problems. *arXiv preprint arXiv:2505.15000*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Linye Wei, Shuzhang Zhong, Songqiang Xu, Runsheng Wang, Ru Huang, and Meng Li. 2025b. Specasr: Accelerating llm-based automatic speech recognition via speculative decoding. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*, pages 1–7. IEEE.
- Tony Woo, Sehun Lee, Kang-wook Kim, and Gunhee Kim. 2025. Think, verbalize, then speak: Bridging complex thoughts and comprehensible speech. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 14373–14390.
- Jin Xu, Zhifang Guo, Jinzheng He, Hangrui Hu, Ting He, Shuai Bai, Keqin Chen, Jialin Wang, Yang Fan, Kai Dang, et al. 2025a. Qwen2. 5-omni technical report. *arXiv preprint arXiv:2503.20215*.
- Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025b. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*.
- Xin Zhang, Xiang Lyu, Zhihao Du, Qian Chen, Dong Zhang, Hangrui Hu, Chaohong Tan, Tianyu Zhao, Yuxuan Wang, Bin Zhang, et al. 2024. Intrinsicvoice: Empowering llms with intrinsic real-time voice interaction abilities. *arXiv preprint arXiv:2410.08035*.

A Usage of LLM

Large language models (ChatGPT) were used in two ways. First, they were used to improve the clarity and fluency of English writing. Second, they were used as a data-generation and annotation component in our synthetic data pipelines: (i) to insert special trigger tokens (e.g., “[T]”) for synthesizing training data for the Dynamic Semantic Trigger, and (ii) to rewrite/augment text benchmarks into scenario-based, disfluent spoken-style inputs (e.g., filler insertion and self-correction patterns) for constructing the Pause-and-Repair benchmark, following fixed prompts described in the paper/appendix. These models were not used to design the core method, choose evaluation metrics, or perform statistical analysis. All experimental results, comparisons, and interpretations were conducted by the authors, who take full responsibility for the content.

B Extended Related Work

Incremental dialogue processing. A long-standing line of work in dialogue systems studies how to consume input incrementally and decide *when* to respond. The Incremental Unit (IU) framework (Schlangen and Skantze, 2011) provides general vocabulary for incremental modules, and pragmatic completion (Ford and Thompson, 1996) explains how listeners use syntactic, intonational, and pragmatic cues to project turn ends. DeVault et al. (2009) and Paetzel et al. (2015) learn classifiers that predict when enough has been said to commit a response, while more recent work such as TAPIR (Kahardipraja et al., 2023) performs incremental NLU with revision-aware transformer decoders. Our Dynamic Semantic Trigger inherits the same goal of *learned semantic completeness detection*, but is deployed inside a contemporary ASR–LLM cascade where each LLM invocation is expensive, and where premature triggers are penalized not only by acoustic re-segmentation but by wasted long-form generation. Conversation-analytic findings on universal turn-taking timing (Stivers et al., 2009; Levinson, 2016) and devices that decouple acknowledgement from content delivery (Gambino et al., 2017) further inform our scope: we do not seek to override these regularities, but to reduce the system-induced waiting that is orthogonal to them.

Simultaneous translation. Simultaneous translation also studies a latency–quality trade-off, with

wait- k and prefix-to-prefix decoding as canonical solutions (Ren et al., 2020; Ma et al., 2019). Our setting differs in two ways. First, voice agents typically need a single committed answer rather than a monotonically expanding output, and the “correct” commit point is more semantically determined than translation chunk boundaries. Second, our system must perform deeper reasoning (planning, key-variable tracking, and self-correction) over partial input, rather than incrementally translating what has already been said. We therefore borrow the latency–quality framing but add explicit state maintenance and dual-role scheduling.

C Formal Definitions of Evaluation Metrics

We provide the formal definitions corresponding to the metrics summarized in Sec. 5.1.3. Let $t_{\text{user_end}}$ denote the time at which the user finishes the current utterance (i.e., turn end is confirmed by VAD), $t_{\text{first_token}}$ the time of the first text token emitted by the Speaker, and $t_{\text{first_sentence}}$ the time of the first complete sentence (detected via punctuation), which is the earliest moment TTS can begin fluid playback. Streaming latency L and Time-to-First-Sentence (TTFS) are defined as

$$\begin{aligned} L &= t_{\text{first_token}} - t_{\text{user_end}}, \\ \text{TTFS} &= t_{\text{first_sentence}} - t_{\text{user_end}}. \end{aligned} \quad (1)$$

Within a single conversational turn, NFE denotes the discrete number of Speaker invocations triggered by the orchestrator, quantifying LLM compute overhead, and NIT counts how many of those invocations are preemptively terminated due to a subsequent semantic shift. The interruption rate is then

$$\text{IR} = \frac{\text{NIT}}{\text{NFE}}, \quad (2)$$

where lower values indicate fewer mid-reasoning interruptions and therefore more precise trigger timing.

