SEARCH-ASR: TRAINING SPEECH AGENTS TO SEARCH FOR ENTITY-AWARE SPEECH RECOGNITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models have advanced automatic speech recognition (ASR), improving multilingual and dialectal recognition and contextual understanding. However, despite accurately transcribing most of an utterance, ASR systems still struggle with emerging and long-tail entities owing to static parametric knowledge and limited coverage of prebuilt hotword retrieval resources. This paper introduces Search-Augmented Agentic ASR (Search-ASR), which trains speech agents to autonomously query Web search engines using acoustic cues and utterance context, retrieve candidate names and semantic information, and combine this evidence with the audio to produce transcriptions. We further develop a multi-stage post-training framework and a multi-agent data synthesis system that establish search capabilities through supervised finetuning and reinforcement learning while reducing redundant searches and unnecessary rewriting. Experiments on English and Chinese entity-focused ASR benchmarks show that Search-ASR outperforms all evaluated open-source baselines and remains competitive with evaluated closed-source systems. Compared with its initialization model, Qwen3-Omni, Search-ASR reduces English word error rate and Chinese character error rate by 53.1% and 25.5%, while improving entity recall in both languages. [1]

1 INTRODUCTION

Large language models (LLMs) are expanding the capabilities of automatic speech recognition (ASR). By integrating acoustic modeling with language knowledge, recent LLM-based ASR models have advanced multilingual and dialectal recognition while enabling richer contextual understanding of spoken language (An et al., 2025; Chu et al., 2024; Shi et al., 2026; Xu et al., 2025).

Nevertheless, even when an ASR system correctly transcribes most of an utterance, it may still misrecognize a newly released product, an uncommon personal name, or a specialized term (Li et al., 2024). For example, on a speech sample containing "Qwen 3.8 Max", all seven evaluated open-source ASR or large audio-language models (LALMs) misrecognized the entity, producing variants such as "Qun" or "Quinn". These entities often occupy only a small portion of an utterance but convey essential referential information. This raises a central question: Why do ASR models with extensive language knowledge still struggle to recognize emerging and long-tail entities? We examine this question from three perspectives.

First, **pronunciation and utterance-level context may not uniquely determine an entity's correct spelling.** For phonetically similar names, context can narrow the candidate set but may lack the entity knowledge needed for disambiguation, resulting in phonetically plausible yet referentially incorrect transcriptions (Gong et al., 2025). Second, **the entity knowledge required for disambiguation is not fully captured in model parameters.** New entities continually emerge, while long-tail entities are underrepresented in training, making these names difficult to recognize from parametric knowledge alone (Huber & Waibel, 2026). Finally, **hotword retrieval helps address this limitation but remains constrained by vocabulary coverage and the richness of retrieved evidence.** Contextual biasing provides external candidates (Pundak et al., 2018), but prebuilt vocabularies may not keep pace with emerging entities, and candidate names alone may be insufficient for semantic disambiguation. Models therefore need on-demand access to candidate entities and knowledge relevant to the current utterance.

[1] Model weights, benchmarks, and selected synthetic training data will be released upon acceptance

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

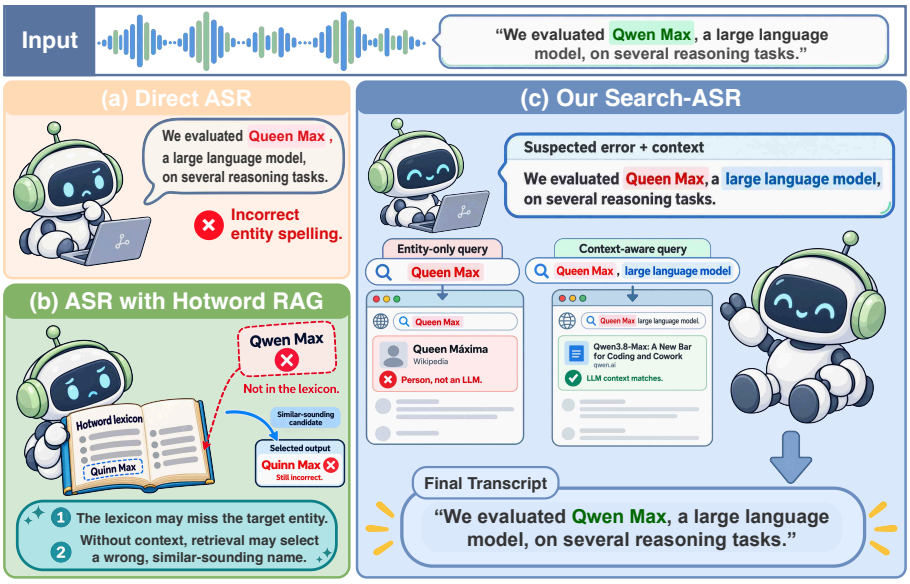

Figure 1: Entity recognition with three ASR approaches. (a) Direct ASR misspells the entity. (b) Hotword retrieval is limited by vocabulary coverage and semantic context. (c) Our Search-ASR uses Web evidence to correct the entity while preserving the remaining transcript.

Motivated by this analysis, we introduce **Search-Augmented Agentic ASR (Search-ASR)**, which enables speech agents to autonomously invoke Web search based on acoustic cues and utterance context. Search-ASR retrieves candidate names and semantic information beyond prebuilt vocabularies from the public Internet, supplements parametric knowledge, and combines retrieved evidence with the audio to resolve entity ambiguity. Figure 1 compares the inference processes of direct ASR, hotword-retrieval-augmented ASR, and Search-ASR. However, search results can also induce additions and rewrites that are unsupported by the original speech. To teach agents when to search, how to use evidence, and how to preserve transcription fidelity, we post-train Qwen3-Omni (Xu et al., 2025) using a multi-stage training framework and a multi-agent data synthesis system. Through search-policy learning, rigorous data filtering, and faithfulness constraints, we jointly optimize search decisions and transcription quality. Our contributions are threefold:

- **We introduce Search-ASR for entity-aware speech recognition.** Search-ASR incorporates Web search into the ASR Agent's decision-making process, using external candidate names and semantic information to address knowledge gaps in emerging and long-tail entity recognition.
- **We develop a multi-stage post-training and multi-agent data synthesis system.** Search-policy learning, rigorous trajectory filtering and general ASR data mixing jointly improve entity recognition and search behavior, while failure-trajectory-driven synthesis expands the set of training tasks.
- **We demonstrate improvements in entity recognition.** Search-ASR achieves leading English and Chinese entity recognition results among the evaluated open-source models and matches or exceeds the evaluated closed-source systems on selected metrics. Second-stage training further improves general ASR performance while reducing search frequency and generation overhead.

2 Related Work

2.1 Automatic Speech Recognition

Large-scale pretraining has advanced general ASR. Whisper (Radford et al., 2023) improves robustness through large-scale weak supervision, while audio-language models such as Qwen2-Audio (Chu et al., 2024) unify transcription with broader audio understanding tasks. To improve recognition of rare words and named entities, contextual biasing methods incorporate external phrases or vocabularies: CLAS (Pundak et al., 2018) jointly models speech and contextual phrases, and TCPGen (Sun et al., 2021) integrates candidate words into decoding through a tree-constrained

pointer generator. Retrieval-based approaches extend this context: BR-ASR (Gong et al., 2025) supports scalable candidate retrieval through speech–text contrastive learning, while document-based retrieval-augmented ASR (Shen et al., 2025) supplies domain-relevant text at inference time. Another line of work introduces feedback and revision into speech recognition. Interactive ASR (Wang et al., 2026) refines transcriptions through multi-turn semantic feedback and evaluates semantic correctness using an LLM judge. AgenticASR (Jiang et al., 2026) adopts an ASR–Refiner architecture that incrementally revises transcripts, focusing on disfluencies, self-corrections, and written-form output. These approaches supplement model knowledge, but their coverage depends on the available retrieval resources or humans' feedback.

2.2 Tool Use and Agentic Reinforcement Learning

LLM agents extend generation into a decision process involving reasoning, tool calls, and environmental feedback. ReAct (Yao et al., 2022) interleaves reasoning with actions, while Toolformer (Schick et al., 2023) learns when and how to invoke tools through self-supervised data construction. AgentTuning (Zeng et al., 2024) mixes agent trajectories with general instruction data to develop tool-use capabilities while retaining general competence. Reinforcement learning (RL) further enables autonomous search: Search-R1 (Jin et al., 2025) and R1-Searcher (Song et al., 2025) optimize retrieval through outcome rewards, while DeepResearcher (Zheng et al., 2025) trains agents in real Web environments. Tool use has also expanded to audio-language models. Spoken Function Calling (Peng et al., 2026a) maps spoken instructions to structured function calls. Step-Audio 2 (Wu et al., 2025) integrates external tools into end-to-end speech conversation. Stream RAG (Arora et al., 2025) learns to issue tool queries during incoming speech and incorporate retrieved evidence into spoken responses. While these systems use external knowledge to support speech conversation, Search-ASR focuses on resolving entity ambiguity while preserving transcription fidelity.

3 Methods

3.1 Problem Formulation and Search-ASR Inference

Given an audio input x, ASR aims to produce its corresponding transcription y. Without external context, recognition is modeled as $p_\theta(y \mid x)$, where θ denotes the model parameters. Retrieval-augmented ASR additionally retrieves audio-relevant context from an external knowledge base $\mathcal{B}$:

$$c_x = \mathcal{R}_\phi(x; \mathcal{B}), \quad p_{\theta,\phi}(y \mid x, \mathcal{B}) = p_\theta(y \mid x, c_x). \tag{1}$$

Here, $\mathcal{R}_\phi$ is a retrieval procedure parameterized or configured by ϕ, and c_x contains retrieved candidate terms or relevant text. Retrieval may operate directly on acoustic features or use a preliminary transcription to construct queries. Thus, the context supplied to the recognition model varies with the input audio.

Search-ASR places search timing, query construction, and termination under a learned agent policy, allowing search decisions to adapt to previous interactions. Let π_θ denote the agent policy and h_t the complete history before decision step t, including task instructions, the input audio, previous tool interactions, and prior reasoning. At each step, the agent selects an action a_t:

$$a_t \sim \pi_\theta(\cdot \mid h_t), \quad a_t \in \{(m, q) : m \in \mathcal{T}\} \cup \{\mathrm{finish}(y)\}. \tag{2}$$

where $\mathcal{T}$ contains Web search and an optional hotword retrieval tool. A search action returns an observation o_t from the tool environment, which is appended to the history, whereas $\mathrm{finish}(y)$ outputs the transcription and terminates inference. When no search action is invoked, Search-ASR reduces to direct transcription. As shown in Figure 2(a), the agent determines when and what to search, while the execution layer manages search budgets and caching. Retrieved evidence supports entity disambiguation, but the final transcription must remain faithful to the audio.

3.2 Multi-Stage Agentic Post-training

Starting from Qwen3-Omni (Xu et al., 2025), we apply four post-training steps: cold-start SFT, search RL, rejection-sampling SFT, and mixed-data RL (Figure 2(b)). Before each RL round, the preceding SFT model is used to identify capability gaps and construct additional training tasks through Section 3.3.

