

RASSG: REPAIR-AWARE SEMANTIC SURVIVAL GATING FOR STREAMING TOOL EXECUTION

Daijin Hu

Beihang University

Xi Wang

Beihang University

ABSTRACT

Streaming speech assistants can launch tools before users finish speaking, overlapping execution with incoming speech to reduce user waiting time. However, subsequent self-corrections can invalidate apparently stable arguments, causing stale execution whose irreversible effects persist even after the final call is corrected. We propose Repair-Aware Semantic Survival Gating (RASSG) to jointly control tool launch and cancellation during streaming speech. RASSG launches a tool when three consecutive speech recognition results arrive with identical nonempty arguments, then combines calibrated argument invalidity risk with execution costs to decide cancellation at the last feasible opportunity before effects become irreversible. Evaluation on ReCAST, FDB-v3, and Audio2Tool uses a controlled execution clock with fixed tool runtimes and cancellation parameters. On ReCAST’s prospective-repair subset, RASSG reduces the stale execution rate from LocalAgreement-2’s 50.00% to 3.15%, with a mean delay of 3.676 s across successful trajectories. The stale reduction also holds against every early-execution baseline on both public speech datasets. Ablations with identical initial launches support active cancellation, showing the value of controlling ongoing calls before their effects become irreversible. Code and data are available at <https://github.com/daijinhu/RASSG>.

Index Terms— Speech agents, tool execution, semantic survival

1. INTRODUCTION

Voice assistants are expanding from conversational responses to completing user tasks through external tools [1, 2, 3, 4]. Streaming speech processing lets systems interpret incoming speech incrementally and form the intents and arguments needed for tool use [5, 6]. Building on this capability, streaming tool execution allows operations to be prepared and launched before an utterance ends, so that speech understanding and task processing proceed in parallel [7]. This overlap can shorten the time users wait for results and support more responsive, continuous interaction.

However, users may correct an argument before finishing their utterance [8], leaving an already launched action inconsistent with their final intent. When an action using a superseded argument crosses its irreversible deadline, it causes *stale execution*, even if the system eventually completes the correct call. Existing strategies face different limitations in handling this problem. Waiting for the complete utterance postpones work that could begin earlier. Recognition stability, as captured by LocalAgreement [9, 10], supports early output but cannot guarantee that the user will not subsequently revise an argument. Waiting until a replacement argument is fully available before updating an action can miss the cancellation window. Streaming tool execution therefore requires joint consideration of whether the current argument is stable enough to launch and whether execution should continue while the tool remains cancellable.

To address the risk of stale execution when users revise arguments after tool launch, we propose Repair-Aware Semantic Survival Gating (RASSG). It coordinates early launch and repair handling in streaming speech tool use. RASSG launches tools based

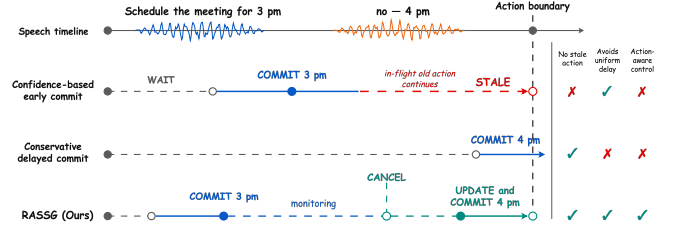


Fig. 1. An illustrative scheduling request changes from 3 pm to 4 pm. Confidence-based early commitment leaves the obsolete call running past its irreversible deadline (the action boundary), while conservative delayed commitment waits for the correction before launching. RASSG launches on stable arguments, monitors invalidity risk during execution, and cancels the old call before the deadline, then launches the revised request.

on observed argument stability and combines predicted argument invalidity with the tool’s cancellation deadline to decide whether to cancel, allowing early execution to remain responsive to subsequent semantic changes.

We evaluate RASSG on synthetic speech with simulated tool execution. On the subset containing subsequent user repairs, RASSG reduces stale execution from 49.92% to 3.15% and successful delay from 4.434 s to 3.676 s relative to a control with identical initial launches but no active cancellation.