D Dual-Role Stream Orchestrator: Pseudo-code

Algorithm 1 formalizes the asynchronous coordination mechanism described in Sec. 3.3.3. The orchestrator drives a single Thinker–Speaker pair, launched concurrently at each trigger, under a *latest-context-first* and *preemptive-termination* discipline: every Speaker activation receives the latest

Algorithm 1 Dual-Role Stream Orchestrator

Require: Streaming audio chunks $\mathcal{C} = \{c_1, c_2, \dots\}$

Require: Thinker T , Speaker S , Semantic Trigger TR

Ensure: Final response y committed to TTS

```

1:  $H \leftarrow \emptyset$  ▷ background state
2:  $P_s \leftarrow \text{null}$  ▷ active speaker process
3: for each chunk  $c_i \in \mathcal{C}$  do
4:    $\text{txt}_i \leftarrow \text{ASR}(c_i)$ 
5:   if TR.fires( $\text{txt}_i$ ) then
6:     if  $P_s$  is running then
7:        $P_s.\text{terminate}()$  ▷ preemptive
       termination
8:     end if
9:      $\bar{H} \leftarrow H$  ▷ freeze committed state
10:     $P_s \leftarrow S.\text{start\_speculation}(\text{txt}_i, \bar{H})$ 
    ▷ non-blocking
11:     $H \leftarrow T.\text{update}(\text{txt}_i, \bar{H})$  ▷
    concurrent; stale updates dropped
12:  end if
13:  if end_of_utterance( $c_i$ ) then
14:     $y \leftarrow P_s.\text{latest\_output}()$ 
15:    return  $y$ 
16:  end if
17: end for

```

complete ASR transcript together with the most recent committed Thinker snapshot, and any in-flight Speaker is terminated as soon as the trigger fires on a new semantic prefix.

E Full Prompts for Thinker and Speaker

We provide the exact prompt templates used by the background Thinker and the foreground Speaker. Bracketed fields (e.g., [Prior State]) are populated by the orchestrator at runtime. The prompts encode three risk-control rules for input sanitization—*contextual anchoring*, *conservative editing*, and *clarification fallback*—to mitigate the risk of hallucinated rewrites altering user intent.

F Positioning Against Commercial Voice LLMs

For broader context, Table 4 situates LTS-VoiceAgent (Qwen3-8B backbone) alongside leading commercial voice LLMs on BigBench Audio. Numbers for the commercial systems are taken from public leaderboard reports of speech-to-speech models. Latency definitions and deploy-

ment conditions for these closed systems are not fully controllable or reproducible to us; the table therefore serves as *contextual positioning* rather than a strict head-to-head comparison.

Model	Acc. (%)	Latency (s)
Nova 2.0 Sonic	87.0	1.39
GPT Realtime	83.0	0.98
Gemini 2.5 Flash	71.0	0.63
LTS-VoiceAgent (Qwen3-8B)	71.9	0.21

Table 4: Contextual positioning on BigBench Audio. LTS-VoiceAgent matches Gemini 2.5 Flash in accuracy at roughly one-third of its latency under our measurement protocol, while top-tier commercial systems achieve higher accuracy peaks at substantially larger scale.

This positioning highlights that a lightweight 8B cascaded framework can already deliver a competitive accuracy–latency profile relative to far larger closed-source systems, particularly on the latency axis that our orchestration design directly targets.

G Audio-Side Quality and Audible Rollbacks

Because LTS-VoiceAgent is cascaded with a TTS engine, an audio-side check is needed to verify that the latency and efficiency gains do not introduce perceptible artifacts. We report two complementary measurements.

TTS naturalness (NMOS). We adopt the ITU-T P.808-style NMOS protocol. Using CosyVoice 2 as our default TTS, its standalone NMOS is 3.96. When we route the Speaker’s committed text through the same TTS in the end-to-end LTS-VoiceAgent pipeline, the measured NMOS is 3.94, essentially matching the CosyVoice baseline. This indicates that inserting LTS-VoiceAgent between the user and the TTS does not degrade synthesis quality.