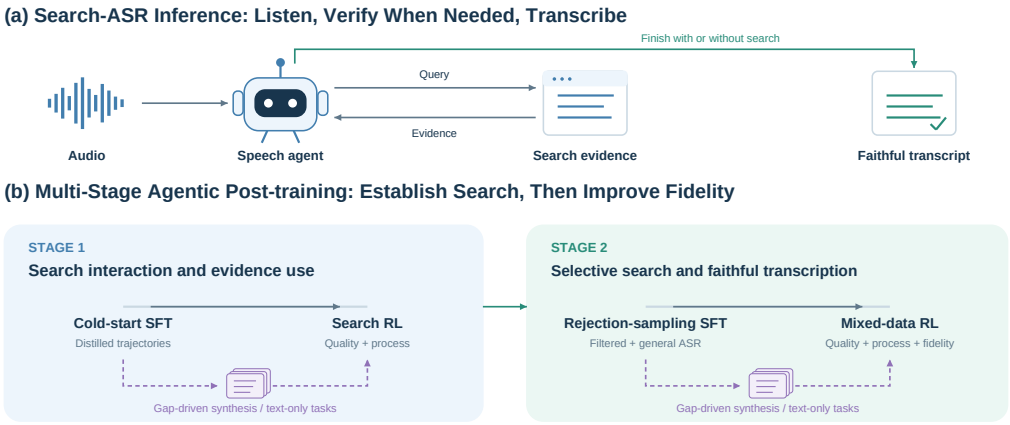

Figure 2: Overview of Search-ASR. (a) The agent either transcribes directly or searches for entity knowledge and transcribes. (b) Cold-start SFT and search RL establish tool-use capabilities, while rejection-sampling SFT and mixed-data RL improve transcription faithfulness.

Cold-start SFT. We first train on search-augmented ASR trajectories distilled from Qwen3.7-Max (Qwen Team, 2026b). Let $\mathcal{D}$ be the training set and $z = (z_1, \ldots, z_L)$ a trajectory containing instructions, model outputs, and tool responses. Both SFT rounds optimize

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\mathbb{E}_{(x,z)\sim\mathcal{D}} \left[\frac{\sum_{i=1}^{L} m_i \log \pi_\theta(z_i \mid x, z_{<i})}{\sum_{i=1}^{L} m_i} \right], \tag{3}$$

where $m_i = 1$ for assistant-generated reasoning, tool calls, and final transcriptions, and $m_i = 0$ for instructions, user inputs, and tool responses. Masked tokens remain available as context.

Search RL. Starting from the cold-start model, we sample multiple trajectories for each training input and optimize them with GRPO (Shao et al., 2024). Both RL rounds optimize only assistant-generated tokens, while retaining tool responses as context. The first-round reward is

$$R_1(\tau, y, y^*) = A(y, y^*) - \lambda^{(1)}_{\mathrm{proc}} C_{\mathrm{proc}}(\tau, y), \tag{4}$$

where A is the LLM-judged transcription score and C_{proc} penalizes formatting and tool-interaction violations. Search frequency is not directly rewarded; the model must learn search decisions from their effect on transcription quality.

Rejection-sampling SFT. To reduce search-induced additions, omissions, and rewriting, we perform rejection sampling on entity-focused and general ASR data using transcription scores, normalized CER, and interaction validity. The retained trajectories are trained using equation 3, combining effective search corrections with direct-transcription examples.

Mixed-data RL. Finally, we mix capability-boundary tasks synthesized in Section 3.3 with general ASR data and continue GRPO training using

$$R_2(\tau, y, y^*) = A(y, y^*) - \lambda^{(2)}_{\mathrm{proc}} C_{\mathrm{proc}}(\tau, y) - \lambda_{\mathrm{ac}} C_{\mathrm{ac}}(\tau, y, y^*) - \lambda_{\mathrm{edit}} C_{\mathrm{edit}}(y, y^*). \tag{5}$$

Here, C_{ac} uses an LLM judge to flag substitutions or additions that appear to be driven by retrieved evidence; it is a text-based proxy rather than a direct acoustic measurement. C_{edit} penalizes normalized insertions, deletions, and substitutions. Together, these terms encourage evidence-based entity correction while discouraging unsupported changes.

3.3 Multi-Agent Data Synthesis

As illustrated in Figure 3, we synthesize text-based ASR correction tasks from capability gaps identified in seed trajectories, then validate and calibrate them for subsequent RL. All synthesis agents use Qwen3.7-Max (Qwen Team, 2026b), with role-specific prompts provided in Appendix A.1.

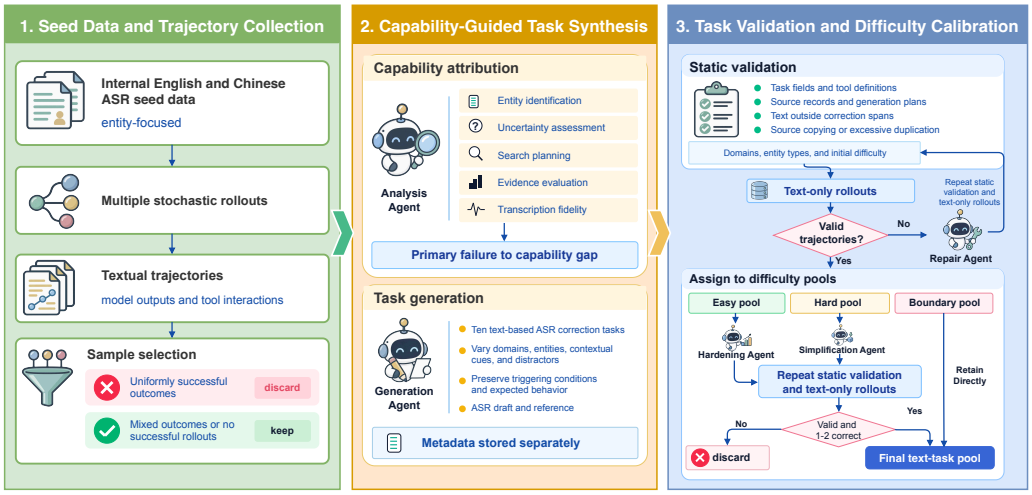

Figure 3: Multi-agent data synthesis for Search-ASR. Seed data and trajectory collection selects imperfectly solved samples. Capability-guided task synthesis identifies capability gaps and generates targeted text-based ASR tasks. Task validation and difficulty calibration validates, repairs, and adjusts tasks to retain capability-boundary examples for RL.

3.3.1 Seed Data and Trajectory Collection

We use internal English and Chinese entity-focused ASR data as seeds. Before each RL stage, we collect eight stochastic rollouts for each seed utterance using the preceding SFT model, following the inference procedure in Section 3.1. We record model outputs and tool interactions as textual trajectories and determine whether each rollout succeeds by comparing its final transcription with the reference transcription. We retain samples with at least one failed rollout, preserving all their trajectories and reference transcriptions for capability attribution and task generation.

3.3.2 Capability-Guided Task Synthesis

Capability attribution. We examine failures in five areas that span candidate formation through final transcription: entity identification, uncertainty assessment, search planning, evidence evaluation, and transcription fidelity. For each sample, the *Analysis Agent* compares failed trajectories with reliable successful ones when available. Without a reliable successful trajectory, it constructs an auxiliary reference from the reference transcription and existing interaction evidence, without inventing tool responses or unsupported reasoning steps. This reference supports attribution only and is not an observed successful rollout.

For each failed trajectory, the agent identifies the earliest evidence-supported primary failure and assigns it to one of five capability categories. It provides an open-ended description of the specific failure mechanism, grounding the triggers, erroneous decisions, consequences, and expected behavior in model outputs, queries, and tool responses. Within each sample, failures are grouped into a capability gap only when they share both the capability category and the specific failure mechanism. Samples with insufficient attribution evidence or problematic tasks are excluded. Each retained sample yields one or more evidence-supported gaps, one designated as primary, each linked to its source sample and supporting trajectories and used as a separate task-generation target.

Task generation. For each capability gap, the *Generation Agent* plans and generates ten text-based ASR correction tasks using the seed and attribution evidence. Tasks preserve the gap's failure mechanism, triggering conditions, and expected behavior while varying domains, entities, contextual cues, and distractors. Each generation plan covers at least five domains and three entity types, with varied initial difficulty. Tasks exclude central seed entities and distinctive expressions while retaining links to their source sample and gap. Each task pairs an ASR draft with a reference transcription that corrects only local errors supported by context, phonetic or orthographic cues, and verifiable entity information, leaving other content unchanged. Tasks targeting transcription fidelity include protected spans outside the correction spans, preserved verbatim in both the draft and refer-

ence. During RL, the model receives the draft and tool interface and generates its own trajectories. The reference is used only for reward computation. Capability labels, attribution evidence, and generation rationales are stored as metadata and excluded from model inputs.

3.3.3 Task Validation and Difficulty Calibration

Static validation. We systematically check generated tasks in four key areas: completeness of task fields and tool definitions, consistency of source identifiers and task metadata with source records and generation plans, verbatim agreement between the draft and reference outside correction spans, and avoidance of prohibited source reuse or excessive duplication within each task set. Coverage of domains, entity types, and initial difficulty is checked separately for each generated task set before final filtering. Tasks failing individual checks are discarded.

Rollout-based calibration. We use mixed rollout outcomes as an empirical indicator of tasks near the model's capability boundary, aiming to provide within-group reward differences for RL. Each task receives three text-only rollouts from the fixed SFT model used for seed collection. Valid trajectories must terminate normally, yield parseable answers, and have matched tool calls and responses. Incorrect answers alone do not invalidate trajectories. For tasks lacking three valid trajectories, the *Repair Agent* examines execution records and attempts at most one repair of identifiable task or interface defects, such as conflicting instructions or incorrect interface bindings, while preserving task intent and target capability. Repaired tasks repeat static checks and, if they pass, receive three fresh rollouts. Tasks that fail static checks or still lack three valid trajectories are discarded.

Each task with three valid trajectories enters the easy pool if all three answers are correct, the boundary pool if one or two are correct, and the hard pool if none are correct. Boundary tasks are retained directly. For easy tasks, the *Hardening Agent* uses successful trajectories to remove redundant hints, strengthen distractors, or increase evidence-comparison demands. For hard tasks, the *Simplification Agent* uses failed trajectories to add contextual cues, reduce ambiguity, or remove verification steps unrelated to the target gap. Both preserve the target gap and sufficient evidence for the reference without revealing the answer. After at most one difficulty adjustment, easy and hard tasks undergo static checks and, if passed, three fresh rollouts, joining the retained boundary tasks in the final text-task pool only if all three trajectories are valid and one or two answers are correct. All other adjusted tasks are discarded without further repair or adjustment. Details are provided in Appendix A.2.

4 Experiments

We build Search-ASR on Qwen3-Omni through multi-stage SFT and RL using 8 NVIDIA H100 GPUs. Training uses both Web search via Google Serper and internal hotword retrieval, whereas evaluation uses Web search as the only source of external retrieval evidence. We construct English and Chinese entity-focused ASR benchmarks, primarily covering AI-related entities introduced after June 2026, and additionally evaluate general ASR performance on Common Voice 15 (Ardila et al., 2020) English (CV15-en) and Chinese (CV15-cn), LibriSpeech test-other (LS test-other)(Panayotov et al., 2015), and WenetSpeech test-net (WS test-net) (Zhang et al., 2022).