Our main contributions are as follows:

1. We characterize execution correctness beyond final call success by jointly modeling argument versions and irreversible deadlines. This formulation captures stale execution that can coexist with eventual success and makes opportunities for repair during execution explicit.
2. We introduce RASSG, which uses observed argument stability for launch and predicted argument invalidity together with cancellation deadlines for cancellation. This design lets the system act on stable input while continuing to assess whether an action should be withdrawn in response to subsequent repairs.
3. We demonstrate reductions in stale execution and successful delay in controlled synthetic speech experiments relative to a control with identical initial launches and no active cancellation. Component and execution-semantics analyses identify the roles of launch stability, cancellation signals, and progress reuse, providing empirical evidence for the risk–latency trade-off in streaming tool execution.

2. RELATED WORK

Tool execution after complete input generates calls from a full utterance or a received dialogue turn. Speech-to-tool systems either map audio directly to calls or transcribe speech before parsing tools and

arguments [1]. When requests lack necessary information, clarification methods recover unspecified intents and correct errors across turns [11]. Related benchmarks examine goal revisions, missing parameters, argument correction, and recovery from tool failures [12, 13]. These developments improve the handling of incomplete or revised requests using available dialogue context. However, workflows that wait for the complete utterance leave tool processing until after the user finishes speaking.

Early execution from incremental input overlaps processing with ongoing speech. Incremental dialogue and streaming transcription provide mechanisms for revision propagation and stable-prefix confirmation [14, 10]. Related approaches move response generation and reasoning into the input period through speculative computation or semantic triggering [15, 16], while speech agents interleave or synchronize reasoning and tool actions with incoming audio [6, 2]. Two works directly address changes to early calls. Stream RAG [7] generates queries during speech; its model-triggered variant terminates earlier queries when a new one is issued. Speculative Interaction Agents [17] handles user input and tool returns asynchronously, permits modification or removal of unfinished calls, and defers sensitive tool execution until confirmation. These mechanisms establish support for revising or withdrawing speculative calls. For tools that permit cancellation within a finite window, the remaining decision concerns whether an ongoing action should continue given future argument invalidity, cancellation cost, and the time remaining before irreversibility.

Evaluation has also expanded beyond isolated call accuracy to cover conversational correction and execution progress. Stateful and speech-based tool benchmarks collectively assess intermediate states, repairs in human speech, and verifiable task completion during full-duplex interaction [18, 8, 19, 20]. These evaluations provide a basis for examining how evolving intent affects ongoing tool use. Within the incremental execution approach, RASSG focuses on cancellation decisions under finite tool deadlines. It uses observed argument stability for launch and predicted argument invalidity for cancellation, accounting for actual recognition arrivals and cancellation acknowledgements. This connects the decision to continue or withdraw an action with the joint evaluation of stale execution and successful delay.

3. METHODOLOGY

RASSG coordinates early tool launch with repairs during execution. As shown in Fig. 2, parsed arguments feed stability-based launch and invalidity prediction, while newly recognized arguments update active calls. The controller combines calibrated risk with tool configuration and execution state to decide whether to cancel at the last feasible opportunity before the current version becomes irreversible.

3.1. Task Formulation and Execution Clock

At observation time t , the parser extracts a candidate argument v_t from ASR results that have already arrived. Its history contains only observations available up to t . Transcription changes need not change the normalized argument. Different argument values define distinct execution versions. A version inconsistent with the final target causes stale execution if it survives its irreversible deadline, even when a later call succeeds. The final target is used only for offline supervision and scoring.

For tool a , we assume that runtime ℓ_a , cancellation window w_a , and cancellation cost are fixed and known during an execution. Cancellation acknowledgement latency is also a fixed configuration parameter. A version launched at c on a pipeline started at s becomes irreversible at

$$b = \min\{c + w_a, \max(c, s + \ell_a)\}. \quad (1)$$

The deadline is bounded by both the cancellation window and completion time. A fresh call has $s = c$, while a timely argument update

can retain the original pipeline start s . Progress reuse requires completed computation to remain valid after the argument changes, for example when preparation is independent of the revised field. Conveyor [21] demonstrates tool interfaces that overlap such partial work with input generation; reuse across argument versions additionally requires that work to remain val

3.2. Stability-Based Tool Launch

Let $\hat{v}_i$ denote the normalized argument from the i th actual ASR arrival. An idle controller issues COMMIT when the latest three arrivals satisfy $\hat{v}_{i-2} = \hat{v}_{i-1} = \hat{v}_i = v_t \neq \emptyset$; otherwise it issues WAIT. This applies local agreement [9, 10] to argument values rather than complete transcriptions. Only a new ASR arrival changes the agreement history. Repeated control ticks using the same recognition therefore do not increase the agreement count. After the final ASR flush arrives, an idle controller can directly commit a nonempty candidate, ending regular waiting. Agreement provides evidence of current recognition stability, but a user may still revise that argument. Execution therefore remains subject to invalidity prediction and cancellation.

