

StreamCorrect: Self-Supervised Error Correction for Real-Time ASR with Offline Recognizers

Tung D. Vu^{1,3}, Zeying Xie², Weiwei Zhao², Victor Junqiu Wei¹, Di Jiang², Yi R. Fung¹

¹The Hong Kong University of Science and Technology ²WeBank ³VinUniversity
dtvu@connect.ust.hk, yrfung@ust.hk

Abstract

Streaming Automatic Speech Recognition (ASR), also known as online or real-time ASR, transcribes speech incrementally while audio is being spoken and is critical for many industry applications. Despite the strong performance of modern offline ASR models, streaming deployment remains challenging because limited future context can cause early recognition errors to propagate across later chunks. We define the task of streaming ASR error correction and propose StreamCorrect, a modular framework that fine-tunes a lightweight error corrector on self-generated streaming ASR errors and integrates it with existing streaming ASR systems. To enable training without human annotation, StreamCorrect introduces a timestamp-free boundary recovery procedure based on Prefix-to-Reference Alignment for generating chunk-level correction data. Across multiple languages and ASR backbones, StreamCorrect achieves up to 39% relative character error rate (CER) reduction with minimal latency overhead. Error propagation analysis reveals progressively larger gains over time, with up to 7.2% absolute CER reduction in later audio chunks. StreamCorrect bridges offline ASR models and streaming deployment through a lightweight correction plug-in while preserving the pretrained ASR component without additional tuning or distillation.¹

1 Introduction

In industrial applications, automatic speech recognition (ASR) models have achieved remarkable progress across languages (team et al., 2025), audio lengths (Peng et al., 2025), and domains (Rangappa et al., 2025), enabled by large-scale data and powerful Transformer- or Conformer-based architectures. However, real-time applications such as online captioning and voice assistants remain

challenging because decoding must proceed with limited future context. Traditional streaming ASR addresses this issue through causal architectures, including Recurrent Neural Network Transducers (RNN-T) (Kurata and Saon, 2020), Transformer Transducers (Zhang et al., 2020), and CTC-based models (Li et al., 2025b). More recently, offline-based streaming ASR has emerged as an alternative paradigm that deploys strong offline models in real time through streaming inference, avoiding architectural changes or retraining of the speech recognizer. Approaches such as SimulWhisper (Wang et al., 2024), WhisperStreaming (Macháček et al., 2023), and SimulStreaming (Macháček and Polák, 2025) adapt offline models such as Whisper for streaming transcription and translation.

Despite this practicality, offline-based streaming ASR systems remain limited by restricted decoding context and error propagation, where early speech recognition errors can accumulate over time. Error correction offers a natural way to mitigate such errors by leveraging linguistic context, alternative hypotheses, or external language models, and has been studied for offline ASR (Li and Wang, 2024; Lin et al., 2025; Zhao and Liu, 2025). However, applying error correction to streaming ASR is substantially more challenging: hypotheses are incomplete, the recognition context evolves over time, and corrections must be made under real-time constraints. As a result, streaming ASR error correction remains underexplored despite its practical importance for deploying strong offline recognizers in real-time settings.

This paper presents StreamCorrect, a training and inference framework for streaming error correction integrated with existing offline-based streaming ASR systems. We formulate streaming error correction as generating corrected transcript segments at each chunk using (i) the confirmed transcript prefix, (ii) top- k streaming hypotheses, and (iii) observed audio, enabling error correc-

¹Code is available at <https://github.com/vu-duy-tung/StreamCorrect>.

tion while keeping the module decoupled from the underlying ASR module. A key challenge is constructing streaming-specific correction data without additional human annotation. StreamCorrect generates supervision from the streaming system itself by exposing diverse error patterns through different chunk and beam configurations. These outputs are converted into chunk-level training data using Prefix-to-Reference Alignment (PRA), a timestamp-free boundary recovery method. The synthesized data is then used to fine-tune a lightweight Speech Language Model (SpeechLM) as the error corrector.

StreamCorrect reduces error propagation during streaming inference, achieving lower error rates across languages and ASR backbones with minimal latency overhead. Our contributions are summarized as follows:

- We introduce and formalize streaming error correction for offline-based streaming ASR systems, capturing the incremental and real-time nature of streaming recognition.
- We propose StreamCorrect, a practical framework with a self-supervised training recipe that generates streaming-specific correction data from self-generated errors and timestamp-free Prefix-to-Reference Alignment. We release the correction data, model checkpoints, and source code to support future research and development.
- Comprehensive experiments demonstrate consistent improvements across languages and ASR backbones, achieving a 39% relative CER reduction on Mandarin (AIShell-1) and a 35% relative CER reduction on Cantonese (WSYue) with minimal latency overhead. StreamCorrect also generalizes to industrial conditions, substantially narrowing the performance gap with offline upper bounds. Furthermore, StreamCorrect mitigates error accumulation over time, achieving up to a 7.2% absolute CER reduction in later streaming chunks.

2 Related Work

Offline-based Streaming ASR. Offline-based streaming ASR adapts strong offline ASR models to real-time inference by processing incoming audio in chunks while keeping the original model unchanged. Representative Whisper-based approaches mainly differ in how they decide which part of each partial decoding is reliable enough

to confirm. WhisperStreaming (Macháček et al., 2023) repeatedly decodes the growing audio buffer and confirms the longest prefix shared by consecutive hypotheses, providing a simple but potentially variable-latency solution. SimulWhisper (Wang et al., 2024) instead uses Whisper’s cross-attention to estimate where the current decoding has reached in the audio and stops accordingly, while additionally detecting truncated words at chunk boundaries. SimulStreaming (Macháček and Polák, 2025) builds on this attention-based stopping mechanism while incorporating the streaming interface and buffer management of WhisperStreaming, enabling unbounded speech processing. StreamCorrect is complementary to this line of work: rather than re-designing the streaming recognizer, it adds a modular correction stage that targets the errors induced by streaming inference.