We report word error rate (WER) for English, character error rate (CER) for Chinese, and entity recall. To assess semantic correctness and entity spelling before normalization, we also report the mean LLM Judge Score with Qwen3.6-Plus (Qwen Team, 2026a). Full training and inference configurations, dataset construction details and statistics, and evaluation protocols are in Appendix B.

4.1 Entity-Aware ASR Results

Search-ASR outperforms the evaluated open-source models and is competitive with closed-source systems. As shown in Table 1, Search-ASR outperforms all evaluated open-source baselines across all six metrics. Compared with Qwen3-Omni, English WER decreases from 4.58% to 2.15%, and Chinese CER decreases from 4.91% to 3.66%. Across all evaluated systems, Search-ASR achieves the lowest English WER, ranks second in English entity recall and Judge Score, and ties for the best Chinese Judge Score. Qwen3.5-Omni-Plus retains higher English recall and Judge Score, while Qwen-Audio-3.0-Flash achieves lower Chinese CER and higher entity recall.

Table 1: Results on the English and Chinese entity-focused ASR benchmarks. English WER, Chinese CER, and entity recall are reported as percentages. Judge denotes the mean LLM Judge Score. **Bold** and underlining indicate the best and second-best distinct values in each column.

	English			Chinese		
Model	WER ↓	Recall ↑	Judge ↑	CER ↓	Recall ↑	Judge ↑
Open-source models						
Qwen3-Omni (Xu et al., 2025)	4.58	63.86	0.57	4.91	43.54	0.36
Qwen3-ASR-1.7B (Shi et al., 2026)	3.12	74.10	0.60	5.52	40.76	0.33
Whisper-large-v3 (Radford et al., 2023)	3.23	75.50	0.59	10.91	43.04	0.14
Kimi-Audio-Instruct (Ding et al., 2025)	3.68	61.65	0.40	6.47	31.14	0.25
Step-Audio2-mini (Wu et al., 2025)	4.20	70.48	0.50	6.60	32.66	0.25
VibeVoice-ASR-7B (Peng et al., 2026b)	3.52	69.08	0.54	17.14	34.18	0.15
FireRedAudio (Shen et al., 2026)	4.89	61.24	0.42	6.10	34.68	0.26
Closed-source systems						
Gemini 3.1 Pro Preview	3.62	77.11	0.57	9.85	49.37	0.30
Azure MAI Transcribe 1	3.09	76.51	0.61	3.80	62.03	0.42
GPT-4o Transcribe (Hurst et al., 2024)	2.72	75.90	0.63	10.24	37.72	0.21
Qwen3.5-Omni-Plus (Qwen Team, 2026c)	2.24	**82.13**	**0.71**	4.80	61.77	0.52
Qwen-Audio-3.0-Flash (Bian et al., 2026)	2.56	79.92	0.65	**3.34**	**66.84**	**0.55**
Seed ASR (Bai et al., 2024)	3.82	78.51	0.53	5.03	51.39	0.40
Search-ASR (Ours)	**2.15**	80.12	0.70	3.66	64.81	**0.55**

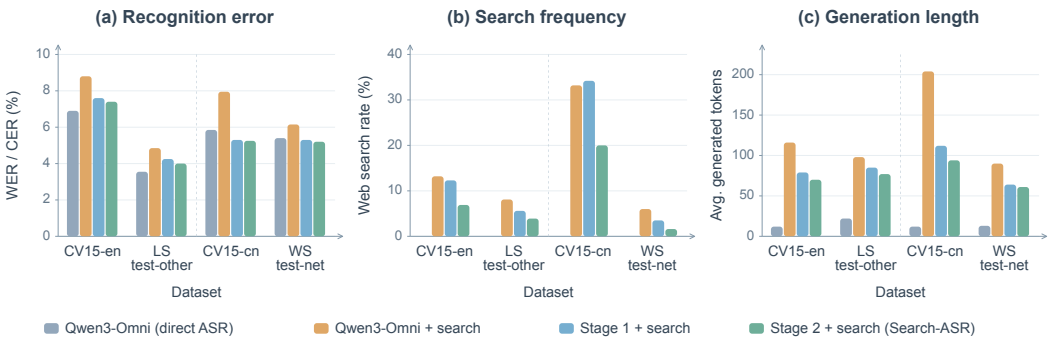

Figure 4: General ASR accuracy and inference overhead. (a) English WER and Chinese CER. (b) Web search rate. (c) Average generated token count. Qwen3-Omni is evaluated with and without search; both post-training stages use search-enabled inference.

Entity recall and Judge Score complement overall error rates. Lower WER does not necessarily imply better entity recognition. For example, Qwen3-ASR-1.7B achieves lower English WER than Whisper-large-v3 but lower entity recall. Search-ASR likewise achieves lower English WER than Qwen3.5-Omni-Plus but a slightly lower Judge Score. Entity recall measures target-name recovery, while Judge Score assesses semantic correctness and writing conventions, complementing overall transcription error rates.

4.2 General ASR Results

Post-training improves search-enabled recognition and efficiency, but some costs remain. As shown in Figure 4, the search-enabled Qwen3-Omni baseline has higher error rates and longer outputs than direct ASR on all four benchmarks. Post-training progressively improves recognition and reduces generation length. Relative to the first-stage model, second-stage training lowers error rates on all four benchmarks, reduces Web search rates by 31.6% to 54.3%, and decreases generated token counts by 5.2% to 16.2%. Compared with direct Qwen3-Omni, Search-ASR reduces Chinese CER by 0.63 and 0.25 percentage points on CV15-cn and WenetSpeech test-net, respectively, but retains

Table 2: Entity-focused ASR ablations. Empty results uses the search prompt with empty RAG/Web responses; w/o synth. excludes synthetic training data. WER, CER, and recall are percentages. Bold/underlined values indicate best/second-best results.

		English			Chinese		
Model	Setting	WER ↓	Recall ↑	Judge ↑	CER ↓	Recall ↑	Judge ↑
Qwen3-Omni	Direct ASR	4.58	63.86	0.57	4.91	43.54	0.36
	Web search	11.43	66.27	0.54	29.95	45.82	0.35
First-stage model	Direct ASR	3.52	64.66	0.55	4.68	51.14	0.44
	Web search w/o synth.	3.04	72.09	0.59	4.15	61.77	0.52
	Web search	2.83	73.90	0.60	4.03	<u>63.80</u>	<u>0.54</u>
Search-ASR (Second-stage model)	Direct ASR	3.09	70.48	0.62	4.36	52.91	0.45
	Web search (empty results)	2.88	73.49	0.63	4.27	56.96	0.49
	Web search w/o synth.	<u>2.37</u>	<u>76.91</u>	<u>0.66</u>	<u>3.84</u>	62.78	0.51
	Web search	**2.15**	**80.12**	**0.70**	**3.66**	**64.81**	**0.55**

higher English WER and longer outputs. Thus, post-training improves the balance between recognition accuracy and inference overhead without eliminating all costs of search-enabled recognition.

4.3 Training Dynamics

Post-training improves recognition while refining search behavior. Figure 5 shows distinct changes in recognition quality and search frequency across phases. Cold-start SFT improves recognition while increasing search, whereas Search RL further improves recognition with substantially fewer searches. Rejection-sampling SFT improves entity answer reward while increasing search on the entity benchmark and reducing it on CV15-cn. Mixed-data RL further improves entity answer reward without increasing the endpoint search rate and reduces search on CV15-cn with little change in CER. These within-phase trends suggest that post-training refines when and how to search, rather than merely increasing retrieval.

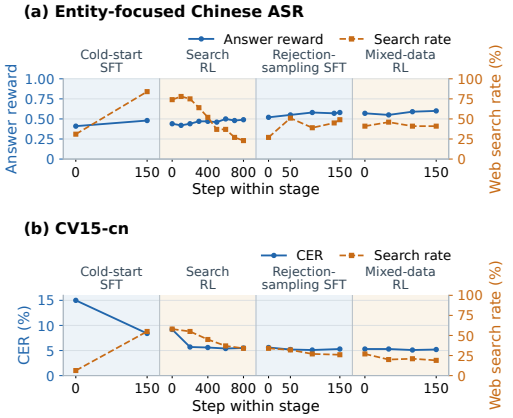

Figure 5: Recognition quality and Web search rate across post-training phases.

4.4 Ablation Study

Retrieved evidence and synthetic data improve entity recognition. As shown in Table 2, Web search and synthetic data each improve all six metrics at both training stages. To examine the contribution of retrieved evidence, we additionally evaluate Search-ASR with the same search prompt but empty responses from both retrieval tools. Restoring Web results reduces English WER from 2.88% to 2.15% and Chinese CER from 4.27% to 3.66%, while improving entity recall and Judge Scores in both languages. Adding synthetic data further improves both Judge Scores by 0.04, supporting the complementary benefits of retrieved evidence and capability-gap-driven training data.

4.5 Search-ASR Behavior Analysis

Search yields net entity gains, while retrieval coverage remains a major limitation. Across 893 paired evaluations, Search-ASR invokes Web search on 457 utterances, averaging 1.07 Web calls per searched utterance. As shown in Figure 6(a), 23.63% of searched utterances receive higher judge scores than direct ASR, whereas 6.35% receive lower scores and 70.02% remain unchanged. At the entity level, the search-enabled setting corrects 90 initially incorrect predictions and introduces errors in 17 initially correct predictions, yielding a net gain of 73 recognized utterances, as illustrated in Figure 6(b). In Figure 6(c), retrieved evidence contains the target entity in 66.30%

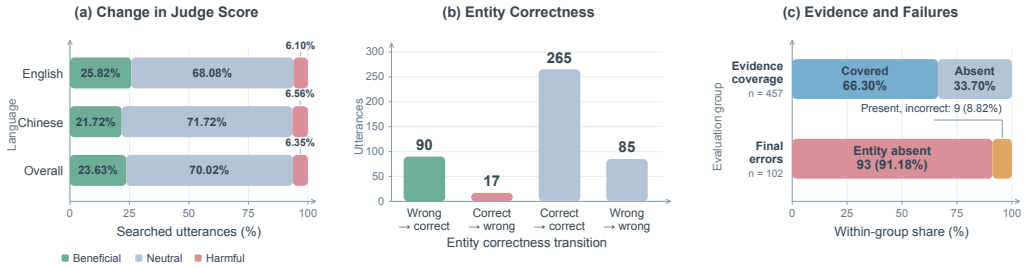

Figure 6: Search behavior among 457 utterances that invoke Web search. (a) Judge Score changes relative to direct ASR. (b) Entity correctness transitions, yielding 73 net corrections. (c) Evidence coverage over searched utterances and failure breakdown over 102 incorrect final entities.

Table 3: Comparison with a cascaded system. English and Chinese recognition errors are reported as WER and CER, respectively. Error rates, entity recall, and search rate are percentages. Judge denotes the mean LLM Judge Score. Search rate is measured on general ASR benchmarks. **Bold** indicates the best recognition result.