3.3. Calibrated Argument Invalidity Prediction

A causal GRU with 64 hidden units [22] encodes confidence, parse history, prefix text, pauses, and version age. A supersession event is the earlier of the current version’s annotated user-repair onset and the next nonempty ASR argument change. An unresolved repair denotes ongoing invalidity. The predicted distribution assigns masses $p_{0,t}$ to ongoing invalidity, $p_{k,t}$ to eight future intervals, $p_{>,t}$ to events beyond the prediction horizon, $p_{T,t}$ to a wrong version persisting to termination, and $p_{\emptyset,t}$ to a correct version persisting to termination. These outcome groups form a normalized distribution over the current argument version. The aggregate risk used for control is

$$p_{\text{bad},t} = p_{0,t} + \sum_{k=1}^8 p_{k,t} + p_{>,t} + p_{T,t}. \quad (2)$$

This quantity estimates argument invalidity, with tool timing and costs entering the subsequent decision.

The training objective combines censored negative log-likelihood [23] and masked binary cross-entropy at cumulative event boundaries, with each argument version receiving equal total weight. A source-trained logistic calibrator rescales total supersession-event mass, preserving relative proportions within event and non-event branches and the conditional time shape. The probabilities in Eq. 2 denote the calibrated masses. Event annotations and final targets supply offline labels, whereas online prediction uses causal observations. The resulting aggregate risk drives cancellation. Launch remains governed by observed agreement. Conditional time shape is used by the separately evaluated lookahead-launch variant.

3.4. Deadline-Aware Cancellation and Execution Updates

For an active call whose argument still matches the nonempty candidate, the last-chance veto checks risk at the final regular decision that permits timely cancellation. Let d_t include processing and any wait for previous acknowledgements, Δ be the control period, and κ the cancellation acknowledgement delay. This opportunity satisfies $d_t + \kappa \leq b < d_t + \Delta + \kappa$. Timeliness is determined by acknowledgement. Earlier decisions continue execution, allowing a replacement to arrive while progress remains reusable.

The veto compares the cost of continuing a potentially invalid version with the cost of cancelling and losing useful progress. Let C_{stale} and C_{cancel} denote stale-execution and cancellation costs, and let β_a price delay. At decision time d_t , reusable progress is $g_t = \min\{\ell_a, \max(0, d_t - s)\}$. The controller cancels when

$$C_{\text{stale}}p_{\text{bad},t} > C_{\text{cancel}} + \beta_a g_t (1 - p_{\text{bad},t}). \quad (3)$$