Error Correction in ASR. ASR error correction aims to improve recognition outputs by modeling systematic errors using linguistic and acoustic information. Early neural approaches, such as FastCorrect2 (Leng et al., 2021) and SoftCorrect (Leng et al., 2023), demonstrate effective sequence-level correction, while recent LLM-based methods (Chen et al., 2023; Fang et al., 2025; Li et al., 2024; Ma et al., 2023) leverage stronger language modeling capabilities for post-correction. Benchmarks such as ClozeGER (Hu et al., 2024) and the GenSec Challenge (Yang et al., 2024) further study ASR refinement under realistic error distributions. Most of these methods assume complete utterance-level hypotheses and full context, which are unavailable during streaming recognition. For streaming ASR, Fast-Slow encoders (Li et al., 2022) and Partial Rewriting (Bruguier et al., 2023) explore refinement through multi-scale modeling and hypothesis revision, respectively. These approaches either modify the ASR architecture or rely on heuristic hypothesis combination, whereas StreamCorrect trains a modular SpeechLM-based corrector on chunk-level streaming errors and can be attached to existing streaming systems.

3 Approach

StreamCorrect bridges the gap between offline recognizer quality and real-time streaming deployment by adding a lightweight correction module to an existing offline-based streaming ASR system. The framework first fine-tunes a SpeechLM error corrector on self-generated streaming errors,

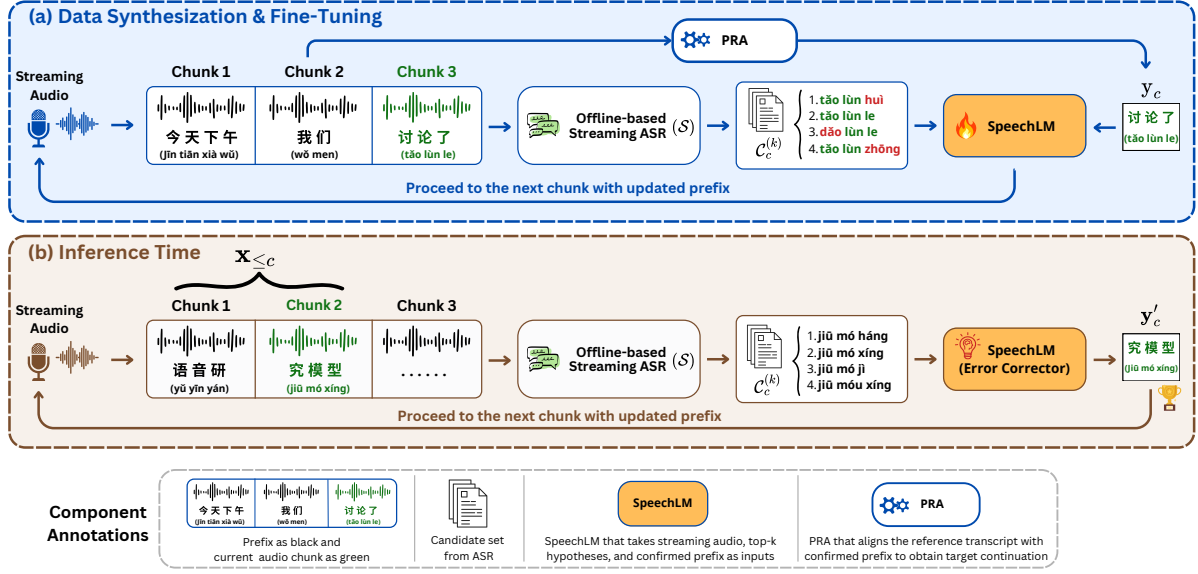


Figure 1: Overview of StreamCorrect. (a) Data synthesis and fine-tuning. We use an offline-based streaming ASR system to generate streaming-specific error correction data. Prefix-to-Reference Alignment (PRA) recovers chunk-level supervision without timestamps by identifying the boundary between confirmed prefixes and future content. (b) Inference time. The fine-tuned SpeechLM is integrated as a modular error corrector, refining the top- k hypotheses of each chunk using available context to produce corrected transcripts during streaming inference.

then integrates the trained corrector into the same streaming pipeline at inference time to improve recognition quality, as illustrated in Figure 1.

3.1 Problem Definition

In streaming ASR, recognition is performed incrementally with limited future acoustic context. Since emitted tokens are typically committed and no longer revised, early recognition errors can propagate to later predictions, motivating error correction during streaming inference. Formally, let $\mathbf{x}_{\leq c}$ denote the audio observed up to chunk c , and let $\mathbf{y}'_{<c}$ denote the confirmed prefix consisting of previously committed tokens. The objective of *streaming error correction* is to generate the corrected transcript segment $\mathbf{y}'_c$ at chunk c conditioned on the confirmed prefix, recognition hypotheses, and observed audio:

$$\mathbf{y}'_c = \arg \max_{\mathbf{y}_c} P(\mathbf{y}_c | \mathbf{y}'_{<c}, \mathcal{C}_c^{(k)}, \mathbf{x}_{\leq c}).$$

The generated segment $\mathbf{y}'_c$ is appended to the confirmed prefix and used as context for subsequent chunks. Here, $\mathcal{C}_c^{(k)}$ represents the top- k recognition hypotheses produced at chunk c by the streaming ASR system $\mathcal{S}$:

$$\mathcal{C}_c^{(k)} = \mathcal{S}(\mathbf{y}'_{<c}, \mathbf{x}_{\leq c}),$$

where k denotes the beam size. In practice, the prefix does not necessarily align exactly with the

most recent audio chunk, as token emission may be delayed until sufficient acoustic context becomes available under the streaming mechanism.

3.2 Overview of StreamCorrect

As shown in Figure 1, StreamCorrect leverages self-generated data from an offline-based streaming system to fine-tune a SpeechLM as a lightweight error corrector. The corrector is then integrated back into the streaming ASR pipeline to refine recognition outputs online. In this work, SimulStreaming is adopted as the representative offline-based streaming ASR system.

Self-generated streaming error correction data.

Training a streaming error corrector requires chunk-level supervision, where each streaming prediction is paired with the corresponding reference transcript segment. However, most speech datasets provide only utterance-level transcripts without token-level timestamps. Directly assigning reference tokens based on audio boundaries is unreliable, since the textual emission boundary of a streaming ASR system may differ from the temporal boundary of an audio chunk. Such mismatches can introduce incorrect supervision and accumulate errors during data synthesis.