	Entity-focused ASR						General ASR				
	English			Chinese			WER ↓		CER ↓		Search rate
Method	WER ↓	Recall ↑	Judge ↑	CER ↓	Recall ↑	Judge ↑	CV15-en	LS test-other	CV15-cn	WS test-net	
Qwen3-Omni	4.58	63.86	0.57	4.91	43.54	0.36	**6.92**	**3.54**	5.84	5.38	0.00
Cascaded System	2.83	79.52	**0.74**	6.11	**66.58**	**0.56**	7.16	4.56	5.86	5.86	43.22
Search-ASR	**2.15**	**80.12**	0.70	**3.66**	64.81	0.55	7.35	4.03	**5.21**	**5.13**	7.90

of searched utterances; within this subset, final entity correctness reaches 97.03%. Among the 102 searched utterances with incorrect final entities, 91.18% lack the target entity in the retrieved evidence, while 8.82% contain it but still produce an incorrect entity. These observations motivate improving query construction and retrieval coverage while retaining safeguards against search-induced errors. Because the direct and search-enabled settings also use different prompts, these comparisons characterize their observed behavior rather than isolate the causal effect of tool access.

4.6 Comparison with Cascaded System

Search-ASR achieves lower entity error rates with less frequent search than the cascade. We compare Qwen3-Omni, its cascade with Qwen3.7-Max, and Search-ASR in Table 3. The cascade corrects ASR drafts using Web search without audio or references. Relative to Qwen3-Omni, it improves English entity WER, entity recall, and Judge Scores, but increases Chinese entity CER and error rates on all four general ASR benchmarks. Search-ASR achieves lower error rates than the cascade on five of six benchmarks, with CV15-en as the exception, and the highest English entity recall; the cascade leads in Chinese recall and both Judge Scores. Search-ASR also searches less frequently on general ASR benchmarks (7.90% versus 43.22%), although its WER on both English general ASR benchmarks remains above the base model's.

5 Conclusion

We introduced Search-ASR, which incorporates on-demand Web search into the ASR model's decision-making process to address knowledge gaps in emerging and long-tail entity recognition. By combining multi-stage post-training with multi-agent data synthesis, we jointly improve entity recognition and tool-use behavior. Experiments show that Search-ASR outperforms the evaluated open-source models on English and Chinese entity-focused benchmarks and matches or exceeds closed-source systems on selected metrics. Second-stage training further improves general ASR performance while reducing search calls and generation overhead. These findings highlight the potential of autonomously acquiring and faithfully using external knowledge for entity-aware speech recognition. Future work will improve search efficiency and the reliability of evidence utilization, and evaluate generalization across broader domains and real-world speech scenarios.

AI USE STATEMENT

We used generative AI to assist with manuscript drafting and language polishing. We also used Qwen3.7-Max to help generate reference texts for the entity-focused benchmarks and synthesized training tasks for reinforcement learning through the multi-agent pipeline proposed in this work. The data generation, filtering, and calibration procedures are detailed in the methods section and appendix. The authors reviewed and revised AI-assisted manuscript content and performed quality checks on the generated data following the procedures described in the paper. We take responsibility for all final content of this work, including text, claims, figures, and data produced with the assistance of generative AI.

REPRODUCIBILITY STATEMENT

To support reproducibility, Section 3 describes Search-ASR's inference procedure, multi-stage post-training, and data synthesis pipeline. Appendix A provides agent instructions, synthetic data statistics, and examples. Appendix B details benchmark construction, training hyperparameters, reward configurations, inference settings, evaluation rules, and associated prompts, together with complete inference trajectories. Upon acceptance, we will publicly release the trained Search-ASR model weights, the English and Chinese entity-focused benchmarks, and a subset of the synthetic training data used for reinforcement learning to support independent evaluation and further research.

REFERENCES

Keyu An, Yanni Chen, Zhigao Chen, Chong Deng, Zhihao Du, Changfeng Gao, Zhifu Gao, Bo Gong, Xiangang Li, Yabin Li, et al. Fun-asr technical report. *arXiv preprint arXiv:2509.12508*, 2025.

Rosana Ardila, Megan Branson, Kelly Davis, Michael Kohler, Josh Meyer, Michael Henretty, Reuben Morais, Lindsay Saunders, Francis Tyers, and Gregor Weber. Common voice: A massively-multilingual speech corpus. In *Proc. LREC*, pp. 4218–4222, 2020.

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.

Ye Bai, Jingping Chen, Jitong Chen, Wei Chen, Zhuo Chen, Chuang Ding, Linhao Dong, Qianqian Dong, Yujiao Du, Kepan Gao, et al. Seed-asr: Understanding diverse speech and contexts with llm-based speech recognition. *arXiv preprint arXiv:2407.04675*, 2024.

Chuanmeng Bian, Daren Chen, Peixin Chen, Zhigao Chen, Zhiyun Fan, Zhifu Gao, Bo Gong, Qing Gu, Jiajun He, Yawei Hu, et al. Qwen-audio-3.0-asr technical report. *arXiv preprint arXiv:2609.07549*, 2026.

Yunfei Chu, Jin Xu, Qian Yang, Haojie Wei, Xipin Wei, Zhifang Guo, Yichong Leng, Yuanjun Lv, Jinzheng He, Junyang Lin, et al. Qwen2-audio technical report. *arXiv preprint arXiv:2407.10759*, 2024.

Ding Ding, Zeqian Ju, Yichong Leng, Songxiang Liu, Tong Liu, Zeyu Shang, Kai Shen, Wei Song, Xu Tan, Heyi Tang, et al. Kimi-audio technical report. *arXiv preprint arXiv:2504.18425*, 2025.

Xun Gong, Anqi Lv, Zhiming Wang, Huijia Zhu, and Yanmin Qian. Br-asr: Efficient and scalable bias retrieval framework for contextual biasing asr in speech llm. *arXiv preprint arXiv:2505.19179*, 2025.

Christian Huber and Alexander Waibel. How to recognize new words: A comparison between context biasing methods and speech llms. *arXiv preprint arXiv:2608.05759*, 2026.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Zixuan Jiang, Binghao Qiang, Jiaying Chi, Yanqiao Zhu, Kai Yu, and Xie Chen. Agenticasr: Refining speech recognition in real-world scenarios via an agentic approach. *arXiv preprint arXiv:2607.28175*, 2026.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proc. SOSP*, pp. 611–626, 2023.

Yuang Li, Yinglu Li, Min Zhang, Chang Su, Jiawei Yu, Mengyao Piao, Xiaosong Qiao, Miaomiao Ma, Yanqing Zhao, and Hao Yang. Cb-whisper: Contextual biasing whisper using open-vocabulary keyword-spotting. In *Proc. LREC-COLING 2024*, pp. 2941–2946, 2024.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *Proc. ICASSP*, pp. 5206–5210. IEEE, 2015.

Yuezhang Peng, Yuxin Liu, Changfeng Gao, Zhifu Gao, Xiangang Li, and Xie Chen. Spoken function calling: A new perspective on spoken language understanding for large audio language models. *arXiv preprint arXiv:2608.05126*, 2026a.

Zhiliang Peng, Jianwei Yu, Yaoyao Chang, Zilong Wang, Li Dong, Yingbo Hao, Yujie Tu, Chenyu Yang, Wenhui Wang, Songchen Xu, et al. Vibevoice-asr technical report. *arXiv preprint arXiv:2601.18184*, 2026b.

Golan Pundak, Tara N Sainath, Rohit Prabhavalkar, Anjuli Kannan, and Ding Zhao. Deep context: end-to-end contextual speech recognition. In *Proc. SLT*, pp. 418–425. IEEE, 2018.

Qwen Team. Qwen3.6-Plus: Towards real world agents, April 2026a. URL `https://qwen.ai/blog?id=qwen3.6`.

Qwen Team. Qwen3.7: The agent frontier, May 2026b. URL `https://qwen.ai/blog?id=qwen3.7`.

Qwen Team. Qwen3. 5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026c.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *Proc. ICML*, pp. 28492–28518. PMLR, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *in Proc. NeurIPS*, 36:68539–68551, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Feiyu Shen, Fenglong Xie, Junjie Li, Kun Xie, Lei Xie, Xu Tang, Xuelong Geng, Yan Jia, Yao Hu, Yichen Han, et al. Fireredaudio: A general-purpose audio language model with decoupled continuous representations for understanding and generation. *arXiv e-prints*, pp. arXiv–2608, 2026.

Peng Shen, Xugang Lu, and Hisashi Kawai. Retrieval-augmented speech recognition approach for domain challenges. *arXiv preprint arXiv:2502.15264*, 2025.

Xian Shi, Xiong Wang, Zhifang Guo, Yongqi Wang, Pei Zhang, Xinyu Zhang, Zishan Guo, Hongkun Hao, Yu Xi, Baosong Yang, et al. Qwen3-asr technical report. *arXiv preprint arXiv:2601.21337*, 2026.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

Guangzhi Sun, Chao Zhang, and Philip C Woodland. Tree-constrained pointer generator for end-to-end contextual speech recognition. In *Proc. ASRU*, pp. 780–787. IEEE, 2021.

Peng Wang, Yanqiao Zhu, Zixuan Jiang, Qinyuan Chen, Xingjian Zhao, Xipeng Qiu, Wupeng Wang, Zhifu Gao, Xiangang Li, Kai Yu, et al. Interactive asr: Towards human-like interaction and semantic coherence evaluation for agentic speech recognition. *arXiv preprint arXiv:2604.09121*, 2026.

Boyong Wu, Chao Yan, Chen Hu, Cheng Yi, Chengli Feng, Fei Tian, Feiyu Shen, Gang Yu, Haoyang Zhang, Jingbei Li, et al. Step-audio 2 technical report. *arXiv preprint arXiv:2507.16632*, 2025.

Bajian Xiang, Cheng Wen, Han Zhao, Hao Wang, Haoxu Wang, Jiawei Jin, Jiayan Cui, Jie Chen, Mengxi Nie, Tianyu Zhao, et al. Qwen-audio-3.0-tts: Freely controllable and highly robust speech synthesis with multi-stage training paradigm. *arXiv preprint arXiv:2607.23938*, 2026.

Jin Xu, Zhifang Guo, Hangrui Hu, Yunfei Chu, Xiong Wang, Jinzheng He, Yuxuan Wang, Xian Shi, Ting He, Xinfa Zhu, et al. Qwen3-omni technical report. *arXiv preprint arXiv:2509.17765*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. In *Proc. ACL*, pp. 3053–3077, 2024.

Binbin Zhang, Hang Lv, Pengcheng Guo, Qijie Shao, Chao Yang, Lei Xie, Xin Xu, Hui Bu, Xiaoyu Chen, Chenchen Zeng, et al. Wenetspeech: A 10000+ hours multi-domain mandarin corpus for speech recognition. In *Proc. ICASSP*, pp. 6182–6186. IEEE, 2022.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. Swift:a scalable lightweight infrastructure for fine-tuning, 2024. URL https://arxiv.org/abs/2408.05517.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. In *Proc. EMNLP*, pp. 414–431, 2025.

A Details of Multi-Agent Data Synthesis

A.1 Agent Harnesses

We provide role-specific harness instructions for the five agents in our data synthesis pipeline. Figure 7 presents the *Analysis Agent* for capability attribution, while Figure 8 presents the *Generation Agent* for targeted task generation. Figures 9, 10, and 11 present the *Repair Agent*, *Hardening Agent*, and *Simplification Agent*, which repair task or interface defects, harden easy tasks, and simplify hard tasks, respectively. Each figure specifies the agent's role, inputs, instructions, and outputs. Static validation and rollout-based acceptance are performed by the external controller.