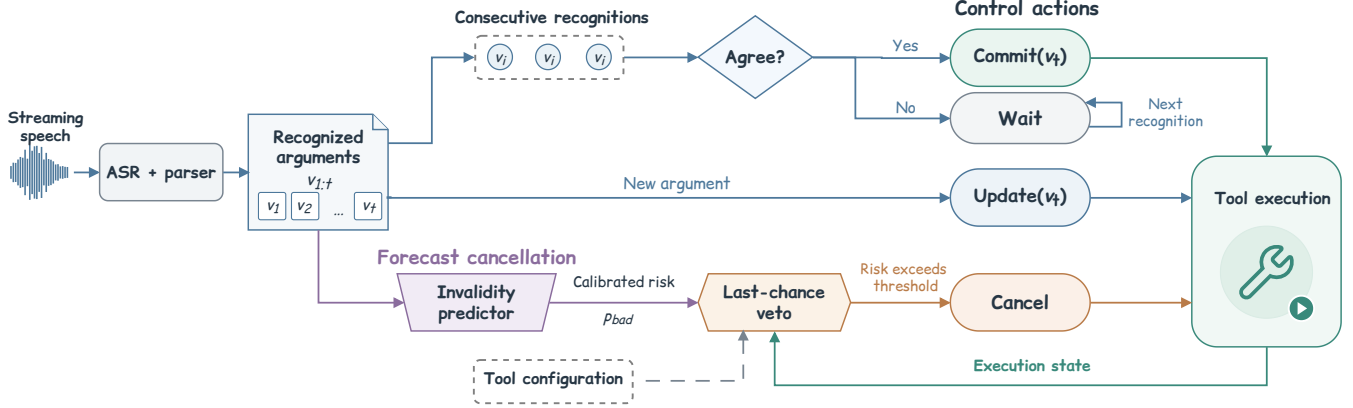


Fig. 2. RASSG control loop. Agreement across three actual ASR arrivals triggers launch. New arguments update active execution, while calibrated invalidity risk, tool configuration, and execution state determine cancellation at the last feasible opportunity.

The left side assigns stale-execution cost to predicted invalidity. The right side combines cancellation cost with the delay cost of discarding progress when the argument would remain valid. Thus, the cancellation threshold depends on both tool costs and accumulated execution progress. The rule uses no additional cost margin.

A new nonempty argument differing from the active version takes priority over the veto. If its UPDATE is acknowledged by the current deadline, the replacement retains s and receives a new version start and deadline under Eq. 1. A late replacement starts fresh execution and preserves any stale-execution record already incurred. CANCEL instead clears the active call and its progress upon acknowledgement, without recommitting in the same observation. At a later observation, the idle controller may commit again once the launch condition holds. If the veto condition is not met, execution continues. Each transition updates the execution state used for the next decision.

4. EXPERIMENTS

4.1. Dataset Design

We construct ReCAST (Repair and Early Commitment Assessment for Streaming Tools), a synthetic speech dataset for evaluating early execution and subsequent repairs. Time, amount, location, and contact requests cover ordinary speech and different repair timings. Stable and corrected pairs share audio before branching. Source data contain 320 training and 128 development recordings. Speech-connected groups divide development equally into model-selection and policy-calibration sets. The test set contains 2,400 recordings from two new voices and new templates, with 1,200 prospective-repair recordings used for the main comparison.

Public evaluation covers all 100 FDB-v3 recordings [8], and 2,145 Audio2Tool Tier 6 correction queries meeting the input-duration limit [1], with one released rendition per query.

4.2. Baselines

We compare RASSG with five baselines: End-of-Utterance Execution, LocalAgreement-2, Confidence commit, and adaptations of Stream RAG and Speculative Tool Calling. End-of-Utterance Execution (EOU) waits for speech to end and final recognition to arrive. LocalAgreement-2 [9, 10] launches after argument agreement across two actual ASR arrivals. Confidence commit [24] is our threshold control using source-selected confidence 0.95. Stream RAG [7] launches parallel calls for each arriving nonempty block proposal, including duplicates, then selects the earliest call matching the final observed plan and cancels unfinished alternatives

at EOU. Speculative Tool Calling [17] executes the first proposal immediately. Changed arguments trigger withdrawal and fresh execution after acknowledgement. Stream RAG assumes parallel tool capacity. Speculative Tool Calling evaluates eager execution, while its final-confirmation branch reduces to EOU under our atomic-plan representation.

4.3. Evaluation Setup

All policies share Whisper small.en [25] prefixes decoded every 960 ms. ReCAST uses a fixed-lexicon parser. Public speech uses Qwen2.5-3B-Instruct [26] with 4-bit weights to parse current prefixes and public tool schemas into complete plans. Arrivals include measured frontend service times. We execute tool calls within an event-clock execution model that processes commitment, update, cancellation acknowledgement, and irreversible-boundary events. Tool runtimes and cancellation costs are static configuration parameters. Control intervals, processing budgets, and acknowledgement delays are 320, 10, and 20 ms. Timely UPDATE reuses progress. Restarting after CANCEL does not.

We report two metrics. Stale execution rate is the fraction of all trajectories with at least one stale execution. Successful delay is measured only when the final argument is correct and an active call exists. Its origin is final-argument acoustic availability for ReCAST and waveform end for public speech, with negative values clipped to zero. Because these origins differ, delay comparisons are made within each dataset. Table delay means condition on each policy’s own successful trajectories, whereas stale execution retains all trajectories.