To address this issue, StreamCorrect introduces a *Prefix-to-Reference Alignment (PRA)* procedure for timestamp-free boundary recovery. Given the

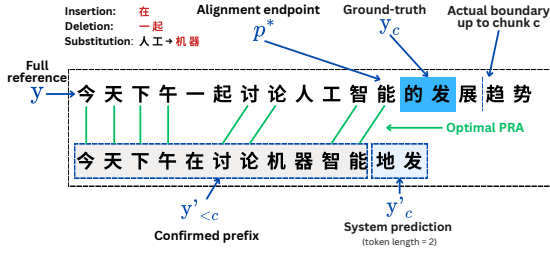


Figure 2: Illustration of Prefix-to-Reference Alignment (PRA) for timestamp-free boundary recovery.

ground-truth transcript y and the confirmed prefix $y'_{<c}$, PRA identifies the reference position p^* corresponding to the end of the confirmed prefix. The key observation is that although streaming ASR outputs may contain local insertion, deletion, and substitution errors, the confirmed transcript generally preserves the ordering of the underlying utterance and can be viewed as an approximate prefix of y . Therefore, the endpoint of the optimal alignment path between $y'_{<c}$ and y provides a reliable estimate of the current textual decoding boundary. An example of PRA is illustrated in Figure 2, where the confirmed prefix contains recognition errors but the recovered alignment endpoint successfully identifies the corresponding reference boundary.

Specifically, PRA formulates boundary recovery as a semi-global sequence alignment problem. Let $m = |y'_{<c}|$ and $n = |y|$. A dynamic programming matrix $\mathbf{D} \in \mathbb{R}^{(m+1) \times (n+1)}$ is constructed, where $D(i, j)$ denotes the minimum Levenshtein distance (Levenshtein, 1966) between the first i tokens of $y'_{<c}$ and the first j tokens of y . The recurrence is defined as:

$$D(i, j) = \min \begin{cases} D(i-1, j) + 1, \\ D(i, j-1) + 1, \\ D(i-1, j-1) + \mathbb{I}[y'_{<c,i} \neq y_j]. \end{cases}$$

The three transitions correspond to deletion, insertion, and substitution operations, respectively. To fully specify the dynamic program, the boundary conditions are

$$D(0, 0) = 0, \quad D(i, 0) = i, \quad D(0, j) = j,$$

where aligning a non-empty prefix with an empty sequence requires i deletions, aligning an empty prefix with j reference tokens requires j insertions. The recurrence is complete for the Levenshtein objective because the final operation of any alignment must be one of these three cases. Since each transition extends an optimal solution of its corresponding preceding subproblem, $D(i, j)$ gives the

minimum edit cost between the first i tokens of $y'_{<c}$ and the first j tokens of y .

Unlike conventional sequence matching, PRA adopts a semi-global alignment objective: all tokens in $y'_{<c}$ are aligned, while the alignment is allowed to terminate at any position in y . Therefore, minimizing the final row over all reference positions yields the globally minimum-cost endpoint for the confirmed prefix.

The alignment boundary is obtained as:

$$p^* = \arg \min_j D(m, j),$$

where p^* corresponds to the endpoint of the optimal alignment path. When multiple endpoints attain the same minimum cost, PRA deterministically selects the smallest endpoint. This conservative tie-breaking rule avoids prematurely advancing the reference boundary when the alignment provides equally good alternatives. In particular, it allows the error corrector to retain potentially recoverable reference content for subsequent correction rather than forcing the current prediction to immediately catch up to a later endpoint. Thus, even when the recovered boundary does not exactly coincide with the real-time emission boundary, the supervision remains deliberately biased toward preserving content that can be corrected with additional acoustic context, rather than discarding it prematurely.

The current chunk target is then extracted using the number of tokens newly committed by the streaming ASR system:

$$y_c = y[p^* + 1 : p^* + |\hat{y}_c|].$$

PRA also naturally handles code-switched speech because it operates directly on token sequences without language-specific alignment rules. Tokens from different languages are treated uniformly under the same edit-distance objective. For example, if an English word in code-switched speech is tokenized as *com*, *put*, *er* while the ASR predicts *com*, *pat*, *er*, PRA simply treats *put* $\rightarrow$ *pat* as a substitution while aligning the surrounding tokens normally.

This design preserves the streaming decoder’s own emission behavior and stopping mechanism while replacing unavailable timestamps with an alignment-based textual boundary. Consequently, PRA converts utterance-level annotations into chunk-level supervision without additional timestamp information, while remaining robust to local insertion, deletion, and substitution errors.

Real-time inference with error corrector. After being fine-tuned on the self-generated streaming error correction data, the error corrector is integrated immediately after the offline ASR model during inference. In each streaming iteration, the offline model receives an additional audio chunk and produces top- k hypotheses for the newly generated transcript segment. The error corrector then takes the current audio chunk, the top- k hypotheses, and the previously confirmed prefix as input, and outputs a finalized transcript segment for the current step. This finalized segment is appended to the confirmed prefix and retained as context for processing subsequent audio chunks.

4 Experiments

4.1 Benchmark and Experimental Setup

Benchmark. StreamCorrect is evaluated on AIShell-1 (Bu et al., 2017) (178 hours), a widely used Mandarin ASR benchmark, and an internal industrial Mandarin benchmark consisting of banking-domain conversations with realistic in-house noise. The industrial benchmark contains 30 minutes of audio but represents challenging real-world scenarios with short context-poor utterances, financial entities (e.g., amounts, rates, and phone numbers) and disfluent speech patterns. To assess generalization across languages, we additionally evaluate on WSYue (Li et al., 2025a), a 9.5-hour Cantonese ASR dataset covering 2,861 speakers and a wide range of usage scenarios, including Cantonese-English code-switching and multi-domain conditions.

Metrics. Character Error Rate (CER) is the primary accuracy metric. Latency is measured by First Token Latency (FTL), the time from audio reception to the first emitted token, and Last Token Latency (LTL), the time from the end of the audio stream to transcript finalization. All latency measurements are obtained on NVIDIA H20 GPU.

Base models. Belle-Whisper-large-v2-zh (Group, 2023) is adopted as the base offline ASR model for Mandarin in the streaming system, and is referred to as Belle Whisper for brevity. For Cantonese, we use Whisper Medium Yue (Li et al., 2025a) as the base offline ASR model. For the error corrector, Ultravox-v0.5-Llama-3.2-1B (Fixie AI, 2024) is adopted as the primary SpeechLM and fine-tuned accordingly.