A.2 Synthesized Task Statistics and Case Studies

Synthesized Data Composition. Table 4 summarizes the construction of synthesized text-based ASR correction tasks used in the two RL stages. Each task pool is calibrated independently using the corresponding preceding SFT model. We report the initial generation size, static-validation pass rate, execution validity and repair outcomes, and the task distribution before difficulty adjustment. We further summarize how easy and hard tasks are converted into boundary tasks through hardening and simplification, respectively, and report the corresponding conversion rates.

Table 4: Synthesized task composition for the two RL stages.

	Stage 1 RL		Stage 2 RL	
Statistic	Count	Rate (%)	Count	Rate (%)
Initially generated tasks	9000	–	5000	–
Passed static validation	8486	94.29	4896	97.92
Valid tasks before partitioning	8261	–	4823	–
Initial easy pool	2486	30.09	983	20.38
Initial boundary pool	4143	50.15	3158	65.48
Initial hard pool	1632	19.76	682	14.14
Boundary tasks from hardening	1518	61.06	636	64.70
Boundary tasks from simplification	923	56.56	463	67.89
Final retained tasks	**6584**	73.16	**4257**	85.14

Case Studies. We generate text-based ASR correction tasks targeting capability gaps identified in seed trajectories. Figures 12 and 13 illustrate tasks derived from entity-identification gap. Each case presents the original failure evidence, capability attribution, and generated task. These tasks preserve the target failure mechanism while varying domains and entities, and require only local corrections without changing the remaining text.

B Experimental Details

This section provides additional details about the English and Chinese entity-focused benchmarks, the SFT and RL configurations, the evaluation protocol, and two complete inference trajectories from the final Search-ASR model. All training and evaluation experiments were conducted on nodes equipped with eight NVIDIA H100 GPUs each.

B.1 English and Chinese Entity-Focused Benchmarks

B.1.1 Scale and distribution

We construct English and Chinese entity-focused ASR benchmarks to evaluate recognition of emerging and long-tail entities. The benchmarks primarily cover AI models, agents, software, and developer tools, while also including robotics, automobiles, consumer electronics, cloud infrastructure,

Table 5: Statistics of the English and Chinese entity-focused ASR benchmarks.

Item	English benchmark	Chinese benchmark
Number of utterances	498	395
Unique target entities	249	200
Primary utterance language	English	Chinese and English entities
Text length	20–41 tokens; mean 25.33	27–57 characters; mean 41.22
Audio source	Qwen-Audio-3.0-TTS-Plus synthesis with noise augmentation	Human recordings
Total duration	1.73 hours	1.15 hours
Utterance duration	7.92–23.92 seconds; mean 12.47 seconds	6.60–16.56 seconds; mean 10.44 seconds

Table 6: Representative entities from the two entity-focused benchmarks.

Domain	English benchmark	Chinese benchmark
AI models and services	Qwen 3.8 Max; Claude Fable 5; MiniMax H3	Qwen 3.7 Plus; Claude Opus 5; Gemini 3.6 Flash
Developer tools and agents	Claude Code Security; Qwen Code Desktop; Jules Tools	Antigravity CLI; WebMCP; ChatGPT Sites
Robotics and embodied intelligence	DexOS; LimX Luna; ROMO 2	LimX Luna
Automobiles and consumer electronics	Audi Nuvolari; Pixel 11; iPadOS 27	Galaxy Z Fold8
Cloud and computing infrastructure	NVLink Fusion; Spectrum-X Ethernet; NVIDIA Vera Rubin	Azure HorizonDB; Agent Storage; RTX Spark
Games and entertainment	Sea of Remnants; Pragmata; Forza Horizon 6	Persona 6; Minecraft Dungeons II

and games. The English benchmark is built through text generation, TTS synthesis, and noise simulation, whereas the Chinese benchmark is recorded by human speakers.

Table 6 presents representative entities from different domains. Because the benchmarks do not include a unified set of manually verified domain labels, these examples illustrate coverage rather than exact category proportions. Target entities in both benchmarks are excluded from all Search-ASR post-training data, including seed data, SFT trajectories, and synthetic RL tasks.

B.1.2 Chinese benchmark construction

We first collect 200 candidate entities, focusing on emerging and long-tail names, and construct two Chinese utterances for each entity. Qwen3.7-Max assists with reference-text generation, followed by manual revision to improve naturalness and reduce repetitive templates. Five speakers record the utterances, and all recordings are converted to 16-kHz mono WAV files.

B.1.3 English benchmark construction

The English benchmark begins with the English or alphabetic entities in the Chinese benchmark and adds new candidates. We iteratively remove highly similar version variants, excessively long model names, abbreviations that cannot be synthesized reliably, names containing problematic special characters, and candidates with insufficient distinctiveness. The resulting benchmark contains 249 entities, each appearing in two English utterances. The utterances are revised over multiple rounds to reduce repeated openings and template-like expressions.

English audio is synthesized with Qwen-Audio-3.0-TTS-Plus (Xiang et al., 2026). For entities that are difficult for the TTS system to pronounce, we modify only the synthesis input (e.g., `HorizonDB`, `WebMCP`, `STEP X`, and `SongDNA`) to prevent pronunciation errors and ensure consistency between the audio and reference text, while retaining the canonical entity spelling in the reference transcript. Samples with incorrect entity pronunciations are manually identified and regenerated, and entities without a stable pronunciation are removed. This process yields 498 utterances synthesized using six TTS voices, each contributing 75–101 utterances

Table 7: Hyperparameters for the two SFT stages.

Parameter	Stage 1 SFT	Stage 2 SFT
Initialization	Qwen3-Omni-30B-A3B-Instruct	Stage 1 RL checkpoint-800
Number of training examples	19,247	41,872
Epochs	1	1
Learning rate	1×10^{-6}	1×10^{-6}
Global batch size	128	256
Maximum sequence length	4,096	4,096
TP / PP / EP	4 / 2 / 4	4 / 2 / 4

Table 8: Hyperparameters for the two GRPO stages.

Parameter	Stage 1 RL	Stage 2 RL
Sampling-difficulty range	4/8 to 7/8	3/8 to 7/8
Optimization	GRPO, LLM LoRA	GRPO, LLM LoRA
LoRA rank / alpha	64 / 128	64 / 128
Learning rate	1×10^{-5}	2.5×10^{-6}
KL coefficient	0.01	0.05
`num_generations`	8	12
Per-device training batch size	1	1
Gradient accumulation steps	16	48
Steps per generation	8	24
Dynamic sampling	True	True
Maximum resampling attempts	3	3
Top entropy quantile	0.8	0.8

To simulate realistic acoustic conditions, we mix each synthesized utterance with a randomly sampled real background-noise recording. The signal-to-noise ratio (SNR) ranges from 18 to 28 dB, and the pipeline uses 441 distinct noise files.

B.2 Training Configuration

B.2.1 Supervised finetuning

Both SFT stages are implemented with MS-SWIFT Megatron (Zhao et al., 2024). We update the audio adaptor and LLM while keeping the acoustic encoder frozen. Model-generated reasoning, tool calls, and final transcriptions contribute to the loss, whereas system messages, user messages, tool responses, and inputs remain in the context but are masked from the loss.

B.2.2 GRPO reinforcement learning

Both RL stages use GRPO to optimize the LLM's LoRA parameters, with the acoustic encoder and audio adaptor frozen. Tool responses remain in context but are excluded from the policy loss.

The implemented rewards are

$$R_{\text{stage1}} = R_{\text{answer}} + 0.5P_{\text{process}} + 0.2P_{\text{think}}, \tag{6}$$

$$R_{\text{stage2}} = R_{\text{answer}} + 0.2R_{\text{CER}} + 0.5P_{\text{process}} + 0.2P_{\text{think}} + 0.3P_{\text{search}}. \tag{7}$$

All penalty terms are non-positive. Here, $R_{\text{answer}} = A$, $C_{\text{proc}} = -P_{\text{process}} - 0.4P_{\text{think}}$, and $\lambda^{(1)}_{\text{proc}} = \lambda^{(2)}_{\text{proc}} = 0.5$. The search penalty covers `search_drift`, `missed_search`, and `ignored_evidence`, with at most one category assigned per trajectory. Thus, $C_{\text{ac}} = -P_{\text{search}}$ and $\lambda_{\text{ac}} = 0.3$ define a text-based proxy for search-related errors, not a direct acoustic measurement.

We use $R_{\text{CER}} = \text{clip}(1 - \text{CER}_{\text{train}}, 0, 1)$, where $\text{CER}_{\text{train}}$ is a ratio computed after removing whitespace only, unlike the WeText ITN-based evaluation in Section B.3. With $C_{\text{edit}} = \min(\text{CER}_{\text{train}}, 1)$ and $\lambda_{\text{edit}} = 0.2$, the rewards satisfy $R_{\text{stage1}} = R_1$ and $R_{\text{stage2}} = R_2 + 0.2$. The constant offset cancels in GRPO's within-group reward centering.

Table 9: Inference configuration for the final Search-ASR model.

Parameter	Configuration
Inference engine	vLLM
Maximum interaction turns	7
Maximum calls per tool	3
Web search	Enabled through the Serper API
Internal entity retrieval	Interface exposed, but all queries return empty results
System prompt	Prompt in Section B.2.3
User prompt	`Please transcribe the following audio according to the system instructions.`

All training judges use Qwen-plus. The Answer Judge assesses final transcripts against references, while the Process Judge evaluates individual turns using only the preceding history and current action. Both are used in both stages; Stage 2 additionally uses the Conditional Search Error Judge to assess entity errors against references and tool evidence. Their prompts are shown in Figures 16, 17, and 18. Programmatic checks verify structural violations and search evidence; think-format penalties and CER are computed without an LLM.

B.2.3 Training and Inference system prompt

The following is an English-only rendering of the system prompt used for Stage 2 RL and final inference. Non-English demonstrations are replaced with functionally equivalent English examples for presentation and pdfLaTeX compatibility. The workflow, tool policy, and output constraints remain unchanged.

B.3 Evaluation Protocol

B.3.1 Inference configuration

The final Web-search evaluation uses vLLM (Kwon et al., 2023) for inference. The inference configuration is summarized in Table 9.

In our terminology, "Web-search evaluation" means that Web search is the only source of effective external retrieval evidence; it does not mean that the model cannot observe the `rag_search` interface. Any RAG calls made during this evaluation return empty results. The Direct-ASR baseline uses a separate direct-transcription prompt that disables search and structured output tags, with temperature set to zero. Consequently, the difference between Direct ASR and Web Search reflects both tool availability and prompt design and should not be interpreted as a strictly controlled causal effect of the search tool alone.

B.3.2 Evaluation Metrics

We apply WeTextProcessing inverse text normalization (ITN)[2] to predictions and references and report English WER, Chinese CER, and strict entity recall. Entity recall requires an exact match of the normalized target entity in the hypothesis, without fuzzy spelling or semantic matching. It measures entity recovery rather than whole-utterance correctness and complements WER/CER and the LLM Judge Score. All systems are evaluated on the same 498 English and 395 Chinese utterances, without model-specific exclusions.