5. RESULTS

5.1. Main Results

RASSG achieves the lowest stale execution rate among all four early-execution baselines on every dataset in Table 1. On ReCAST, it reduces stale execution by 93.71% relative to LocalAgreement-2 while shortening successful delay by 0.278 s. It also removes 5.000 s of waiting relative to EOU. Against Speculative Tool Calling, stale execution falls from 50.00% to 3.15% for 0.197 s additional paired delay.

On public speech, RASSG lowers stale execution relative to Speculative Tool Calling by 20.00 percentage points on FDB-v3 and 26.39 on Audio2Tool. Reductions against Stream RAG are 29.00 and 29.84 points. All four early-execution comparisons on both public cohorts have stale-rate intervals excluding zero. With identical parsed proposals, these differences arise from execution control.

Table 1. Comparison of RASSG and baseline execution policies on ReCAST, FDB-v3, and Audio2Tool. Lower is better for all metrics.

Method	ReCAST		FDB-v3		Audio2Tool	
	Stale execution rate (%) ↓	Successful delay (s) ↓	Stale execution rate (%) ↓	Successful delay (s) ↓	Stale execution rate (%) ↓	Successful delay (s) ↓
EOU	0.58	8.676	40.00	23.880	52.73	9.196
LocalAgreement-2	50.00	3.954	65.00	9.210	62.91	8.135
Confidence commit	21.08	5.633	73.00	9.507	73.92	7.910
Stream RAG	50.00	5.312	86.00	19.997	84.50	8.255
Speculative Tool Calling	50.00	3.480	77.00	9.280	81.05	8.216
RASSG	3.15	3.676	57.00	11.072	54.65	8.797

Table 2. Component ablations of RASSG on ReCAST, changing one design choice at a time.

Variant	Stale execution rate (%) ↓	Successful delay (s) ↓
RASSG (full)	3.15	3.676
<i>Component removals</i>		
Remove active cancellation	49.92	4.434
Remove progress reuse	3.06	4.510
<i>Launch consensus</i>		
One-arrival consensus	28.78	3.038
Two-arrival consensus	16.86	3.274
Four-arrival consensus	0.88	4.450

Final selection among parallel calls or withdrawal after a replacement arrives can leave earlier effects irreversible; RASSG combines stronger launch evidence with a veto that can act before the parsed argument changes.

The high absolute stale rates on FDB-v3 and Audio2Tool largely reflect persistent plan errors in the shared ASR-to-tool parsing frontend. Of RASSG’s stale trajectories, 70.18% and 96.47%, respectively, have a wrong nonempty final parser output and also incur stale execution under EOU. This exposure persists even when launch waits for complete input. On recordings with correct final parser outputs, RASSG lowers stale execution from LocalAgreement-2’s 50.00% to 32.14% and from 20.10% to 3.33%. Thus, RASSG suppresses obsolete intermediate plans even when the frontend eventually resolves the request correctly.

5.2. Ablation Study

Table 2 separates component removals from launch consensus. With first commitments and reactive argument updates fixed, removing active cancellation raises stale execution from 3.15% to 49.92%. Cancellation reduces stale execution by 46.77 percentage points, with a 95% interval of [45.71, 47.78], and common-success delay by 0.758 s [0.702, 0.815]. The gain therefore comes from controlling an ongoing call, rather than delaying its first launch.

Removing progress reuse increases successful delay from 3.676 to 4.510 s, a paired penalty of 0.834 s [0.788, 0.881]. Its stale rate falls slightly, from 3.15% to 3.06%, because restarting can move the irreversible boundary later. The 0.083-percentage-point reduction comes from one recording across eight actions and three checkpoints, with a paired 95% interval of [0.000, 0.256]. Progress reuse primarily saves waiting.

Consensus controls how much speech evidence precedes launch. Moving from two to three arrivals removes 13.71 percentage points of stale execution for 0.402 s additional waiting; moving from three to four removes another 2.27 points for 0.774 s. The fourth arrival costs more delay for a smaller absolute risk reduction. We adopt

three-arrival consensus as the risk–delay operating point studied here and report the trade-offs of all other evaluated consensus settings.

5.3. Results Analysis

The method separates launch evidence from continuation risk. Argument agreement filters transient proposals before commitment; calibrated invalidity forecasts supersession after launch. Launch-only confidence or agreement rules do not make this second decision, and replacement-triggered withdrawal waits for the new argument. RASSG can act before replacement resolution, using the deadline and reusable progress to determine whether intervention remains feasible and its predicted benefit exceeds discarded work.

