Deploying AtomicGPT for Real Time Meeting Summarization: A Comparative Analysis of Local Processing Strategies

Inhye Park[a,c,d], Janghwan Kim[b,c], Hogeon Seo[c,d,*]
[a]*Daegu University, 201, Daegudae-ro, Gyeongsan-si, Gyeongsangbuk-do, 38453, Korea*
[b]*Hanbat National University, 125, Dongseo-daero, Yuseong-gu, Daejeon, 34158, Korea*
[c]*Korea Atomic Energy Research Institute, 111, Daedeok-daero 989 beon-gil, Yuseong-gu, Daejeon, 34057, Korea*
[d]*University of Science & Technology, 217, Gajeong-ro, Yuseong-gu, Daejeon, 34113, Korea*
**Corresponding author: hogeony@kaeri.re.kr*

***Keywords :** On-premise LLM, Speech-to-Text, Real-time Summarization, AtomicGPT

1. Introduction

Nuclear facilities operate under strict network-isolation requirements to protect sensitive information, which makes cloud-based AI summarization services unsuitable[1, 2]. Nevertheless, accurate documentation of extended, knowledge-intensive expert discussions is critical for safety review and operational decision-making. Although local large language models (LLMs) address data-governance concerns, they introduce a practical challenge: maintaining timely summarization of continuously growing dialogue under the resource constraints of on-premises environments.

Accordingly, this study investigates summarization strategies that balance runtime efficiency and summary quality. This study utilizes AtomicGPT, the language model fine-tuned specifically for the nuclear domain to comprehend specialized terminology and regulatory contexts in Korea Atomic Energy Research Institute. Using this model, we compare four meeting summarization scenarios: full-text accumulation, segment-wise summarization with incremental aggregation, iterative summary refinement, and unconstrained iterative refinement. Using International Atomic Energy Agency (IAEA) panel data, we evaluate these strategies in terms of average runtime and summary quality and identify the most practical approach for secure on-premises deployment in nuclear facilities.

2. Methods

To evaluate meeting summarization in a constrained local environment, we first establish an automated transcription pipeline that converts meeting audio into structured text. As illustrated in Fig. 1, the incoming audio stream is processed by a voice activity detection (VAD) module based on Silero to isolate speech segments. These segments are then routed to a Whisper model [3] for transcription and a Pyannote model [4, 5] for speaker diarization. The synchronized output yields a continuous, time-stamped transcript with speaker attribution.

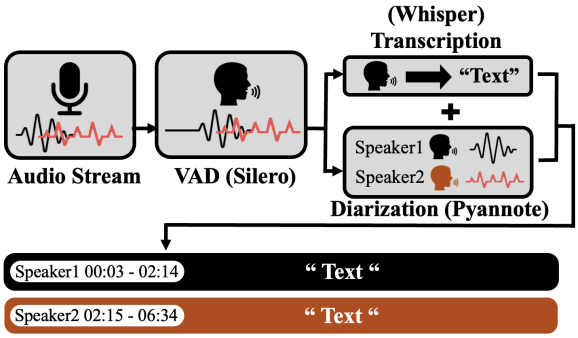

Fig. 1. Architecture of the automated transcription and speaker-diarization workflow.

For the summarization experiments, the transcript is organized at the speaker-turn level, and each speaker turn is treated as one utterance. The core question is then how the local LLM (AtomicGPT) should process these utterances as the meeting progresses. To analyze this, we define three core summarization strategies based on how context is passed between updates, together with an unconstrained variant of iterative refinement for ablation.