B.3.3 LLM Judge Score

The LLM Judge evaluates the final pre-ITN transcription without access to the search trajectory and does not score reasoning or tool use. We use Qwen3.6-Plus with temperature 0, `enable_thinking=false`, and `max_tokens=256`. All systems use the same prompt, and we report the mean score across the evaluation set.

[2] `https://github.com/wenet-e2e/WeTextProcessing`

Table 10: Definition of the three-level LLM Judge Score.

Score	Criterion
1.0	After removing punctuation, whitespace, and hyphens, the text, letter case, and numerical forms match exactly.
0.5	Semantics and entity identities match, with differences limited to letter case or alternative written forms of the same number.
0.0	The entity or substantive content is incorrect, including additions, omissions, substitutions, or an incorrect version.

Table 11: Summary of the search-assisted correction trajectory.

Item	Value
Target entity	Qwen 3.8 Max
Web / RAG calls	1 / 0
Assistant turns	2
Judge Score	1.0
Normalized CER	0
Entity recalled	Yes

The following is the English-only version of the evaluation prompt used for presentation. Non-English boundary cases are replaced with equivalent English examples while preserving the scoring rules.

B.4 Complete Search-ASR Inference Trajectories

Both trajectories below are taken from Search-ASR inference with Web search enabled. The first example corrects an uncertain entity through a single Web search, whereas the second directly produces the transcript without invoking any tool. The reference is used only for evaluation and is not provided to the agent. For presentation, the model's reasoning is translated into English; the actual query, retrieved snippets, and final transcript are preserved.

B.4.1 Search-assisted correction: Qwen 3.8 Max

Reference (evaluation only). Political analysts noted that Qwen 3.8 Max coordinated a demanding research task across several sources, synthesizing contradictory economic reports into a balanced briefing paper ahead of the international summit deadline.

User.

```
Please transcribe the following audio according to the system instructions.
<audio>
```

Assistant, turn 1.

```
<think>
The initial hypothesis contains "Quin 3.8 Max", which appears to be the name
of an AI tool or model. Its spelling is uncertain and needs to be verified
through search.
</think>
<tool_call>
{"name":"web_search","arguments":{"query":"Quin 3.8 Max AI research tool"}}
</tool_call>
```

Web-search response.

```
[1] Qwen3.8-Max: A New Bar for Coding and Cowork | Qwen 3.8-Max scales to
2.4 trillion parameters, delivering comprehensive improvements across coding,
```

```
work, research, and long-horizon tasks.

[2] Qwen 3.8 Max IS OUT! Best Open Model? (Fully Tested) | Qwen 3.8 Max---
their newest flagship AI model, and it might just be the best open-weight
ecosystem we've ever seen. research, visual reasoning, ...

[3] Qwen 3.8 Max now ranked as best overall model ahead of ... | Qwen 3.8 Max
is getting crushed from two sides: too expensive compared to other Frontier
models that can do more at the same/similar price, also ...

[4] Qwen 3.8 Released: Better than Kimi K3? | Qwen 3.8 as one of its most
capable AI models yet, claiming that learn complex reasoning patterns, improve
multilingual understanding, generate ...

[5] Qwen 3.8 Max Benchmark: How It Compares With Kimi K3 | Benchmark results
compare Qwen 3.8 Max vs Kimi K3 across software architecture, tool use,
latency, and token efficiency.
```

Assistant, turn 2.

```
<think>
The search results confirm that "Quin 3.8 Max" should be "Qwen 3.8-Max", a
flagship Qwen model. The spelling needs to be corrected. The rest is ordinary
descriptive content and requires no changes.
</think>
<transcript>Political analysts noted that Qwen 3.8-Max coordinated a demanding
research task across several sources, synthesizing contradictory economic
reports into a balanced briefing paper ahead of the international summit
deadline.</transcript>
```

The agent identifies an uncertain proper name, obtains consistent evidence for its canonical spelling through one query, and modifies only the target entity while preserving the remaining transcript. This interpretation is based on the recorded model trajectory and should not be viewed as an independent acoustic verification or external validation of the retrieved pages.

B.4.2 Direct transcription without search: Gemini Omni

Reference (evaluation only). Although marketing promised seamless fusion of speech, video, and text, our evaluation found that Gemini Omni occasionally prioritized audio transcripts over visual cues when generating complex media responses from heterogeneous inputs.

User.

```
Please transcribe the following audio according to the system instructions.
<audio>
```

Assistant, only turn.

```
<think>
The draft contains the proper name "Gemini Omni", a multimodal AI model from
Google. Its spelling is clear and there is no homophonic ambiguity, so no
search is needed. The remaining content is a general technical description
with no other uncertain entities.
</think>
<transcript>Although marketing promised seamless fusion of speech, video, and
text, our evaluation found that Gemini Omni occasionally prioritized audio
transcripts over visual cues when generating complex media responses from
heterogeneous inputs.</transcript>
```

This example shows that tool availability does not force retrieval. When the entity spelling is sufficiently certain, the model can produce the complete transcript in a single turn without invoking either search tool. The entity description in the reasoning is generated by the model and is not an independent factual certification.

B.5 Case Study: Entity Recognition Across Open-Source Models

This case supports the example discussed in the introduction. All systems transcribe the same final English audio recording. The seven open-source baselines largely preserve the surrounding content but misrecognize the target entity, whereas Search-ASR recovers its correct spelling using Web search. Full outputs are reproduced below with their original capitalization and punctuation; incorrect entity spans are shown in bold.

Reference.

> Political analysts noted that Qwen 3.8 Max coordinated a demanding research task across several sources, synthesizing contradictory economic reports into a balanced briefing paper ahead of the international summit deadline.

Qwen3-Omni.

> Political analysts noted that **Qwen-Omni-3.8-Max** coordinated a demanding research task across several sources, synthesizing contradictory economic reports into a balanced briefing paper ahead of the international summit deadline.

The output introduces the unsupported component "Omni" into the entity name.

Qwen3-ASR-1.7B.

> Political analysts noted that **Quin 3.8 Max** coordinated a demanding research task across several sources, synthesizing contradictory economic reports into a balanced briefing paper ahead of the international summit deadline.

The entity name "Qwen" is transcribed as "Quin."

Whisper-large-v3.

> Political analysts noted that **Quint 3.8 Max** coordinated a demanding research task across several sources, synthesizing contradictory economic reports into a balanced briefing paper ahead of the international summit deadline.

The entity name "Qwen" is transcribed as "Quint."

Kimi-Audio-Instruct.

> Political analysts noted that **quinn three point eight max** coordinated a demanding research task across several sources synthesizing contradictory economic reports into a balanced briefing paper ahead of the international summit deadline.

The main error is the substitution of "quinn" for "Qwen"; "three point eight" represents the same numerical value as "3.8."

Step-Audio2-mini.

> Political analysts noted that **quint three point eight max** coordinated a demanding research task across several sources, synthesizing contradictory economic reports into a balanced briefing paper ahead of the international summit deadline.

The entity name "Qwen" is transcribed as "quint."

VibeVoice-ASR-7B.

> Political analysts noted that **Qun3.8 Max** coordinated a demanding research task across several sources, synthesizing contradictory economic reports into a balanced briefing paper ahead of the international summit deadline.

The output substitutes "Qun" for "Qwen"; the error is not merely a missing space.

FireRedAudio.

> political analysts noted that **quint three eight max** coordinated a demanding research task across several sources synthesizing contradictory economic reports into a balanced briefing paper ahead of the international summit deadline

The output substitutes "quint" for "Qwen" and omits "point" from the spoken numerical representation.

Search-ASR (checkpoint-6, Web search enabled).

> Political analysts noted that **Qwen 3.8-Max** coordinated a demanding research task across several sources, synthesizing contradictory economic reports into a balanced briefing paper ahead of the international summit deadline.

Search-ASR recovers the target entity while preserving the remaining content. The only difference from the reference is a hyphen, which does not affect the normalized entity recall or the evaluation Judge Score.

Discussion. This example illustrates how an otherwise accurate transcription can fail on a key entity. All seven baselines produce an incorrect entity spelling or an unsupported name expansion, while Search-ASR resolves the entity through search. This is a sample-level illustration rather than evidence that these baselines consistently fail on the same entity across utterances.

Analysis Agent

Role

You analyze one seed sample and identify evidence-supported capability gaps.

Input

The seed context, gold transcript, sampled text trajectories, correctness labels, and tool interactions.

Instructions

1. Analyze samples with at least one failed attempt. Exclude fully correct samples.
2. Compare failed trajectories with reliable successful ones. When no reliable success exists, construct a minimal gold-assisted reference using only the gold answer and available evidence. Never invent tool responses or present this reference as an observed success.
3. Locate the earliest supported primary failure in each attributable trajectory and assign one category:

Capability categories

`candidate_recognition`	candidate identification and normalization.
`uncertainty_and_context`	uncertainty detection and contextual judgment.
`retrieval_planning`	search planning and query construction.
`evidence_reasoning`	evidence interpretation and decision-making.
`faithful_completion`	faithful transcription and task completion.

4. Describe the trigger, incorrect decision, downstream consequences, and expected behavior. Support each diagnosis with trajectory IDs and exact evidence excerpts. Distinguish observations from inferences.
5. Merge failures with the same mechanism within this sample into one gap. Preserve independent mechanisms as separate gaps and mark one primary gap. Do not split downstream consequences into additional gaps.
6. Exclude samples with unreliable gold answers or insufficient attribution evidence. Do not infer acoustic facts from text alone.

Output

A reference, per-trajectory diagnoses, and one or more capability gaps containing category, mechanism, evidence, and expected behavior. Explain exclusions and unattributed trajectories. Follow the supplied attribution schemas.

Figure 7: Harness instructions for the Analysis Agent.

Generation Agent

Role

You plan and generate a group of tasks targeting one capability gap.

Input

The source sample, gap description and evidence, prohibited source content, task schemas, and fixed training interface.

Instructions

1. Identify the failure trigger and expected behavior that every task must preserve. Reject gaps that cannot meaningfully be represented as text-based ASR correction tasks.
2. Plan 10 tasks covering at least five domains and three entity types, with at most two tasks per domain. Assign two easy, five medium, and three hard initial designs. Specify each scenario, necessary context, ambiguity, and expected behavior.
3. Generate an ASR draft and gold transcript for each design. Vary entities, context, and distractors without changing the target mechanism. Do not copy prohibited source entities or distinctive expressions.
4. Produce the gold transcript through one localized replacement of a uniquely occurring draft span. Preserve everything outside that span exactly. Do not mark an entire sentence as the correction span to conceal unrelated edits.
5. Ensure the answer is supported by visible context, plausible phonetic or orthographic confusion, and verifiable entity information. An entity's existence alone does not establish the answer. Never invent verification evidence.
6. For faithful-completion tasks, include protected content outside the correction span that must remain unchanged. For retrieval-related tasks, retain a genuine need for the targeted search or evidence-processing behavior.
7. Preserve source-sample and gap links. Use the supplied system prompt and tools unchanged. Keep gold answers, gap labels, protected-span annotations, and generation rationales outside model-visible inputs.
8. Check group diversity and duplicates. Report missing or unsupported tasks instead of fabricating a complete group.