Chunk size	Sub.	Ins.	Del.	Overall
100ms	13.1	6.8	1.7	21.6
500ms	8.4	6.6	3.2	18.2
1000ms	7.0	4.5	3.2	14.7
1500ms	6.2	3.4	3.2	12.8

Table 1: Substitution, insertion, and deletion error statistics (% CER) in the self-generated AIShell-1 training data.

Details on error corrector fine-tuning. The error corrector is fine-tuned from the Ultravox-v0.5-Llama-3.2-1B backbone via instruction tuning with LoRA. Adapters are injected into the attention projections (Q, K, V, O) and feed-forward layers. Training uses cross-entropy loss and is optimized with AdamW at a learning rate of 1×10^{-4} and a cosine scheduler with a 0.01 warmup ratio.

4.2 Statistics of Self-Generated Training Data

Following the StreamCorrect pipeline, SimulStreaming is applied to the AIShell-1 training set under diverse streaming configurations, including chunk sizes of 100ms, 500ms, 1000ms, and 1500ms, and beam sizes of 1, 2, 4, and 8. This process yields approximately 1 million self-generated training samples for Mandarin.

Table 1 shows that the generated training data expose the error corrector to a wide range of error scales, with CER ranging from 12.8% at a 1500ms chunk size to 21.6% at a 100ms chunk size. Substitution errors dominate, increasing from 6.2% at 1500ms to 13.1% at 100ms. This indicates that streaming systems largely maintain token count but struggle with lexical selection. Using the homophone dictionary from SoftCorrect (Leng et al., 2023), we find that about 22% of substitutions arise from homophone confusion. For instance, the top-1 hypothesis may predict 在 (“at”) while the ground truth is 再 (“again”), which share identical pronunciation but differ semantically.

4.3 Results and Analysis

Table 2 compares StreamCorrect with offline-based streaming systems, including WhisperStreaming and SimulStreaming, together with their corresponding offline backbones (Belle Whisper for Mandarin and Whisper Medium Yue for Cantonese). On AIShell-1, StreamCorrect consistently improves over existing streaming baselines across all chunk sizes. At 100ms, it reduces CER from 27.3% to 11.7% compared with SimulStreaming, with only 49ms additional FTL and 84ms addi-

Dataset	Metric	Offline Backbone (upper bound [†])	WhisperStreaming			SimulStreaming			StreamCorrect (Ours)		
			100ms	500ms	1000ms	100ms	500ms	1000ms	100ms	500ms	1000ms
AIShell-1 (Mandarin)	CER(%)↓	2.5	38.4	23.6	15.5	27.3	16.3	13.9	11.7	9.8	7.9
	FTL(ms)↓	—	441	441	442	158	158	158	207	219	234
	LTL(ms)↓	—	325	363	971	200	213	227	284	295	321
Our industrial data (Mandarin)	CER(%)↓	16.2	55.8	39.4	31.2	38.7	27.8	24.1	26.8	20.7	18.5
	FTL(ms)↓	—	432	433	435	152	153	154	201	213	229
	LTL(ms)↓	—	337	375	1001	195	210	224	279	292	318
WSYue (Cantonese)	CER(%)↓	6.3	31.8	22.4	15.7	21.2	15.6	14.1	13.9	10.2	8.2
	FTL(ms)↓	—	228	228	229	82	82	82	197	197	197
	LTL(ms)↓	—	201	217	569	123	127	133	289	294	301

Table 2: Performance comparison across chunk sizes on AIShell-1 (Mandarin), an industrial Mandarin benchmark, and WSYue (9.5h Cantonese, 2,861 speakers). Offline backbones are Belle Whisper for Mandarin and Whisper Medium Yue for Cantonese, serving as the base models for the streaming systems and evaluated with full context. All streaming methods use beam size 4. [†]Offline Backbone is the reference that streaming ASR aims to approach.

tional LTL. At 500ms and 1000ms, it further reduces CER by about 6%, corresponding to 39% and 43% relative reductions, while maintaining low latency overhead. In contrast, WhisperStreaming suffers severe degradation under low-latency settings, reaching 38.4% CER at 100ms and 15.5% at 1000ms. We further evaluate StreamCorrect on our industrial Mandarin benchmark and WSYue Cantonese to assess cross-domain and cross-lingual generalization. On the industrial dataset, where distorted speech and domain shift make the offline upper bound weaker (16.2% CER), StreamCorrect achieves 20.7% CER at 500ms and substantially narrows the gap compared with SimulStreaming (27.8%). On WSYue, StreamCorrect also improves consistently, reducing CER from 21.2% to 13.9% at 100ms and from 14.1% to 8.2% at 1000ms over SimulStreaming. These results show that StreamCorrect improves streaming ASR quality across languages and domains while introducing only modest latency overhead.

We additionally analyze StreamCorrect under varying beam sizes and base ASR models to examine its robustness and adaptability. As shown in Table 3, StreamCorrect achieves its lowest CER at

Beam Size	SimulStreaming CER(%)↓	StreamCorrect CER(%)↓
1	23.4	11.9
2	14.4	10.0
4	16.3	9.8
8	17.2	10.6

Table 3: CER comparison of SimulStreaming and StreamCorrect (ours) at 500ms chunk size across beam sizes on the AIShell-1 test set.

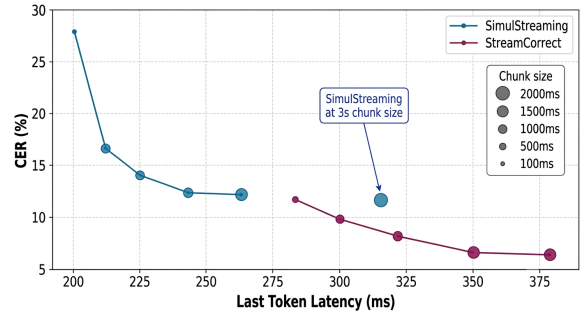


Figure 3: CER vs. LTL trade-off between SimulStreaming and StreamCorrect (ours) on the AIShell-1 test set.

a beam size of 4, which corresponds to our primary streaming configuration. The fine-tuned error corrector consistently improves performance across all beam sizes. At beam size 1, error correction yields the largest absolute gain, reducing CER by 11.4%. Even at beam size 2, where the baseline already performs relatively well, StreamCorrect still achieves a 4.4% absolute reduction, demonstrating its robustness under varying decoding settings.