Audible rollbacks. Speculative streaming methods are sometimes criticized for producing audible cut-offs when an in-flight response is invalidated by a late ASR revision. By construction, LTS-VoiceAgent commits Speaker output to TTS only after the orchestrator confirms turn end (Sec. 3.3.3); any preemptive termination happens inside the listening phase and is never rendered to audio. To verify this, we tracked *audible rollback events*—i.e., cases where the user would hear either a truncated

System Prompt: Thinker (Stream Analysis & Dynamic Planning)

Role: You are a background reasoning module that analyzes the live ASR stream and maintains a structured understanding of the user's intent. You do NOT solve the problem; your output is for internal state tracking only.

Inputs:

1. [Prior State]: the most recent committed JSON snapshot.
2. [New Transcript]: the latest cumulative ASR transcript.

Analysis Steps:

1. *Input Sanitization.* Apply only minimal, high-confidence local fixes for obvious ASR errors (homophones, phonetic mismatches). Anchor every fix in the surrounding context (plan / key_variables). If the correction is ambiguous, KEEP the original term and never silently rewrite user intent.
2. *Completeness Check.* Decide whether the user's request is semantically complete enough to commit a final answer.
3. *Strategic Planning.* Outline the next logical steps that the Speaker should follow.

Output Format (JSON only):

```
{"corrected_text": "...", "is_complete_sentence": true/false, "key_variables": {...}, "plan": "..."}

```

System Prompt: Speaker (Conversational Problem Solving)

Role: You are the foreground response generator. Perform the necessary reasoning to answer the user's request in a natural, spoken tone.

Inputs:

1. [Manager's Plan]: the latest plan injected from the Thinker.
2. [Raw Transcript]: the latest cumulative ASR transcript.

Execution Steps:

1. *Restate & Verify.* Implicitly verify the problem (fixing residual ASR errors) and consult the [Manager's Plan].
2. *Consult Plan.* If the plan is clear, follow its steps; otherwise construct logic from the raw transcript.
3. *Solve.* Explain the solution logic step-by-step in a conversational flow.

Output Format:

A natural spoken response that concludes with the final answer inside \boxed{...}.

Figure 6: System prompts used by the Thinker (top) and the Speaker (bottom).

utterance or a jarring restart—across all evaluations. The observed count is 0, consistent with the commit-after-turn-end design.

H Subjective Overhearer Evaluation

To complement automated metrics with subjective interaction-quality evidence, we conducted a small-scale *overhearer* study. We emphasize that this is an overhearer (third-party) rating rather than a participant-based study, and we treat it as a sanity check for whether LTS-VoiceAgent's latency gains come at the cost of perceived interaction naturalness or semantic consistency—not as a comprehensive HCI evaluation.

Protocol. We recruited 20 native English speakers (ages 18–35, no self-reported hearing impairments). We randomly sampled 40 interaction clips

without manual correction from VERA, Spoken-MQA, BigBench Audio, and our Pause-and-Repair benchmark. The same CosyVoice 2 TTS voice was used across all methods to remove voice-identity confounds. Participants were informed that the task involved rating anonymized third-party interaction clips for research purposes and that only aggregate results would be reported; no participant speech, personal conversations, or sensitive personal data were collected.

Compared methods and measures. Three methods were compared: the Serial-reasoning baseline, PredGen, and LTS-VoiceAgent. Raters scored each clip on a 5-point Likert scale (1 = very poor, 5 = excellent) along three dimensions:

- *Perceived latency:* the subjective “waiting feeling” between the end of the user's utter-

ance and the onset of the system response.

- *Interaction naturalness*: whether response pacing and phrasing align with conversational norms (absence of abrupt cut-offs or jarring responses).
- *Semantic consistency*: semantic alignment between the response and user input (no off-topic replies or logical discontinuities).

Dimension	Serial	PredGen	LTS-VA
Perceived latency	2.1±0.4	4.3±0.3	4.5±0.2
Interaction naturalness	4.4±0.3	4.2±0.4	4.3±0.3
Semantic consistency	4.5±0.2	3.8±0.3	4.4±0.3

Table 5: Subjective overhearer ratings ($N=20$; mean $\pm$ std). LTS-VA (ours) substantially improves perceived latency over the serial baseline while maintaining semantic consistency close to it, and is more stable than PredGen under revisions and incremental-input fluctuations.

Results. These ratings are consistent with our automated results: LTS-VoiceAgent’s latency gains do not appear to come at the cost of naturalness or semantic alignment, whereas PredGen’s speculative generation, although fast, produces noticeably weaker semantic consistency under incremental revisions. We acknowledge the limitations of overhearer studies relative to participant-based evaluation; a full HCI study with interaction participants and standardized instruments (e.g., SASSI, SUS) is left for future work.