2.1. Scenario 1 (S1): Full-Text Accumulation

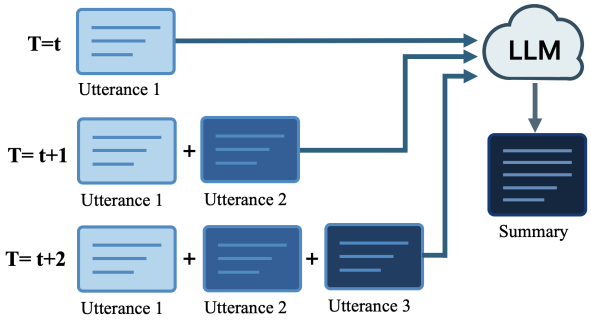

Fig. 2. Scenario 1: Full-text accumulation scheme (T: time). The entire transcript is reprocessed at each update.

In Scenario 1 (S1), whenever a new speaker turn is added, the entire accumulated transcript from the beginning of the meeting is provided to the LLM to regenerate the summary. If n denotes the number of utterances, the input length grows linearly with n, which makes the processing time per update $\mathcal{O}(n)$ and the cumulative processing time $\mathcal{O}(n^2)$. This strategy maximizes contextual access by always referencing the complete discussion history, but it also places the greatest runtime burden on local inference.

2.2. Scenario 2 (S2): Segment-wise Summarization with Incremental Aggregation

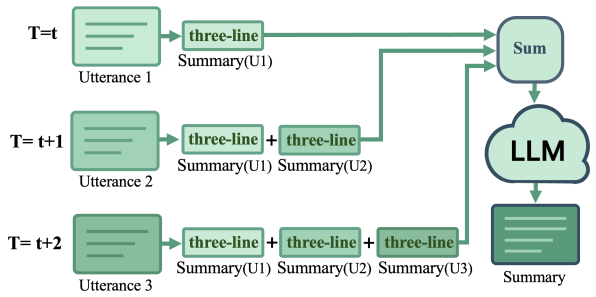

Fig. 3. Scenario 2: Incremental aggregation scheme (T: time). Each speaker turn is summarized independently and re-aggregated at every update.

In Scenario 2 (S2), each speaker-turn utterance is independently compressed into a brief three-line segment summary as it arrives. After each new segment summary is produced, it is appended to the accumulated set, and a global aggregation step synthesizes all segment summaries collected so far into an updated overall meeting summary. Because each online compression step operates on a bounded amount of text, its cost is $\mathcal{O}(1)$ per utterance. The aggregation step processes the growing list of segment summaries, making its per-step cost $\mathcal{O}(k)$, where k is the number of segments processed so far. Consequently, the cumulative cost over n utterances is $\mathcal{O}(n^2)$. However, because the aggregation input consists of compressed three-line segment summaries rather than full transcript content, the per-step input length is significantly shorter, resulting in substantially lower runtimes in practice. This design provides a continuously refreshed overall summary at every speaker turn while keeping the per-step input compact.

2.3. Scenario 3 (S3): Iterative Summary Refinement

In Scenario 3 (S3), only the newly added speaker turn and the most recent summary are provided to the LLM to generate an updated summary. Because the three-line output constraint keeps the carried-over summary short, the input size at each step remains approximately bounded regardless of meeting length. Under this design, the processing time per update is $\mathcal{O}(1)$, and the cumulative processing time is $\mathcal{O}(n)$. This strategy imposes the lowest runtime burden, but it depends on repeated compression of prior content and may gradually discard earlier details.

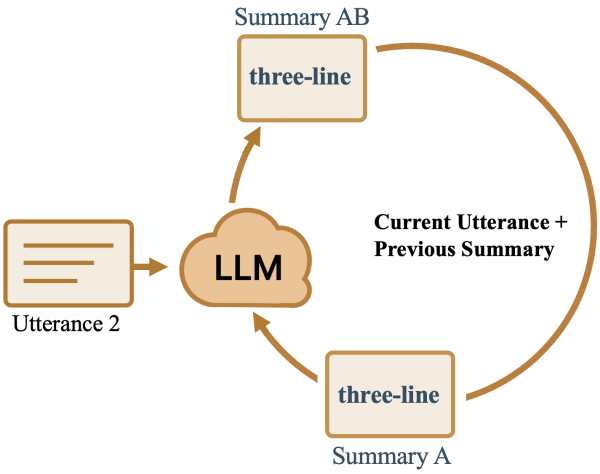

Fig. 4. Scenario 3: Iterative summary refinement scheme (T: time). Each update uses the latest speaker turn and the previous summary.