ReCAST’s matched-launch trajectories directly support this mechanism. Among 28,800 recording–action–checkpoint combinations, cancellation removes stale execution from 13,470 trajectories without introducing a newly stale trajectory. In every avoided case, cancellation precedes the no-cancellation control’s first irreversible error. Late replacements fall from 14,304 to 834: withdrawal followed by recommitment renews the version’s cancellation window, allowing more subsequent corrections to arrive in time. Timely updates preserve valid work, explaining why stale reduction coincides with shorter delay. The reuse removal verifies this division of roles: the veto protects argument versions, while retained computation supplies the latency benefit.

6. CONCLUSION

RASSG combines argument-consensus launch with calibrated, deadline-aware cancellation. Under fixed execution parameters, it reduces stale execution relative to early-execution baselines across three speech datasets. Matched-launch removal experiments distinguish the protection supplied by cancellation from the waiting-time reduction supplied by progress reuse. Beyond three arrivals, further confirmation costs more waiting for less absolute risk reduction. Execution control must therefore remain responsive to repairs after launch.

7. COMPLIANCE WITH ETHICAL STANDARDS

The reported experiments use synthetic ReCAST speech and the publicly released FDB-v3 and Audio2Tool datasets [8, 1]. They involve no participant recruitment, collection of new human recordings, or intervention with human participants.

8. ACKNOWLEDGMENTS

No funding was received for this study. There are no conflicts of interest, financial or non-financial.

Large language models were used for (i) assistance with manuscript drafting and polishing, (ii) related-work search and discovery, and (iii) code generation.