I Ablation Studies

We conduct ablation studies to quantify the contribution of two core components in LTS-VoiceAgent: the Dynamic Semantic Trigger and the Dual-Role Stream Orchestrator. We evaluate two variants, one removing the trigger and falling back to mechanical triggering, and the other removing orchestration and running the pipeline without dual-role coordination. We report accuracy, latency, TTFS, and efficiency metrics on four benchmarks.

Removing the semantic trigger consistently degrades accuracy across benchmarks, with a pronounced drop on BigBenchAudio and VERA. Although mechanical triggering may yield slightly smaller latency or TTFS on some datasets, it substantially increases redundant computation and instability. Table 2 shows that the trigger removal leads to much higher NFE and a markedly higher

interruption rate, indicating frequent invalid reasoning attempts that are later interrupted by streaming transcript updates.

Disabling orchestration results in a larger loss of reasoning quality than removing the trigger on most benchmarks, while often reducing TTFS. This suggests that orchestration primarily contributes to preserving reasoning quality under evolving partial input rather than simply accelerating first responses. In addition, the interruption rate increases noticeably without orchestration, reflecting reduced interaction stability when background state maintenance and foreground response generation are not properly coordinated.

Overall, the full LTS-VoiceAgent achieves the strongest balance across accuracy, responsiveness, and efficiency. The semantic trigger is the main driver of computational efficiency and interruption robustness, while the orchestrator is critical for sustaining reasoning quality and stable streaming behavior.

J Robustness to Backbone Scaling

To complement the main results with Qwen3-8B, we further evaluate LTS-VoiceAgent with a larger Qwen3-32B backbone. We reuse the same ASR/TTS stack and experimental protocol as in Section 5, and report accuracy, latency, TTFS, and efficiency metrics to assess whether LTS-VoiceAgent remains effective when scaling up the underlying model.

Qwen3-32B improves streaming reasoning quality. Table 8 shows that upgrading the backbone from Qwen3-8B to Qwen3-32B consistently increases accuracy across all benchmarks. The gains are particularly clear on Spoken-MQA and our Pause-and-Repair benchmark, indicating that a stronger backbone translates into better end-to-end reasoning even under streaming constraints.

LTS-VoiceAgent preserves responsiveness compared with serial cascades. Even with the larger 32B model, LTS-VoiceAgent maintains sub-second TTFS, while the serial cascade baseline with Qwen3-32B exhibits multi-second TTFS across benchmarks. This confirms that the proposed streaming design remains crucial for interactivity as model size increases.

Efficiency remains stable, with a modest increase in interruption rate. As shown in Table 9, LTS-VoiceAgent with Qwen3-32B keeps

Method	VERA			Spoken-MQA			BigBench			Our Bench		
	Acc(%)	Latency	TTFS	Acc(%)	Latency	TTFS	Acc(%)	Latency	TTFS	Acc(%)	Latency	TTFS
LTS-VoiceAgent w/o Trigger	5.79	224.36	349.62	76.91	183.22	298.71	66.1	195.51	309.37	60.41	239.92	371.16
LTS-VoiceAgent w/o orchestrator	3.26	236.99	288.69	67.96	219.32	272.64	64.5	211.38	263.65	61.77	234.64	304.60
LTS-VoiceAgent (Ours)	6.88	230.74	415.49	78.52	207.46	332.48	71.9	209.41	338.5	62.57	235.97	335.95

Table 6: Ablation results on accuracy, latency, and TTFS across three public benchmarks and our Pause-and-Repair benchmark. Latency and TTFS are measured in milliseconds (ms).

Method	VERA			Spoken-MQA			BigBench			Our Bench		
	NFE↓	NIT↓	Rate(%)	NFE↓	NIT↓	Rate(%)	NFE↓	NIT↓	Rate(%)	NFE↓	NIT↓	Rate(%)
LTS-VoiceAgent w/o Trigger	10.94	7.64	69.64	3.42	1.94	56.73	6.28	3.54	56.37	8.23	4.76	57.75
LTS-VoiceAgent w/o orchestrator	2.37	0.34	14.00	1.96	0.12	6.12	2.05	0.35	17.07	2.02	0.45	22.05
LTS-VoiceAgent (Ours)	2.37	0.13	5.46	1.96	0.16	8.16	2.05	0.20	9.76	2.02	0.19	9.4

Table 7: Efficiency analysis for ablations, reporting NFE, NIT, and interruption rate (NIT/NFE). Lower values indicate fewer redundant evaluations and more stable streaming interaction.