2.4. Scenario 4 (S4): Unconstrained Iterative Refinement

S4 serves as an ablation study to isolate the effect of the output-length constraint used in S3. It follows the same iterative architecture as S3, using the prior summary and the newest speaker turn to generate an updated summary. However, the three-line output restriction is removed, allowing the summary to expand as the discussion progresses. As a result, the carried-over summary is no longer bounded and can grow with the meeting length. In the worst case, the per-step processing time grows to $\mathcal{O}(n)$, and the cumulative processing time becomes $\mathcal{O}(n^2)$.

3. Experiments

3.1. Experimental Setup

All summarization experiments were conducted in an air-gapped on-premises environment completely isolated from external networks. The hardware and software configuration comprised an Apple M4 Max system with 128 GB of unified memory, macOS Tahoe 26.2, and Python 3.12.12. The deployed model was AtomicGPT, a nuclear-domain-specific model based on EXAONE 4.0, executed locally through the Apple MLX framework. The model used 4-bit quantization for local inference and supported a maximum context length of 32k tokens.

The generation parameters were fixed across all scenarios: a maximum of 4,096 new tokens, a temperature

Table I: Comparative analysis of summary quality and average runtime

	G-Eval Metric					Efficiency
Scenario	**Faithfulness**	**Coverage**	**Conciseness**	**Coherence**	**Speaker Balance**	**Average Runtime (min)**
S1	4.90 ± 0.17	4.43 ± 0.23	5.00 ± 0.00	5.00 ± 0.00	5.00 ± 0.00	60.12 ± 7.30
S2	4.43 ± 0.23	4.13 ± 0.51	4.90 ± 0.17	5.00 ± 0.00	4.33 ± 0.58	22.09 ± 2.09
S3	2.67 ± 0.58	2.00 ± 0.00	4.10 ± 0.85	4.10 ± 0.35	2.20 ± 0.17	4.04 ± 0.25
S4	2.57 ± 0.42	3.57 ± 0.42	2.87 ± 0.61	3.33 ± 0.74	3.43 ± 0.42	62.13 ± 12.84

* Quality metrics are scored on a 5-point scale via G-Eval.

of 0.3, top-p of 0.9, and a repetition penalty of 1.05. A common system prompt was used across scenarios to position the model as an expert debate analyst focusing on key arguments, speaker positions, and policy implications. Scenario-specific prompt instructions differed only in the summarization procedure: S1 used full-text summarization, S2 used three-line segment compression with incremental aggregation at every update, S3 used iterative refinement with a three-line output constraint, and S4 used the same iterative refinement procedure without the length constraint.

3.2. Evaluation Dataset

The evaluation data were derived from an open panel discussion hosted by the International Atomic Energy Agency (IAEA) on nuclear energy and climate change mitigation. The discussion includes multiple speakers, domain-specific terminology, and policy-oriented argumentation, making it suitable for evaluating nuclear-domain summarization. For the summarization experiments, the transcript was organized at the speaker-turn level, and each speaker turn was treated as one utterance. The evaluated session contained 34 utterances spanning approximately 72 minutes and involved five primary speakers, including the moderator and domain experts.

3.3. Results and Analysis

Table I summarizes the G-Eval results and average runtime for the four scenarios. Based on the arithmetic mean of the five G-Eval metrics, S1 achieved the highest overall quality score (4.87), followed by S2 (4.56). S4 (3.15) slightly outperformed S3 (3.01) in overall average because removing the three-line constraint improved Coverage and Speaker Balance, although it reduced Conciseness and Coherence. S1 obtained perfect scores in Coherence and Speaker Balance and the highest Faithfulness score, which reflects the benefit of always referencing the full transcript. S2 also maintained strong quality, although compressing each speaker turn before final aggregation slightly reduced Faithfulness, Coverage, and Speaker Balance relative to S1.