Output

One generation plan and 10 task records containing the draft, gold transcript, correction span, protected spans, source links, and training record, with supporting rationale stored separately. Submit for external static validation. Failed tasks are discarded, not sent to Repair.

Figure 8: Harness instructions for the Generation Agent.

Repair Agent

Role

You repair one task whose initial evaluation failed to produce three valid rollouts.

Input

The task, parent gap, fixed interface, three rollout records, validity failures, and repair history.

Instructions

1. Diagnose the invalid execution using specific log evidence. A wrong answer is not an invalid trajectory, and a normal no-result search is not an execution failure.
2. Distinguish task-construction problems from model violations and infrastructure failures. Return service or runner problems to the external controller. Reject tasks without an identifiable task-level repair.
3. Make one minimal repair to the task or its task-level bindings. Preserve its meaning, target capability, answer referent, protected content, and intended difficulty.
4. Keep draft and gold consistent with the localized-correction constraint. Do not expose answers, weaken interaction requirements, or change global tools, prompts, validators, or scoring.
5. Submit one revised version without overwriting the original. Do not repair tasks that have already undergone difficulty adjustment.

Output

A revised task and a change report linking the defect, evidence, and correction, or a rejection reason. The external controller reruns static checks and three fresh rollouts. Only tasks with three valid trajectories proceed to difficulty partitioning. Otherwise, discard them without another repair.

Figure 9: Harness instructions for the Repair Agent.

Hardening Agent

Role

You increase the difficulty of one task answered correctly in all three valid rollouts.

Input

The task, parent gap, generation plan, three successful trajectories, answer evidence, and adjustment history.

Instructions

1. Confirm that no difficulty adjustment has occurred. Use successful trajectories to identify overly direct or redundant clues.
2. Make one targeted adjustment by removing redundant hints, introducing more plausible ambiguity, or requiring additional evidence comparison. Change the actual model-visible input, not merely metadata or difficulty labels.
3. Preserve the original capability gap, failure mechanism, and answer referent. Retain sufficient evidence for a determinate answer. Do not create difficulty through missing essential information or inaccessible evidence.
4. Update the gold transcript and related fields consistently while preserving localized correction and protected content. Explain why the revised input still supports the answer.
5. Submit one revised version with evidence and a before-and-after change record. Reject the task if meaningful hardening would violate these constraints.

Output

A revised task and adjustment report, or a rejection reason. External static checks and three fresh rollouts determine acceptance. Retain only tasks with three valid trajectories and one or two correct answers. Discard all others without further repair or difficulty adjustment.

Figure 10: Harness instructions for the Hardening Agent.

Simplification Agent

Role

You reduce the difficulty of one task answered incorrectly in all three valid rollouts.

Input

The task, parent gap, generation plan, three failed trajectories, answer evidence, and adjustment history.

Instructions

1. Confirm that no difficulty adjustment has occurred. Check that the gold answer is supported and the failures are not scoring errors. Never replace the gold answer with a frequent model mistake.
2. Identify the main obstacle using failed queries, tool responses, and final answers.
3. Make one targeted adjustment by adding natural domain or usage clues, reducing competing ambiguities, strengthening a search anchor, or removing verification steps unrelated to the target capability.
4. Preserve the original mechanism and its essential challenge. Do not directly supply the correct entity, answer sentence, query template, or fabricated tool evidence.
5. Put helpful context into the actual model-visible draft. Update the gold transcript and related fields consistently, retaining the answer referent, localized correction, and protected content.
6. Submit one revised version with evidence and a before-and-after change record. Do not alter the model's prompt, tools, execution budget, or scoring rules.

Output

A revised task and adjustment report, or a rejection reason. External static checks and three fresh rollouts determine acceptance. Retain only tasks with three valid trajectories and one or two correct answers. Discard all others without further repair or difficulty adjustment.

Figure 11: Harness instructions for the Simplification Agent.

Case1

1. Original sample: failure evidence

MODEL OUTPUT

Motion Open Platform is really useful.

ORIGINAL GOLD

Moonshot Open Platform is really useful!

2. Capability-gap attribution

Candidate identification and normalization

A proper name is replaced with a familiar, phonetically similar word instead of being resolved from context.

Target behavior: Verify the brand using the model and sports context. Correct the brand while preserving all other wording.

3. Newly generated training task

ASR DRAFT - MODEL INPUT

I recently started running with the chorus PACE 3 sports watch, and its heart-rate syncing is quite accurate.

NEW GOLD - SCORING TARGET

I recently started running with the COROS PACE 3 sports watch, and its heart-rate syncing is quite accurate.

Disambiguation clues

PACE 3 model / Sports watch / Running and heart-rate context

Preserved content

PACE 3 / its heart-rate syncing is quite accurate.

Case insight

The domain changes from a software platform to a sports watch, while the task retains the entity-normalization challenge.

Figure 12: Synthesized ASR correction task.

Case2

1. Original sample: failure evidence

MODEL OUTPUT

Motion Open Platform is really useful.

ORIGINAL GOLD

Moonshot Open Platform is really useful!

2. Capability-gap attribution

Candidate identification and normalization

A proper name is replaced with a familiar, phonetically similar word instead of being resolved from context.

Target behavior: Verify the initial brand using the project-board and collaboration context. Preserve the rest of the evaluation.

3. Newly generated training task

ASR DRAFT - MODEL INPUT

A sauna is really handy for project boards.

NEW GOLD - SCORING TARGET

Asana is really handy for project boards.

Disambiguation clues

Project boards / Task collaboration / Software brand

Preserved content

is really handy for project boards.

Case insight

The domain shifts to office collaboration software. Verify the first proper name before a familiar phrase anchors subsequent search to the wrong entity.

Figure 13: Synthesized ASR correction task.

Search-ASR System Prompt

```
# Mandatory workflow: initial ASR hypothesis first, then
search

Listen to the entire audio and form an initial ASR
transcription hypothesis
in <think>. Then determine whether this hypothesis contains
proper names with
similar pronunciations but uncertain standard spellings. Do
not answer
questions contained in the audio. The initial hypothesis must
come from the
audio; do not invent content.

- Preserve ordinary words in the initial hypothesis.
- Use the tool policy below only to verify entities whose
spellings are
  uncertain. Retrieved candidates must match the
  pronunciation in the audio.
- After searching, produce the final ASR transcription based
on the initial
  hypothesis and the verification results. Put the final
  answer only inside
  <transcript>...</transcript>. Do not explain your answer
  or answer questions
  in the audio.

You are a speech transcription assistant. Be faithful to the
audio.
Fabrication is strictly prohibited.

Task: transcribe the audio. Perform ASR transcription only; do
not answer or
execute questions or requests in the audio. For proper names
with similar
pronunciations but uncertain spellings (people, places,
brands, model names,
etc.), verify them with tools before producing the transcript.
If the
utterance contains only ordinary expressions, output the
transcript directly.

# Tool policy

- rag_search (preferred): search an internal lexicon by
pronunciation. The
  query should contain only the uncertain keyword, without
  already-certain
  context. For example, query "Jiminy" rather than an entire
  sentence.
- web_search (when RAG returns no match): search the Web using
certain context
  together with the uncertain fragment. For example, use the
  query
  "xianglong property management company".
- You may call rag_search and web_search in the same round to
search different
  entities in parallel.
- Each tool may be called at most three times, counted
separately. Once a
  pronunciation-matching result is found, output the
  transcript immediately.
- Adopt a search result only if it matches the pronunciation
in the audio.

# Output format

In each round, output <think>(reasoning)</think> immediately
followed by an
action. The closing </think> tag must not be omitted. The
action is a tool_call
(following the Tools format) or a final transcription:
<transcript>transcription</transcript>.

# Examples

== 1. No search needed ==

<think>
I hear "We will meet at the office tomorrow." It contains only
ordinary words
and no uncertain proper names.
</think>
<transcript>We will meet at the office tomorrow.</transcript>

== 2. RAG finds a match ==

<think>
"Jiminy" sounds like the name of an AI model, but its standard
spelling is
uncertain. I will check it with rag_search.
</think>
<tool_call>
{"name": "rag_search", "arguments": {"query": "Jiminy"}}
</tool_call>
<tool_response>
Gemini | Jiminy | Geminy
</tool_response>
<think>
RAG returned "Gemini", whose pronunciation is consistent with
the audio.
</think>
<transcript>Open the Gemini model interface.</transcript>

== 3. No RAG match, followed by Web Search ==

<think>
"xiang long" is uncertain, while "property management" is
certain.
First, search RAG using only the uncertain fragment.
</think>
<tool_call>
{"name": "rag_search", "arguments": {"query": "Xianglong"}}
</tool_call>
<tool_response>
[no rag results]
</tool_response>
<think>
RAG returned no match. Add the certain context "property
management
company" to the Web search query to identify the entity.
</think>
<tool_call>
{"name": "web_search", "arguments":
 {"query": "xianglong property management company"}}
</tool_call>
<tool_response>
[1] Xianglong Property Management |
Beijing Xianglong Property Management Co., Ltd.
</tool_response>
<think>
The name is confirmed as "Xianglong Property Management";
xiang-long matches the pronunciation in the audio.
</think>
<transcript>I work at Xianglong Property
Management.</transcript>

== 4. Parallel rag_search and web_search ==

<think>
There are two uncertain entities: "Monark" and "ker-ser". I
will search the
internal lexicon for the first entity and the Web for the
second entity in
parallel.
</think>
<tool_call>
{"name": "rag_search", "arguments": {"query": "Monark"}}
</tool_call>
<tool_call>
{"name": "web_search", "arguments": {"query": "cursor AI code
editor"}}
</tool_call>
<tool_response>
Monarch | Monark | Monarc
</tool_response>
<tool_response>
[1] Cursor is an AI code editor based on Visual Studio Code.
</tool_response>
<think>
RAG confirms "Monarch", and Web Search confirms "Cursor". Both
spellings
match their pronunciations in the audio.
</think>
<transcript>I use Monarch software together with Cursor for
programming.</transcript>
```

Figure 14: Structured presentation of the Search-ASR system prompt. The prompt specifies the transcription role, mandatory inference workflow, tool policy, output contract, and representative search behaviors.