Method	VERA			Spoken-MQA			BigBench			Our Bench		
	Acc(%)	Latency	TTFS	Acc(%)	Latency	TTFS	Acc(%)	Latency	TTFS	Acc(%)	Latency	TTFS
Qwen3-32B (Serial)	5.45	726.35	3929.33	79.89	708.4	2588.86	63.26	711.01	2367.32	66.84	723.66	2262.58
Qwen3-32B (LTS-VoiceAgent)	8.15	377.47	402.78	85.73	309.79	592.22	73.4	301.88	588.92	73.89	357.44	829.98

Table 8: Performance of Qwen3-32B under a serial cascade and with LTS-VoiceAgent across all benchmarks. Latency and TTFS are measured in milliseconds (ms).

Method	VERA			Spoken-MQA			BigBench			Our Bench		
	NFE↓	NIT↓	Rate(%)	NFE↓	NIT↓	Rate(%)	NFE↓	NIT↓	Rate(%)	NFE↓	NIT↓	Rate(%)
Qwen3-32B (LTS-VoiceAgent)	2.37	0.33	13.9	1.96	0.28	14.28	2.05	0.23	11.2	2.02	0.20	10.05

Table 9: Efficiency of LTS-VoiceAgent with Qwen3-32B, reporting NFE, NIT, and interruption rate (NIT/NFE) across all benchmarks. Lower values indicate better efficiency and fewer interruptions.

NFE close to two across benchmarks, suggesting that scaling does not inflate the number of reasoning invocations. At the same time, the interruption rate is slightly higher than with Qwen3-8B, consistent with the intuition that longer per-step inference can make intermediate thoughts more susceptible to being overtaken by subsequent transcript updates.

K Qualitative Case Analysis

To evaluate the robustness of LTS-VoiceAgent against streaming input revisions and ASR artifacts, we present a qualitative analysis of three representative scenarios: verbal self-correction, complex logical reasoning, and long-context transformation. As illustrated in Figure 7, all inputs retain original ASR errors (e.g., transcription noise and segmentation faults) to stress-test the systems. We compare our method against baselines including Serial Processing, as well as existing streaming strategies

such as VAD-based method and PredGen. Table 10 summarizes representative raw-input patterns and error modes to make the qualitative comparison explicit.

Frequent Verbal Self-Correction

As shown in the top panel of Figure 7, the user verbally corrects the duration of a scheduled activity (modifying “3 hours” to “2 hours”).

LTS-VoiceAgent. By maintaining a structured JSON state, the Thinker detects the semantic correction and immediately overwrites the variable Watching TV hours from 3 to 2. This ensures the final calculation yields the correct result (36) based on the updated state.

Baselines. The Serial baseline, lacking intermediate state maintenance, fails to capture the overwrite semantics. It performs a linear pass after the stream ends, using the obsolete value to produce an incorrect result (12). Meanwhile, VAD-based

Input pattern	Serial / chunked baseline behavior	LTS-VoiceAgent behavior	Main takeaway
Self-correction: the user first says “3 hours” and later repairs it to “2 hours”.	Treats the obsolete value as still active or triggers on the repair fragment, leading to inconsistent arithmetic.	The Thinker overwrites the state variable (Watching TV hours: 3 $\rightarrow$ 2) before final commitment.	State maintenance helps absorb semantic repairs.
Logical paradox with incremental truthfulness statements.	Long linear reasoning is delayed until turn end; speculative baselines suffer fragmented intermediate chains.	Background planning records key truthfulness states and leaves only final verification after turn end.	Incremental planning reduces post-turn-end reasoning work.
Long-context math query with ASR noise such as “octuplets” $\rightarrow$ “ouplets”.	Earlier constraints are forgotten or overwritten, producing an incorrect final probability.	The Thinker keeps a compact variable table and performs conservative ASR sanitization while retaining the raw transcript.	Compact state tracking improves robustness to noisy long inputs.

Table 10: Representative qualitative examples from Appendix K. The table reports existing case-study patterns rather than additional experiments, addressing what kinds of dialogues were tested and how errors arise across methods.

methods trigger premature reasoning on hesitation markers (e.g., “um no”), causing unnecessary intermediate reasoning and degrading consistency.

Complex Logical Reasoning

The middle panel illustrates a multi-hop logical paradox involving conflicting truthfulness statements (e.g., “Shalanda lies”, “Jamie says Shalanda tells the truth”).

LTS-VoiceAgent. The Thinker constructs the logical chain in the background, anchoring key states (e.g., Jamie: Liar) while the user speaks. The system resolves the paradox incrementally and only requires a final verification step, avoiding dead loops and minimizing response latency.