9. REFERENCES

- [1] Ramit Pahwa, Apoorva Beedu, Parivesh Priye, Rutu Gandhi, Saloni Takawale, Aruna Baijal, and Zengli Yang, “Audio2tool: Speak, call, act—a dataset for benchmarking speech tool use,” *arXiv preprint arXiv:2604.22821*, 2026.
- [2] Haoyang Zhang, Jun Chen, Donghang Wu, Yuxin Li, Yuxin Zhang, Xiangyu Tony Zhang, Che Liu, Qingjian Lin, Yizhou Peng, Hexin Liu, et al., “Duplexsla: A full-duplex spoken language model with synchronized speech, language, and action,” *arXiv preprint arXiv:2605.20755*, 2026.
- [3] Xiang Li, Duyi Pan, Hongru Xiao, Jiale Han, Jing Tang, Jiabao Ma, Wei Wang, and Bo Cheng, “Dialogueagents: A hybrid agent-based speech synthesis framework for multi-party dialogue,” in *2025 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2025, pp. 1–6.
- [4] Yifan Sui, Han Zhao, Rui Ma, Zhiyuan He, Hao Wang, Jianxun Li, and Yuqing Yang, “Act while thinking: Accelerating llm agents via pattern-aware speculative tool execution,” *arXiv e-prints*, pp. arXiv–2603, 2026.
- [5] Junteng Jia, Gil Keren, Wei Zhou, Egor Lakomkin, Xiaohui Zhang, Chunyang Wu, Frank Seide, Jay Mahadeokar, and Ozlem Kalinli, “Efficient streaming llm for speech recognition,” in *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2025, pp. 1–5.
- [6] Cheng-Han Chiang, Xiaofei Wang, Linjie Li, Chung-Ching Lin, Kevin Lin, Shujie Liu, Zhendong Wang, Zhengyuan Yang, Hung-yi Lee, and Lijuan Wang, “Shanks: Simultaneous hearing and thinking for spoken language models,” in *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2026, pp. 8951–8972.
- [7] Siddhant Arora, Haidar Khan, Kai Sun, Xin Luna Dong, Sajal Choudhary, Seungwhan Moon, Xinyuan Zhang, Adithya Sagar, Surya Teja Appini, Kaushik Patnaik, et al., “Stream rag: Instant and accurate spoken dialogue systems with streaming tool usage,” *arXiv preprint arXiv:2510.02044*, 2025.
- [8] Guan-Ting Lin, Chen Chen, Zhehuai Chen, and Hung-yi Lee, “Full-duplex-bench-v3: Benchmarking tool use for full-duplex voice agents under real-world disfluency,” *arXiv preprint arXiv:2604.04847*, 2026.
- [9] Danni Liu, Gerasimos Spanakis, and Jan Niehues, “Low-latency sequence-to-sequence speech recognition and translation by partial hypothesis selection,” *arXiv preprint arXiv:2005.11185*, 2020.
- [10] Dominik Macháček, Raj Dabre, and Ondřej Bojar, “Turning whisper into real-time transcription system,” in *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics: System Demonstrations*, 2023, pp. 17–24.
- [11] Xuan Zhang, Yongliang Shen, Zhe Zheng, Linjuan Wu, Wenqi Zhang, Yuchen Yan, Qiuying Peng, Jun Wang, and Weiming Lu, “Asktoact: Enhancing llms tool use via self-correcting clarification,” in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025, pp. 13495–13522.
- [12] Tamer Alkhoul, Katerina Margatina, James Gung, Raphael Shu, Claudia Zaghi, Monica Sunkara, and Yi Zhang, “Confetti: Conversational function-calling evaluation through turn-level interactions,” in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025, pp. 7993–8006.
- [13] Zhiwei Dong, Ruihao Gong, Yang Yong, Shuo Wu, Yongqiang Yao, Song-Lu Chen, and Xu-Cheng Yin, “Tool playgrounds: A comprehensive and analyzable benchmark for llm tool invocation,” in *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2025, pp. 1–5.
- [14] David Schlangen and Gabriel Skantze, “A general, abstract model of incremental dialogue processing,” *Dialogue & Discourse*, vol. 2, pp. 83–111, 2011.
- [15] Shufan Li and Aditya Grover, “Predgen: Accelerated inference of large language models through input-time speculation for real-time speech interaction,” *arXiv preprint arXiv:2506.15556*, 2025.
- [16] Wenhao Zou, Yuwei Miao, Zhanyu Ma, Jun Xu, Jiuchong Gao, Jinghua Hao, Renqing He, and Jingwen Xu, “Lts-voiceagent: A listen-think-speak framework for efficient streaming voice interaction via semantic triggering and incremental reasoning,” *arXiv preprint arXiv:2601.19952*, 2026.
- [17] Coleman Hooper, Minwoo Kang, Suhong Moon, Nicholas Lee, Eric Wen, John Wawrzynek, Michael W Mahoney, Yakun Sophia Shao, Amir Gholami, and Kurt Keutzer, “Speculative interaction agents: Building real-time agents with asynchronous i/o and speculative tool calling,” *arXiv preprint arXiv:2605.13360*, 2026.
- [18] Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al., “Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities,” in *Findings of the Association for Computational Linguistics: NAACL 2025*, 2025, pp. 1160–1183.
- [19] Soham Ray, Keshav Dhandhan, Victor Barres, and Karthik Narasimhan, “ τ -Voice: Benchmarking full-duplex voice agents on real-world domains,” *arXiv preprint arXiv:2603.13686*, 2026.
- [20] Bhaskar Gurram, “Evaluating tool-using language agents: Judge reliability, propagation cascades, and runtime mitigation in agentprop-bench,” *arXiv preprint arXiv:2604.16706*, 2026.
- [21] Yechen Xu, Xinhao Kong, Tingjun Chen, and Danyang Zhuo, “Conveyor: Efficient tool-aware llm serving with tool partial execution,” *arXiv preprint arXiv:2406.00059*, 2024.
- [22] Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, “Learning phrase representations using rnn encoder–decoder for statistical machine translation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1724–1734.
- [23] Changhee Lee, William Zame, Jinsung Yoon, and Mihaela Van Der Schaar, “Deephit: A deep learning approach to survival analysis with competing risks,” in *Proceedings of the AAAI conference on artificial intelligence*, 2018, vol. 32.
- [24] Sebastian Loth, Katharina Jettka, Manuel Giuliani, Stefan Kopp, and Jan P de Ruiter, “Confidence in uncertainty: Error cost and commitment in early speech hypotheses,” *PLoS one*, vol. 13, no. 8, pp. e0201516, 2018.
- [25] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever, “Robust speech recognition via large-scale weak supervision,” in *International conference on machine learning*. PMLR, 2023, pp. 28492–28518.
- [26] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al., “Qwen3 technical report,” *arXiv preprint arXiv:2505.09388*, 2025.