The latency–accuracy trade-off with and without error correction is shown in Figure 3. StreamCorrect at a 100ms chunk size achieves lower CER than SimulStreaming at 2000ms (11.7% vs. 12.3%), indicating greater robustness under low-latency settings. When reducing the chunk size from 500ms to 100ms, StreamCorrect incurs only a 1.9% CER increase, whereas SimulStreaming exhibits an 11.0% increase. Across chunk sizes, the LTL increases by an average of 96ms and remains below 300ms for chunk sizes up to 500ms. Even at comparable latency (312ms) and up to 3s chunk size, SimulStreaming still yields higher CER than StreamCorrect.

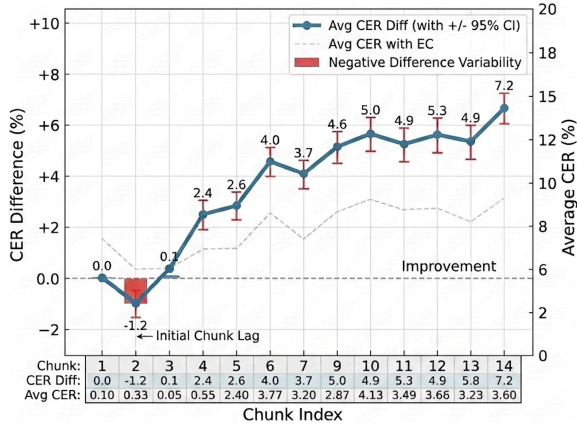


Figure 4: Chunk-wise ASR performance on the AIShell-1 test set (utterances shorter than 4s excluded). The primary axis (blue line) shows the CER difference, where positive values indicate improvement from error correction. The secondary axis (dashed line) tracks the average CER with error correction. Exact values are shown in the bottom table.

4.4 Error Propagation Effect

Figure 4 shows that during the first three chunks (0–1.5s), the error corrector makes only minor edits, with a slight degradation in accuracy at chunk 2 due to over-correction. From chunk 4 onward, it begins to improve performance, yielding a 2.4% absolute CER reduction over the no-correction baseline. The gains continue to grow over time as StreamCorrect progressively mitigates accumulated errors: CER is reduced by 4.6% at chunk 8 (3.5–4.0s) and 5.3% at chunk 11 (5.0–5.5s), eventually peaking at 7.2% at chunk 14. A qualitative example illustrating this error propagation and correction process is provided in Appendix B.

4.5 Industrial Integration

We report an industrial integration of StreamCorrect into an existing Chinese banking transcription pipeline, using the top-4 hypotheses from the offline ASR for correction. The existing ASR system remains unchanged, while the error corrector is added as an independent, plug-and-play module, facilitating independent development and maintenance of the two components.

For large-scale batch transcription, the integrated ASR–corrector pipeline is deployed across multiple NVIDIA H20 GPUs using process-level data parallelism. Each worker is assigned to a GPU and maintains a complete copy of both the Whisper large-v2 backbone and the Ultravox-1B error corrector. Input calls are partitioned across workers, allowing multiple calls to be processed concurrently across

GPUs. In our current deployment, workers can process multiple calls sequentially, while additional workers can be allocated to the same GPU when higher device utilization is beneficial and sufficient memory remains available.

Under FP16/BF16 inference, Whisper large-v2 (1.55B parameters) requires approximately 3.1 GB for model weights, while the Ultravox-1B SpeechLM corrector (1.92B parameters) requires approximately 3.8 GB. Thus, the two models occupy approximately 6.9 GB of model weights per replica, leaving substantial memory headroom on an NVIDIA H20 for runtime states and additional inference workers. The practical number of concurrent workers per GPU depends on runtime memory usage and workload characteristics. Across multiple GPUs, this process-level parallelism provides a straightforward way to increase batch transcription throughput without model sharding.

The correction stage is constrained to at most 8 newly generated tokens per 500ms streaming chunk. This bounds the decoding workload at each step and prevents aberrant corrections, such as repetitive or overly destructive outputs from consuming excessive computation and accumulating into subsequent chunks. By constraining the correction workload for individual streaming steps, the design is better suited to maintaining stable processing as the audio stream grows longer.

5 Conclusion

In this work, we propose StreamCorrect, a streaming ASR error correction framework that mitigates error propagation in offline-based streaming ASR without modifying or retraining the underlying backbone. By formulating correction as a modular component and introducing a self-supervised data generation strategy with Prefix-to-Reference Alignment, StreamCorrect can be integrated into existing streaming ASR systems without additional human annotation. The decoupled design also allows the corrector to be independently improved or reused across different ASR backbones, which is especially useful for industry settings where updating large recognizers is costly. Experiments across Mandarin and Cantonese benchmarks, including an industrial benchmark, demonstrate consistent CER reductions with approximately 100ms additional latency. These results show that StreamCorrect provides a practical and maintainable solution for industry-grade streaming ASR.

Limitations

A limitation of this work is that, although we evaluate StreamCorrect on Mandarin and Cantonese, both are tonal Sinitic languages with similar morphological characteristics. Its effectiveness on morphologically rich languages, highly agglutinative languages such as Turkish or Finnish, and settings involving domain-specific vocabulary has not yet been assessed. In particular, its reliability in high-stakes domains such as medical and legal applications, where specialized terminology and transcription errors may have more serious consequences, remains unexplored. Future work will explore these more diverse linguistic and domain conditions, as well as code-switching and cross-lingual streaming scenarios, to better understand the generalization of streaming error correction across languages and domains.

Ethical Considerations

Streaming ASR systems are increasingly deployed in sensitive applications, such as banking and customer services, where recognition errors or unintended model behaviors may impact privacy and security. In higher-stakes settings, such as medical or legal applications, transcription or correction errors may also lead to consequential misunderstandings or decisions. Since StreamCorrect operates on ASR hypotheses and audio inputs, its deployment should follow appropriate data governance practices, including secure data handling, access control, and careful monitoring to prevent unexpected correction behaviors on sensitive information.