Baselines. The Serial baseline relies on linear chain-of-thought processing. A single hallucination during the intermediate inference causes the entire logical verification to collapse, leading to an incorrect prediction. The PredGen strategy performs poorly here; frequent interruptions (250–350 ms) fragment the reasoning process, preventing the formation of necessary deep logical dependencies.

Long-Context Multi-Step Transformation

The bottom panel depicts a math word problem containing ASR noise (e.g., “octuplets” recognized as “ouplets”) and multiple constraints.

LTS-VoiceAgent. Despite the noise, the Thinker continuously updates state variables (e.g., Blue bows: 2) throughout the stream. This incremental maintenance ensures that early constraints (color distribution) are preserved, leading to the correct probability calculation (50%).

Baselines. The Serial baseline suffers from forgetting; it fails to recall the specific constraints regarding blue babies mentioned early in the prompt, resulting in a logical gap and an incorrect answer (100%). Similarly, VAD-based and PredGen methods degrade to serial-like performance as improper trigger timing disrupts semantic continuity.

L Dataset Construction Details

This appendix details the construction of our synthetic streaming speech dataset described in Section 4. We propose a pipeline to transform static text benchmarks into realistic spoken interactions, focusing on simulating complex disfluencies such as hesitation and self-correction.

L.1 Data Processing Pipeline

To bridge the gap between text-based benchmarks (GSM8K, MMLU-Pro) and real-world speech interaction, we implemented the multi-stage pipeline illustrated in Algorithm 2. The process begins with removing questions unsuitable for audio (e.g., extensive multiple-choice lists). The core transformation involves two phases:

1. *Scenario-based Contextualization.* Abstract queries are rewritten into first-person, daily-life scenarios to ensure contextual relevance.
2. *Hybrid Linguistic Perturbation.* We employ an LLM to simultaneously inject filler words and simulate cognitive backtracking (self-correction), creating a challenging Hybrid test set.

Finally, the processed text is converted into audio using a TTS model.