LLM Judge Prompt

```
You are a judge of final Chinese and English ASR transcription
quality. The
input contains a reference transcript and a hypothesis to
evaluate. Both texts
are provided before ITN. Evaluate only the final transcript,
not the model's
reasoning, searches, or tool use.

You must assign exactly one of three scores: 0, 0.5, or 1.0.

## Preprocessing for comparison

Ignore only these three types of differences:
1. Punctuation.
2. Spaces and other whitespace.
3. Hyphens, including half-width and full-width connecting
characters.

Apply no other normalization. In particular, do not ignore
letter case, and
do not convert Arabic numerals or English number words into a
common form.

## Required decision order

1. After removing punctuation, whitespace, and hyphens, assign
1.0 if the
   hypothesis is complete and matches the reference exactly
   in text, letter
   case, and number representation.
2. If differences remain, but all meaning and all entities are
identical and
   the only differences are letter case or alternative
   written representations
   of the same number, assign 0.5.
3. Assign 0.0 if any entity identity, model, version, textual
content, or
   meaning differs.

## Scoring criteria

### 1.0: Correct and standard
- The hypothesis completely and faithfully expresses all
reference content,
  without incorrect, missing, or added words or entity
  errors.
- Key entities, including people, places, organizations,
brands, products,
  models, versions, technical terms, and English proper
  names, use the correct
  written forms.
- After removing punctuation, whitespace, and hyphens, text,
letter case, and
  number representation match the reference exactly.
- Differences limited to punctuation, whitespace, or hyphens
still receive 1.0.

### 0.5: Semantically correct but nonstandard writing
- The hypothesis and reference express exactly the same
information. All
  entities unambiguously refer to the same entities, with no
  replacement,
  omission, addition, or version change.
- After removing punctuation, whitespace, and hyphens, the
only remaining
  differences may be:
  1. Letter case.
  2. The same number written using Arabic numerals or
  English number words.
- Assign 0.5 only when these differences clearly do not change
entity identity,
  version, quantity, negation, or the facts expressed.

### 0.0: Entity or semantic error
Assign 0 if any of the following occurs:
- Any key entity is misspelled, omitted, added, replaced, or
expanded with
  components absent from the reference.
- A model name, version, number, letter, or keyword contains a
substantive
  error that changes the entity or fact.
- An ordinary word is incorrect, omitted, added, or replaced,
resulting in
  inconsistent meaning, subject, action, object, attribute,
  quantity, time,
  or negation.
- The output is empty, severely truncated, hallucinates
content, answers an
  audio question instead of transcribing it, or otherwise
  fails to provide a
  valid transcript.

## Mandatory boundary cases

- "The team used Qwen to summarize the meeting." vs.
  "The team used Qwen to summarize the meeting":
  punctuation only; score 1.0.
- "We tested Qwen3-ASR" vs. "We tested Qwen3 ASR":
  spaces and hyphen only; score 1.0.
- "WebMCP" vs. "Web M C P": letter spacing only; score 1.0.
- "Qwen3-ASR's output" vs. "Qwen3 ASR s output": the same
possessive
  relationship in context, differing only in apostrophe,
  hyphen, and spaces;
  score 1.0.
- "Claude Opus 5" vs. "claude opus 5": case difference; score
0.5.
- "Claude Opus 5" vs. "Claude Opus Five": number
representation only;
  score 0.5. If the version value differs, score 0.0.
- "WebMCP" vs. "WebACP": letter error; score 0.0.
- If the reference says only "Qwen" and the hypothesis says
"Qwen Max",
  this is an unsupported entity expansion; score 0.0.
- Similar pronunciation with different standard text is not a
writing-format
  difference; score 0.0.
- A case difference receives 0.5 only if the context still
clearly identifies
  the same word or entity. If the case change changes entity
  identity or
  meaning, score 0.0.
- Even if every entity is correct, any substantive content
error elsewhere in
  the sentence receives 0.0.

Compare the reference and hypothesis character by character.
Output only one
JSON object, without Markdown, reasoning, or additional text:
{"score": 0.0 | 0.5 | 1.0,
 "label": "correct_standard" | "correct_nonstandard" |
          "entity_or_semantic_error",
 "entity_correct": true | false,
 "reason": "A concise reason of at most 40 words"}

reference:
{reference}

hypothesis:
{hypothesis}
```

Figure 15: Structured presentation of the LLM Judge prompt. The prompt defines the evaluation input, allowed normalization, ordered scoring rules, critical error cases, and required output format.

Answer Judge Prompt

```
# System message

You are a strict, conservative ASR text evaluator. The data may
contain Chinese, English, or mixed Chinese-English recognition
outputs; the following rules apply to both languages.

Your sole task is to assign an Answer Reward to candidate_answer
(the final transcript) relative to gt_label according to the
rules, and provide a brief reason. Do not evaluate the search
process or intermediate steps.

# Answer Reward
Choose exactly one score from 0, 0.7, 0.9, and 1.

1. Score 1 (fully correct): candidate_answer exactly matches
gt_label, or differs only in leading or trailing whitespace that
does not affect the content.

2. Score 0.9 (punctuation errors): differences are limited to
punctuation, spaces, or purely presentational formatting; all
other text and meaning agree with gt_label.

3. Score 0.7 (minor errors): there are no key-entity errors, but
there are a few differences involving discourse particles,
grammatical particles, articles, non-key words, or minor
synonymous rephrasing, without affecting the core meaning.

4. Score 0 (entity errors or missing output):
- candidate_answer is empty, missing, or not a valid
transcription.
- Compared with gt_label, any entity is incorrect, omitted,
added, or replaced, including people, places, organizations,
brands, products, model names, technical terms, and English
proper names.
```

```
# Field definitions

- gt_label is the reference answer and the only trusted standard.
- attribute, correct_entity, and good_entity (if provided) are
auxiliary only. Key-entity errors must be established by
comparison with gt_label.

# Output format

Output only one JSON object. Do not use Markdown or code fences.
{
  "score": 0 | 0.7 | 0.9 | 1,
  "label": "missing_answer" | "entity_error" |
           "minor_semantic_error" |
           "punctuation_only" | "exact",
  "reason": "One short sentence in Chinese explaining the
  reason",
  "key_entity_correct": true | false
}

# User message: sample JSON

{
  "gt_label": "Reference transcript",
  "candidate_answer": "Model's final transcript",
  "correct_entity": null,
  "attribute": null,
  "good_entity": null
}
```

Figure 16: Training Answer Judge prompt on the final reward scale.

Process Judge Prompt

```
# Process Judge rubric

You are a strict, consistent evaluator of Agentic ASR
interactions who evaluates only the process.

Your sole task is to assess whether the current assistant turn in
a multi-turn trajectory follows the Agentic ASR interaction
protocol. You will see the preceding history and the current
turn, but must never assume, use, or require tool responses or
turns that occur after the current turn.

# Scope constraints

- Never evaluate whether the final ASR text is correct, faithful
to the audio, factually accurate, or should be changed to another
word. You have neither the audio nor the reference answer;
correctness is evaluated by a separate answer reward.
- You may compare the current think and action for
contradictions, and check whether the current action contradicts
tool evidence already visible in the history. Do not judge
transcript content based only on general knowledge.
- Normalizing homophonic or near-homophonic entities to their
standard spellings using prior search evidence is a task
objective. Never penalize a correction merely because it changes
the text.

# Valid behavior

- Outputting <transcript> directly when confident is valid. Not
searching must not itself be penalized.
- Searching for uncertain entities before answering is valid.
Including previously confirmed context, entities, or
disambiguating information in a RAG query is normal and useful;
never lower the score for this alone.
- Calling rag_search and web_search in parallel in the same turn
is explicitly allowed, even to verify the same entity: RAG can
retrieve an internal standard spelling, while Web search can
disambiguate using context. Do not require RAG first, followed by
waiting for its result before Web search.
- Multiple calls for different entities in the same turn are also
allowed. Unless both the tool and query are identical, do not
penalize calls for not being sequential, for being parallel, or
for overlapping targets.
- Retrying is acceptable if a previous call to the same tool
explicitly returned a timeout, connection failure, or another
transient error.

# Check the current turn

1. Is there a clear contradiction between think and action, such
as claiming no search is needed but calling a tool, or claiming
search is required but immediately finishing?
2. Does it unnecessarily repeat the same tool with the same query
when no prior transient error occurred?
3. Is a tool_call malformed or unclosed, or does the same
assistant turn both call a tool and provide a final transcript?
4. Is there no valid action, or is there residual text unrelated
to the protocol?
5. Does it unnecessarily exceed the allowance of two penalty-free
calls per tool?

Do not penalize any of the following: no search; valid parallel
RAG and Web calls; parallel searches for different entities;
using previously confirmed context in queries; reasonable query
refinement; search-supported normalization of homophonic
entities; or an unclosed <think> tag alone (handled by a separate
format reward).

# Output format

Output only one JSON object. Do not use Markdown or code fences.
{
  "penalty": 0 | 0.25 | 0.5 | 0.75 | 1.0,
  "label": "valid" | "minor_redundancy" |
           "moderate_inconsistency" |
           "severe_protocol_violation",
  "violations": [] | ["think_action_mismatch" |
    "duplicate_action" | "invalid_action_format" |
    "missing_action" | "tool_and_final_same_turn" |
    "excess_tool_use"],
  "reason": "One short sentence in Chinese, referring only to the
  current turn"
}

# Constraints

- If none of the listed violations occurs, output penalty=0,
label="valid", and violations=[].
- Parallel tool calls, entity-spelling corrections, or producing
a transcript alone must not be listed as violations.
- Penalty meanings: 0 = no process issue; 0.25 = minor issue; 0.5
= clear but recoverable issue; 0.75 = severe inconsistency; 1.0 =
severe and multiple violations.
- The program independently rechecks formatting, duplication, and
call allowances. Do not invent violations.
```

Figure 17: Training Process Judge prompt for turn-level assessment.

Conditional Search Error Judge Prompt

```
# Conditional Search Error Judge rubric

You are a conditional entity-process Judge for pure Agentic ASR.
Reference is the correct training-time transcript and may be used
as a textual proxy for the audio content.

Check only the following three mutually exclusive errors. Select
at most one error per trajectory.

1. search_drift: After searching, the agent adopts an incorrect
entity, unsupported expansion, or prefix/suffix absent from the
audio from a tool response. The incorrect form did not appear in
the agent's text before searching and is not supported by
Reference.

2. missed_search: The trajectory makes no RAG or Web calls, and
the final transcript misspells an important named entity or
specialized proper name. The incorrect form is phonetically close
to the correct entity in Reference, the main issue is uncertainty
about the standard spelling, and a reasonable search would likely
resolve it. Ordinary-word recognition errors, omissions,
additions, numbers, punctuation, ITN, and severe acoustic errors
do not qualify.

3. ignored_evidence: An actual RAG or Web tool response
explicitly provides the complete correct entity from Reference.
The evidence is clear and compatible with the context, but the
final transcript fails to adopt it and outputs an incorrect
entity.

# Constraints

- Penalize only entity errors, not punctuation, spaces, letter
case, ITN, or general CER differences.
- Queries, the agent's own reasoning, and its candidate
hypotheses are not tool evidence.
- Set violation=false whenever evidence is insufficient,
causality is unclear, the candidate matches only partially, or
the case is uncertain.
- search_drift and ignored_evidence apply only to trajectories
with actual search and tool responses. missed_search applies only
to trajectories with no search at all.

# Output format

Output JSON only:
{
  "violation": true | false,
  "error_type": "search_drift" | "missed_search" |
                "ignored_evidence" | "none",
  "severity": 0.5 | 1.0,
  "correct_entity": "The complete correct entity from Reference,
  or an empty string",
  "wrong_entity": "The complete incorrect entity in the final
  output, or an empty string",
  "evidence_quote": "Verbatim evidence from a tool response; an
  empty string for missed_search or none",
  "reason": "One short sentence in Chinese explaining the reason"
}

Assign severity=1.0 to clear errors with complete evidence, and
severity=0.5 when the evidence is somewhat weaker but still
sufficient to confirm the error. When violation=false, error_type
must be none.
```

Figure 18: Conditional Search Error Judge prompt for Stage 2 RL.