For training data construction, StreamCorrect synthesizes correction data from publicly available speech corpora without requiring additional human annotation. However, models trained or adapted on domain-specific data may inherit biases from the training distribution or memorize sensitive patterns. The SpeechLM-based correction model also introduces additional computational requirements for training and inference, which may increase energy consumption and the associated environmental footprint. Therefore, future deployments should incorporate privacy-aware and computationally responsible training practices, together with comprehensive evaluation to ensure reliable and responsible use.

We emphasize that StreamCorrect is designed as an error correction component and should be integrated into responsible ASR deployment pipelines.

In high-stakes applications, domain-specific validation and appropriate human oversight remain essential, and the system should not be treated as a substitute for professional judgment.

Acknowledgment

This work is supported in part by grant Web24EG01 and the Research Travel Grant. We express sincere gratitude to the anonymous reviewers for their insightful comments and constructive suggestions.

References

- Antoine Bruguier, David Qiu, and Yanzhang He. 2023. [Partial rewriting for multi-stage asr](#). *Preprint*, arXiv:2312.09463.
- Hui Bu, Jiayu Du, Xingyu Na, Bengu Wu, and Hao Zheng. 2017. Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline. In *Oriental COCODA 2017*, page Submitted.
- Chen Chen, Yuchen Hu, Chao-Han Huck Yang, Sabato Marco Siniscalchi, Pin-Yu Chen, and Eng Siong Chng. 2023. Hyporadise: an open baseline for generative speech recognition with large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.
- Yangui Fang, Baixu Chen, Jing Peng, Xu Li, Yu Xi, Chengwei Zhang, and Guohui Zhong. 2025. [Fewer hallucinations, more verification: A three-stage llm-based framework for asr error correction](#). *Preprint*, arXiv:2505.24347.
- Fixie AI. 2024. Ultravox: A fast multimodal llm designed for real-time voice interactions. <https://github.com/fixie-ai/ultravox>.
- BELLE Group. 2023. Belle: Be everyone's large language model engine. <https://github.com/LianjiaTech/BELLE>.
- Yuchen Hu, Chen Chen, Chengwei Qin, Qiushi Zhu, Eng Siong Chng, and Ruizhe Li. 2024. [Listen again and choose the right answer: A new paradigm for automatic speech recognition with large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 666–679, Bangkok, Thailand. Association for Computational Linguistics.
- Gakuto Kurata and George Saon. 2020. [Knowledge distillation from offline to streaming rnn transducer for end-to-end speech recognition](#). In *Interspeech 2020*, pages 2117–2121.
- Yichong Leng, Xu Tan, Wenjie Liu, Kaitao Song, Rui Wang, Xiang-Yang Li, Tao Qin, Ed Lin, and Tie-Yan Liu. 2023. [Softcorrect: error correction with soft](#)

- detection for automatic speech recognition. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press.
- Yichong Leng, Xu Tan, Rui Wang, Linchen Zhu, Jin Xu, Wenjie Liu, Linquan Liu, Xiang-Yang Li, Tao Qin, Edward Lin, and Tie-Yan Liu. 2021. **FastCorrect 2: Fast error correction on multiple candidates for automatic speech recognition**. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4328–4337, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710.
- Ke Li, Jay Mahadeokar, Jinxi Guo, Yangyang Shi, Gil Keren, Ozlem Kalinli, Michael L. Seltzer, and Duc Le. 2022. **Improving fast-slow encoder based transducer with streaming deliberation**. *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.
- Longhao Li, Zhao Guo, Hongjie Chen, Yuhang Dai, Ziyu Zhang, Hongfei Xue, Tianlun Zuo, Chengyou Wang, Shuiyuan Wang, Jie Li, Xin Xu, Hui Bu, Binbin Zhang, Ruibin Yuan, Ziya Zhou, Wei Xue, and Lei Xie. 2025a. **Wenetspeech-yue: A large-scale cantonese speech corpus with multi-dimensional annotation**. *Preprint*, arXiv:2509.03959.
- Longhao Li, Yangze Li, Hongfei Xue, Jie Liu, Shuai Fang, Kai Wang, and Lei Xie. 2025b. **Delayed-KD: Delayed Knowledge Distillation based CTC for Low-Latency Streaming ASR**. In *Interspeech 2025*, pages 4413–4417.
- Sheng Li, Chen Chen, Chin Yuen Kwok, Chenhui Chu, Eng Siong Chng, and Hisashi Kawai. 2024. **Investigating ASR Error Correction with Large Language Model and Multilingual 1-best Hypotheses**. In *Interspeech 2024*, pages 1315–1319.
- Wei Li and Houfeng Wang. 2024. **Detection-correction structure via general language model for grammatical error correction**. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1748–1763, Bangkok, Thailand. Association for Computational Linguistics.
- Yen-Ting Lin, Zhehuai Chen, Piotr Zelasko, Zhen Wan, Xuesong Yang, Zih-Ching Chen, Krishna C Puvvada, Ke Hu, Szu-Wei Fu, Jun Wei Chiu, Jagadeesh Balam, Boris Ginsburg, Yu-Chiang Frank Wang, and Chao-Han Huck Yang. 2025. **NeKo: Cross-modality post-recognition error correction with tasks-guided mixture-of-experts language model**. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 222–236, Vienna, Austria. Association for Computational Linguistics.
- Yingyi Ma, Zhe Liu, and Ozlem Kalinli. 2023. **Correction focused language model training for speech recognition**. *Preprint*, arXiv:2310.11003.
- Dominik Macháček, Raj Dabre, and Ondřej Bojar. 2023. **Turning whisper into real-time transcription system**. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 17–24, Bali, Indonesia. Association for Computational Linguistics.
- Dominik Macháček and Peter Polák. 2025. **Simultaneous translation with offline speech and LLM models in CUNI submission to IWSLT 2025**. In *Proceedings of the 22nd International Conference on Spoken Language Translation (IWSLT 2025)*, pages 389–398, Vienna, Austria (in-person and online). Association for Computational Linguistics.
- Zhiliang Peng, Jianwei Yu, Wenhui Wang, Yaoyao Chang, Yutao Sun, Li Dong, Yi Zhu, Weijiang Xu, Hangbo Bao, Zehua Wang, Shaohan Huang, Yan Xia, and Furu Wei. 2025. **Vibevoice technical report**. *Preprint*, arXiv:2508.19205.
- Pradeep Rangappa, Andrés Carofilis, Jeena Prakash, Shashi Kumar, Sergio Burdisso, Srikanth Madikeri, Esaú Villatoro-Tello, Bidisha Sharma, Petr Motlicek, Kadri Hacioglu, Shankar Venkatesan, Saurabh Vyas, and Andreas Stolcke. 2025. **Efficient Data Selection for Domain Adaptation of ASR Using Pseudo-Labels and Multi-Stage Filtering**. In *Interspeech 2025*, pages 4928–4932.
- Omnilingual ASR team, Gil Keren, Artyom Kozhevnikov, Yen Meng, Christophe Ropers, Matthew Setzler, Skyler Wang, Ife Adebara, Michael Auli, Can Balioglu, Kevin Chan, Chierh Cheng, Joe Chuang, Caley Droof, Mark Duppenhaler, Paul-Ambroise Duquenne, Alexander Erben, Cynthia Gao, Gabriel Mejia Gonzalez, and 14 others. 2025. **Omnilingual asr: Open-source multilingual speech recognition for 1600+ languages**. *Preprint*, arXiv:2511.09690.
- Haoyu Wang, Guoqiang Hu, Guodong Lin, Wei-Qiang Zhang, and Jian Li. 2024. **Simul-whisper: Attention-guided streaming whisper with truncation detection**. In *Interspeech 2024*, page 4483–4487. ISCA.
- Chao-Han Huck Yang, Taejin Park, Yuan Gong, Yuanchao Li, Zhehuai Chen, Yen-Ting Lin, Chen Chen, Yuchen Hu, Kunal Dhawan, Piotr Zelasko, Chao Zhang, Yun-Nung Chen, Yu Tsao, Jagadeesh Balam, Boris Ginsburg, Sabato Marco Siniscalchi, Eng Siong Chng, Peter Bell, Catherine Lai, and 2 others. 2024. **Large language model based generative error correction: A challenge and baselines for speech recognition, speaker tagging, and emotion recognition**. In *2024 IEEE Spoken Language Technology Workshop (SLT)*, page 371–378. IEEE.

Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar. 2020. [Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss](#). *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7829–7833.

Zhilong Zhao and Yindi Liu. 2025. [A hierarchical error framework for reliable automated coding in communication research: Applications to health and political communication](#). *Preprint*, arXiv:2509.24841.

A Instruction Template for Error Corrector

We adopt instruction fine-tuning to train the error corrector. At each streaming step, the model receives the current audio chunk, the previously confirmed transcript, and the top- k ASR hypotheses for the newly generated segment. The following instruction template is used during training:

Streaming Error Correction Instruction

Given the audio and the following ASR transcription candidates, predict the correct continuation.

Previous confirmed transcript:
“{previous_transcript}”

K-best candidates for the continuation:
{candidates_str}

Based on the audio, what is the correct text that should be appended after “{previous_transcript}”?

B Analysis of Error Propagation

Table 9 illustrates the error propagation problem in offline-based streaming ASR, where early incorrect prefix selections lead to semantically incoherent hypotheses in subsequent chunks. Without an error corrector, the system becomes trapped in repetitive numeric outputs, while StreamCorrect leverages contextual correction to recover meaningful phrases. This chunk-level comparison highlights how local corrections prevent cascading errors in later decoding steps.

C Variance Analysis

To assess the stability of our latency measurements, we repeat each inference run five times on both AIShell-1 and our industrial Mandarin benchmark, and report the mean and standard deviation of FTL and LTL in Table 4. All runs are conducted on a single NVIDIA H20 GPU under controlled conditions. Specifically, each run is launched only when GPU utilization drops to 0%, with no other process sharing the GPU during inference, ensuring that the reported latencies are not affected by external workload interference.

Table 4 shows that latency measurements are highly stable across runs. Across all three systems, the standard deviation of both FTL and LTL remains below 2ms on both datasets. SimulStreaming exhibits particularly small variation, with standard deviations below 0.5ms for both metrics on both datasets, while StreamCorrect shows slightly

higher variance due to the additional correction step, with standard deviations of at most 1.35ms for FTL and 0.98ms for LTL on AIShell-1. On the industrial benchmark, StreamCorrect remains similarly stable, with standard deviations below 1ms for both metrics. These small fluctuations are negligible relative to the absolute latency values, which are on the order of hundreds of milliseconds, confirming that the latency overhead reported in the main paper is stable across repeated runs rather than an artifact of measurement noise.

D Noise Robustness

To assess noise robustness, we additionally fine-tune the error corrector on self-generated data from the AIShell-1 training set, augmenting each training sample with a noise level uniformly sampled from SNRs of -5 dB, 0 dB, 5 dB, and 10 dB. We then evaluate on the AIShell-1 test set perturbed at the same set of noise levels.

Table 5 shows that StreamCorrect consistently improves over the SimulStreaming baseline across all SNR conditions, with the largest gains observed at the most challenging settings. At SNR-5, CER drops from 77.9% to 70.9%, a 7.0 absolute reduction, and at SNR0 from 54.1% to 49.4%. The relative improvement remains substantial even as the acoustic conditions become cleaner, with CER reduced from 29.7% to 24.2% at SNR10, an 18.5% relative reduction. These improvements come at a modest latency cost, with FTL increasing from 159ms to 221ms and LTL from 214ms to 296ms, indicating that StreamCorrect maintains its noise robustness without compromising the streaming nature of the system.