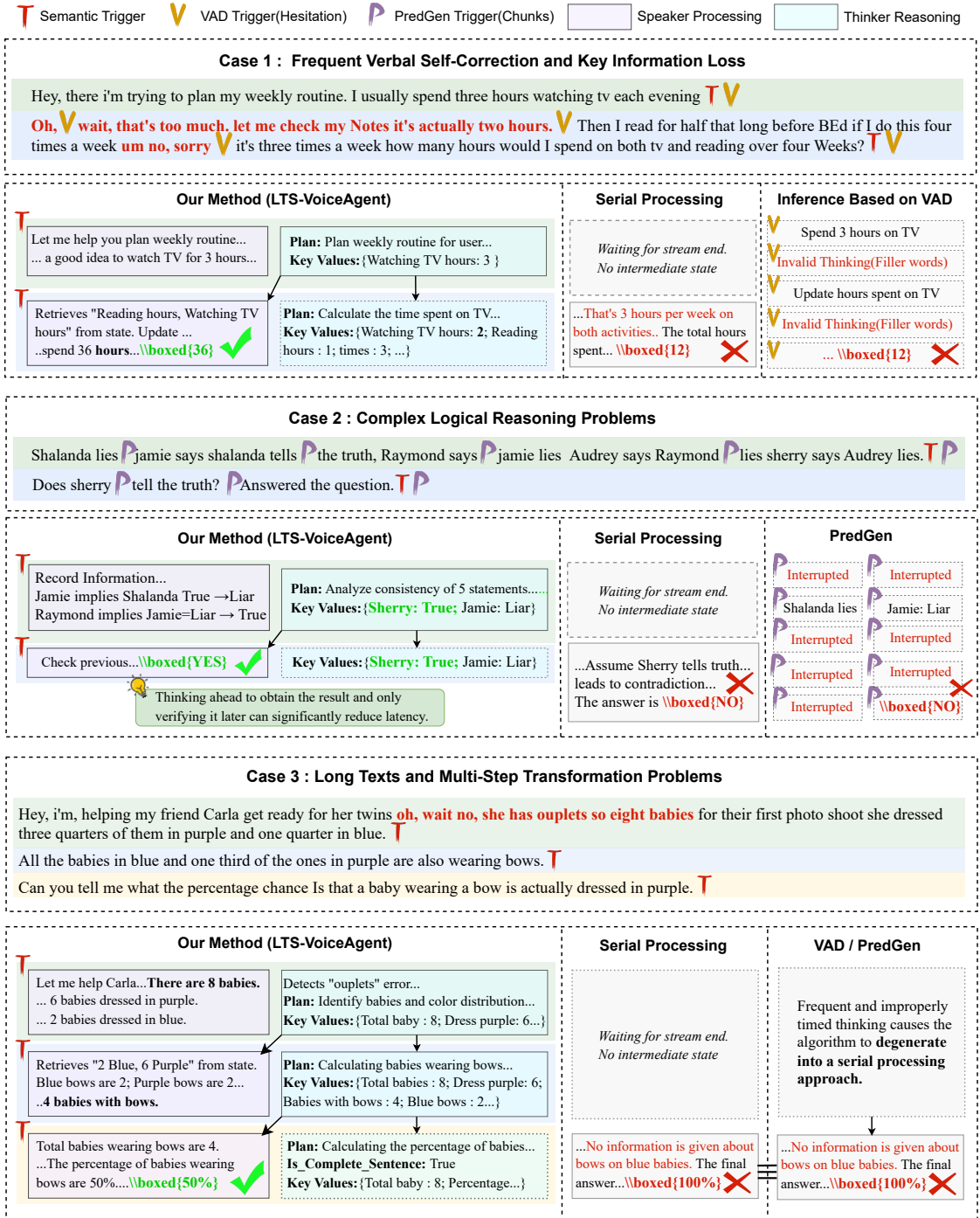


Figure 7: Qualitative comparison of LTS-VoiceAgent against baselines across three scenarios. Top: handling verbal self-corrections via state overwriting. Middle: solving logical paradoxes via look-ahead reasoning. Bottom: managing long-context constraints via incremental updates. The inputs contain raw ASR errors to demonstrate robustness.

Source	Category/Domain	Hybrid Samples	Avg. Tokens	Avg. Audio (s)	Vocab Diversity
GSM8K	Mathematics	1319	67.6	25.2	Med
MMLU-Pro	Math	111	104.7	50.1	Hard
	Physics	104	121.4	57.2	Hard
	Chemistry	92	114.1	60.5	Hard
	Law	89	179.9	71.7	Hard
	Engineering	80	142.5	77.5	Med
	Others (Bio, Chem, etc.)	507	113.3	52.2	Med
	<i>MMLU Subtotal</i>	983	121.7	57.1	Hard
Total	All Domains	2302	90.7	38.9	-

Table 11: Detailed statistics of the synthetic dataset. "Avg. Tokens" refers to the text length after augmentation, and "Avg. Audio" estimates the duration of the synthesized speech. The sample counts for MMLU-Pro subsets are adjusted to ensure a balanced comparison.

Algorithm 2 Synthetic Spoken Dataset Construction Pipeline

Require: Datasets $\mathcal{D}_{src} = \{GSM8K, MMLU-Pro\}$

Require: TTS Model $\mathcal{M}_{tts}$ (CosyVoice 2)

Ensure: Augmented Audio Dataset $\mathcal{A}_{hybrid}$

```

1:  $\mathcal{D}_{clean} \leftarrow \emptyset$ 
   # Phase 1: Scenario-based Contextualization
2: for each sample  $s \in \mathcal{D}_{src}$  do
3:   if  $s$  is suitable then
4:      $s' \leftarrow \text{SCENARIOWRAPPER}(s)$ 
5:      $\mathcal{D}_{clean}.\text{APPEND}(s')$ 
6:   end if
7: end for
   # Phase 2: Hybrid Linguistic Perturbation
8:  $\mathcal{T}_{hybrid} \leftarrow \text{LLM\_AUGMENT}(\mathcal{D}_{clean}, \text{Prompt}_{hybrid})$ 
   # Phase 3: TTS Synthesis
9:  $\mathcal{P}_{audio} \leftarrow \text{"Note pauses in tone, control speed..."}$ 
10: for  $text \in \mathcal{T}_{hybrid}$  do
11:    $audio \leftarrow \mathcal{M}_{tts}(text, \text{prompt} = \mathcal{P}_{audio})$ 
12:    $\mathcal{A}_{hybrid}.\text{APPEND}(audio)$ 
13: end for
14: return  $\mathcal{A}_{hybrid}$ 

```

L.2 Prompt Engineering for Naturalness

We use GPT-4o as the core augmentation engine. To ensure high-quality simulation of spontaneous speech, we designed specific system prompts that leverage the model’s chain-of-thought capabilities.

Phase 1: Scenario-based Contextualization.

This phase transforms third-person exam questions into first-person intents. The prompt (see Figure 8, top) enforces a role-play paradigm to ensure the query sounds like a natural request to a voice assistant.