In terms of runtime, S3 was the fastest strategy, requiring an average of 4.04 minutes to process the full session. S2 followed at 22.09 minutes, whereas S1 and S4 required 60.12 and 62.13 minutes, respectively. These results show a clear trade-off between quality and runtime. S1 preserved the richest context and produced the strongest summaries, but it imposed a substantial runtime cost because the input grew with the full meeting history.

S3 minimized runtime by carrying forward only a heavily constrained summary, but the fixed three-line compression reduced Faithfulness, Coverage, and Speaker Balance. In the nuclear domain, where critical technical details may be involved, this could be a concern: once a detail is omitted during the overwrite, it may not be recoverable in subsequent updates. The iterative carry-over mechanism allows S3 to maintain some continuity across updates, but the low Faithfulness (2.67) and Coverage (2.00) scores suggest that finer details could still be lost, indicating that S3 may be less effective at capturing the full scope of the discussion compared to the other strategies. By contrast, S2 preserves each speaker turn as an independent segment summary, and its Faithfulness score of 4.43 points to a level of reliability that could be more suitable for contexts requiring accurate summarization.

S4 was introduced to test whether relaxing the output-length constraint could recover information lost in S3. The results show that removing the constraint improved Coverage and Speaker Balance relative to S3, but it also reduced Conciseness and Coherence and increased runtime to a level comparable to S1. In other words, the efficiency advantage of iterative refinement depended heavily on strict output control.

Taken together, S2 provided the most balanced trade-off. Its average runtime was substantially lower than that of S1, yet its summary quality remained comparatively strong. Because it compresses each speaker turn during the meeting and re-aggregates the accumulated segments at every update, S2 provides a continuously refreshed overall summary throughout the session while maintaining competitive runtime. This makes S2 the most practical compromise among the evaluated strategies when both runtime and summary quality are considered.

4. Conclusions

This study examined the feasibility of secure local meeting summarization for the nuclear sector using AtomicGPT and compared four processing strategies under on-premises constraints. The results reveal a clear trade-off between summary quality and runtime.

Full-text accumulation (S1) achieved the highest summary quality but required long runtimes because the entire discussion history was repeatedly reprocessed. Iterative refinement with a strict three-line limit (S3) delivered the shortest runtime, but its aggressive compression substantially reduced Faithfulness, Coverage, and Speaker Balance. Removing that constraint (S4) partially recovered Coverage and Speaker Balance, but it also weakened Conciseness and Coherence and increased runtime to a level similar to S1. Segment-wise summarization with incremental aggregation (S2) provided the best overall balance, combining relatively strong summary quality with a much lower average runtime than S1.

Future work will focus on improving the segment-compression stage of S2 so that more critical content can be preserved before final aggregation. In particular, saliency-based filtering will be investigated to retain important expert statements while keeping runtime manageable. These refinements will then be validated in operational nuclear-plant environments.

REFERENCES

[1] Nuclear Safety and Security Commission (NSSC), *Basic Guidelines for Information Security*. NSSC Directive No. 243, 2026.

[2] U.S. Nuclear Regulatory Commission (NRC), "Cyber security programs for nuclear power reactors," tech. rep., 10 CFR 73.54; Regulatory Guide 5.71, 2010.

[3] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," 2022.

[4] A. Plaquet and H. Bredin, "Powerset multi-class cross entropy loss for neural speaker diarization," in *Proc. INTERSPEECH 2023*, 2023.

[5] H. Bredin, "pyannote.audio 2.1 speaker diarization pipeline: principle, benchmark, and recipe," in *Proc. INTERSPEECH 2023*, 2023.