E SpeechLM Backbones Comparison

As shown in Table 6, we compare Ultravox-Llama-3.2-1B with a text-only Llama-3.2-1B of the same size and a larger SpeechLM, Ultravox-Llama-3.1-8B, as the error corrector backbone. Trained on identical self-generated data, the text-only LLM fails to adapt to the error correction task and over-corrects, resulting in extremely high CER and highlighting the importance of audio conditioning for stable streaming correction. The 8B SpeechLM achieves slightly lower CER (9.6% vs. 9.8%) but incurs higher latency, with LTL reaching 417ms at a 500ms chunk size. Given the marginal gain and increased latency, Ultravox-Llama-3.2-1B provides the best trade-off between accuracy and real-time

Dataset / Metric	WhisperStreaming	SimulStreaming	StreamCorrect
<i>AIShell-1</i>			
FTL (ms) ↓	441.13 ± 0.62	158.79 ± 0.09	220.73 ± 1.35
LTL (ms) ↓	325.47 ± 1.20	214.05 ± 0.50	296.40 ± 0.98
<i>Our industrial data</i>			
FTL (ms) ↓	433.52 ± 0.72	153.65 ± 0.16	213.72 ± 0.91
LTL (ms) ↓	375.21 ± 1.87	213.47 ± 0.27	291.92 ± 0.79

Table 4: Latency stability across five runs on AIShell-1 and our industrial Mandarin benchmark. We report mean $\pm$ standard deviation (ms) under computationally unaware mode. All inference runs are performed on a single NVIDIA H20 GPU.

Metric / SNR	SimulStreaming	StreamCorrect
<i>CER (%) ↓</i>		
SNR-5	77.9	70.9
SNR0	54.1	49.4
SNR5	35.9	35.0
SNR10	29.7	24.2
FTL (ms) ↓	159	221
LTL (ms) ↓	214	296

Table 5: Results on the AIShell-1 test set with different levels of added noise, under a 500ms chunk size.

Models	CER(%)↓	FTL(ms)↓	LTL(ms)↓
Llama-3.2-1B	428.5	200	634
Ultravox-Llama-3.2-1B	9.8	219	295
Ultravox-Llama-3.1-8B	9.6	254	417

Table 6: Impact of backbone model choice on error correction performance. Metrics are computed under a 500ms chunk size on the AIShell-1 test set.

efficiency for streaming error correction.

F Scalability Across Recognizers

Table 7 further demonstrates the decoupling between the error corrector and the offline recognizer. StreamCorrect improves performance across Whisper backbones, reducing CER from 28.7% to 14.9% for Whisper-small and from 19.5% to 11.5% for Whisper-large-v3, indicating strong generalization across backbones.

G Comparison with Qwen3-ASR Streaming Inference

To further evaluate StreamCorrect against recent streaming ASR systems, we compare it with Qwen3-ASR under both offline and streaming inference settings. For Qwen3-ASR streaming inference, we evaluate different configurations by varying the index of chunk when the model starts decoding (u_c) and the number of latest tokens left uncon-

Base Model	SimulStreaming CER(%)↓	StreamCorrect CER(%)↓
whisper-small	28.7	14.9
whisper-medium	21.8	13.2
whisper-large-v3	19.5	11.5
Belle Whisper	16.3	9.8

Table 7: CER comparison at 500ms chunk size across speech recognizers on the AIShell-1 test set.

Model	CER	FTL	LTL
Qwen3-ASR (Offline)	2.8	–	–
Qwen3-ASR ($u_c=1, u_t=1$)	18.9	126	133
Qwen3-ASR ($u_c=2, u_t=1$)	6.8	161	132
SimulStreaming	16.3	158	213
StreamCorrect	9.8	219	295

Table 8: Comparison with Qwen3-ASR on AISHELL-1 test set, under different inference settings. FTL and LTL are reported in milliseconds.

firmed at each chunk (u_t). Results are summarized in Table 8. While Qwen3-ASR offline inference provides a strong upper bound, streaming configurations introduce additional errors under limited context. Notably, $u_c = 2$ provides more initial context by delaying decoding to 1000ms, whereas our main setting begins decoding after 500ms and is therefore more comparable to $u_c = 1, u_t = 1$ in terms of initial latency. Beyond this latency consideration, StreamCorrect targets a different deployment scenario: it allows existing domain-specific offline recognizers to operate in streaming mode without modifying or retraining the underlying ASR model, while the correction module can be developed and optimized independently. This decoupled design is particularly suitable for industrial settings where offline ASR systems are continuously improved and deployed for domain-specific applications.

Chunk	With EC (StreamCorrect)	Without EC (SimulStreaming)	Notes
1	VAD: non-speech		Silent
2	Top-4 Hypotheses:	Top-4 Hypotheses:	Error corrector selects closer hypothesis
	1. er (two)	1. er (two)	
	2. er-shi (twenty)	2. er-shi (twenty)	
	3. san-shi (thirty)	3. san-shi (thirty)	
	4. yi-jiu (nineteen)	4. yi-jiu (nineteen)	
	Selected: ying ×	Selected: er ×	
3	Top-4 Hypotheses:	Top-4 Hypotheses:	Error corrector corrects to meaningful phrase
	1. yu-yu-yan-yu-yan	1. ling-ling-ling	
	2. yu-ci-ci-ci	2. ling-yi-ling-nian	
	3. yu-yu-yan-she	3. ling-ling-yi-nian	
	4. yu-ci-ci-yu	4. ling-ling-yi-de	
	Selected: jing-shan-chu-le ✓	Selected: ling-ling-ling ×	
4	Top-4 Hypotheses:	Top-4 Hypotheses:	Top-1 of StreamCorrect happens to be correct with correct prefix transcript from chunk 3
	1. ci (this)	1. ling-ling	
	2. si (four/die)	2. ling-nian-shi	
	3. si (four)	3. ling-ba-nian	
	4. (empty)	4. ling-yi-nian	
	Selected: ci ✓	Selected: ling-ling ×	
... continuing for remaining chunks ...			
Final Transcription Comparison			
GT	yi-jing shan-chu le-ci qian zhen-dui wai-shang tou-zi fang-di chan de quan-bu xian-zhi lei tiao-kuan		
+EC	ying jing-shan-chu-le ci qian zhen dui wai shang-tou zi fang di-chan de-quan-bu xian zhi-lei tiao kuan kuan		
-EC	er ling-ling ling ling ling-ling-ling-ling-ling ling-ling-ling-ling ling ling ling-ling ling ling-ling ling-ling		

Table 9: Chunk-by-chunk comparison of offline-based streaming ASR systems with and without error correction (EC). The error corrector analyzes context to select semantically appropriate hypotheses rather than blindly choosing among top- k hypotheses. GT = ground truth.