Phase 2: Linguistic Perturbation. To simulate spontaneous speech, we focus on the Hybrid setting, which combines hesitation markers with self-correction logic. This prompt (see Figure 8, bottom) directs the LLM to identify “cognitive load points” for inserting fillers and “intent shifts” for

corrections.

L.3 Text-to-Speech Synthesis

The linguistically perturbed text is synthesized using CosyVoice 2. We employ specific instructional prompts (e.g., “Pay attention to pauses, control speaking rate”) to ensure the model renders the inserted text artifacts (“umm”, “wait, no”) as natural prosodic events rather than standard lexical tokens.

L.4 Dataset Statistics

Table 11 presents the detailed statistics of our generated dataset. The dataset covers multiple domains from GSM8K (Mathematics) and MMLU-Pro (STEM, Humanities, etc.). We ensured a balanced distribution across the three augmentation types to robustly test the model’s performance under different noise conditions.

L.5 PII and Content Screening

The benchmark is derived from public academic datasets and LLM-generated scenario rewrites rather than personal conversations. We do not intentionally include personally identifying information; generated samples used in evaluation were checked to avoid uniquely identifying personal details or clearly offensive content. Remaining personal names are fictional placeholders inherited from or generated during benchmark construction.

L.6 Licensing and Terms

We release our Pause-and-Repair Benchmark under the MIT License, ensuring consistency with the source datasets. Regarding the existing artifacts utilized in this work, we strictly adhere to their original licenses: GSM8K and MMLU-Pro are used under the MIT License; BigBench Audio, Qwen3,

and CosyVoice 2 are used under the Apache 2.0 License. Our use of these assets for academic research is consistent with their intended purposes and terms of use.

System Prompt: Scenario-based Contextualization

Role: You are an expert conversational designer and creative writer for voice AI assistants. Your task is to rewrite standard math word problems into natural, first-person spoken queries addressed to a virtual assistant.

Task: Rewrite math word problems into natural, first-person spoken queries.

Core Guidelines:

1. First-Person Perspective: Change all third-person narratives to first-person ("I", "me", "my"). If the original problem has no people, inject yourself as the protagonist.
2. Real-Life Context: Construct a realistic daily life scenario (e.g., budgeting, cooking, helping a child with homework).
3. Conversational Openers: Start every rewritten question with a natural spoken greeting (e.g., "Hey," "Hi there," "Ok computer") and a brief intro phrase.
4. Preserve Logic: The mathematical values and logic must remain exactly the same.
5. Natural Flow: The output should sound like something a human would actually say aloud.

Example (Condensed):

Input: "A bag has 232 candies. 54 are red. How many are pink?"

Output: "Hi there. I bought a huge bag of Starbursts with 232 pieces for the party... I only know there are 54 red ones... Could you tell me how many pink candies are left?"

System Prompt: Hybrid Perturbation

Role: You are a data augmentation expert simulating natural, spontaneous human speech. Rewrite the input text to make it sound like a real person talking by inserting **Filler Words** and simulating **Self-Corrections** (changing mind/fixing errors).

Critical Rules:

1. No Solving: Do NOT answer questions. Only rewrite the text provided.
2. Structure Preservation: Keep labels like "Person A:", "Q:", "Answer:" exactly as they are. If the input contains a list of options (e.g., A, B, C...), DO NOT modify the options. Only insert fillers/corrections into the main question.
3. Density Limits: Mix fillers and corrections naturally. Do not overdo it. Placement: Place fillers at logical pauses; place corrections where a logical thought change could occur.

Filler Words Vocabulary:

- *Hesitation:* "umm", "ah", "let me see", "basically"
- *Correction:* "wait, no", "actually, scrap that", "I mean"
- *Delaying:* "uh", "hmm", "so", "anyway"
- *Transition:* "well", "like", "you know", "actually", "basically"

Self-Correction Patterns:

- *Entity Swap:* "In 2015... wait, 2017."
- *Misreading:* "The man walked... oh, the woman walked."
- *Logic Check:* "So I should agree... actually, thinking about the consequences, I disagree."
- *Distraction:* "I need to buy apples... thinking of pie... anyway, I need apples."

Example (Condensed):

Input: "I want to set the price low to gain market share. What is this strategy called?"

Output: "I want to set the price high... **wait, no**, I want to set the price low to **umm**, let me see, gain market share. What is this strategy called?"

Figure 8: Detailed system prompts for the two phases of data construction. Top: Phase 1 Scenario-based Contextualization; Bottom: Phase 2 Hybrid Perturbation.