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

Inhye Park[1,4] · Janghwan Kim[2,3] · Hogeon Seo[3,4]*

[1] Daegu University [2] Hanbat National University [3] Korea Atomic Energy Research Institute [4] University of Science & Technology

* Corresponding author : hogeony@kaeri.re.kr

Transactions of the Korean Nuclear Society Spring Meeting · Jeju · May 8, 2026

KAERI 한국원자력연구원 Korea Atomic Energy Research Institute

UST 과학기술연합대학원대학교 UNIVERSITY OF SCIENCE & TECHNOLOGY

Contents

1. Introduction

1-1. Background & Motivation

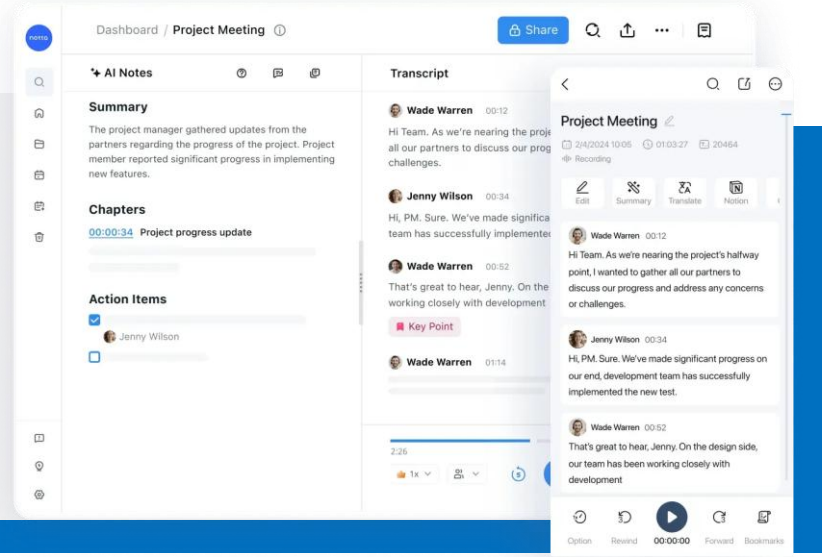
1-2. Objectives & Contributions

1-1. Background & Motivation

AI-based meeting-recording services are now widely available

- Real-time transcription and summarization are offered by commercial platforms
  (Otter.ai, Microsoft Copilot, Zoom AI Companion, Naver Clova Note, etc.)
- Most are cloud-based - transcripts and audio are processed on external servers

→ ***on-premise deployment is required for compliance***

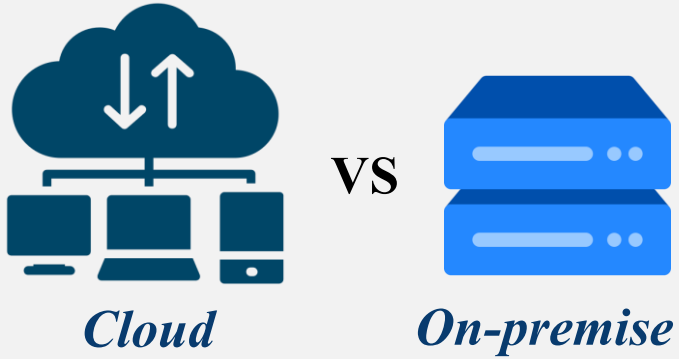


Nuclear facilities are security-critical sites

Cloud-based AI summarization services pose data leakage risks to external servers

All operational data must remain on-site within the facility

→ ***On-premise LLMs offer a viable regulation-compliant approach***



**On-premise : all compute and data remain on-site, fully isolated from external networks*

1-2. Objectives & Contributions

*S : Scenario

How can progressive summarization balance quality and runtime under on-premise resource constraints?

→ Comparison of four strategies (S1-S4)

IAEA ‘Taking Stock of Nuclear Energy's Role in Climate Action’ panel discussion

2. Summarization Strategies

2-1. Transcription Pipeline

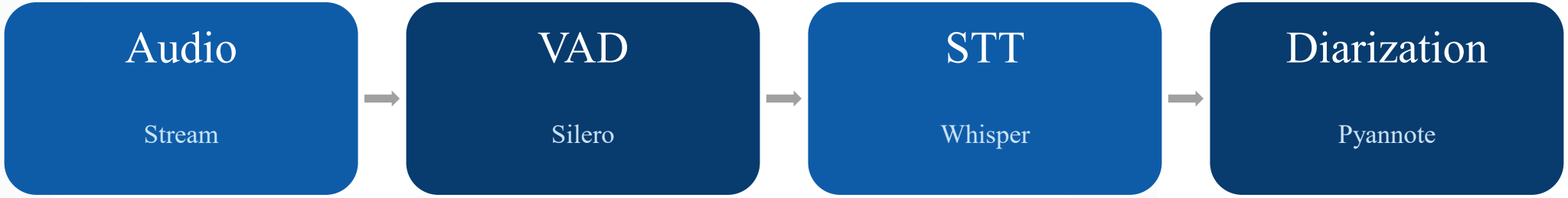
2-2. Summarization Strategies

- S1 · Full-Text Accumulation
- S2 · Segment + Aggregation
- S3 · Iterative Refinement
- S4 · Ablation of S3

2-1. Transcription Pipeline

** VAD : Voice Activity Detection*

** STT : Speech-to-Text*



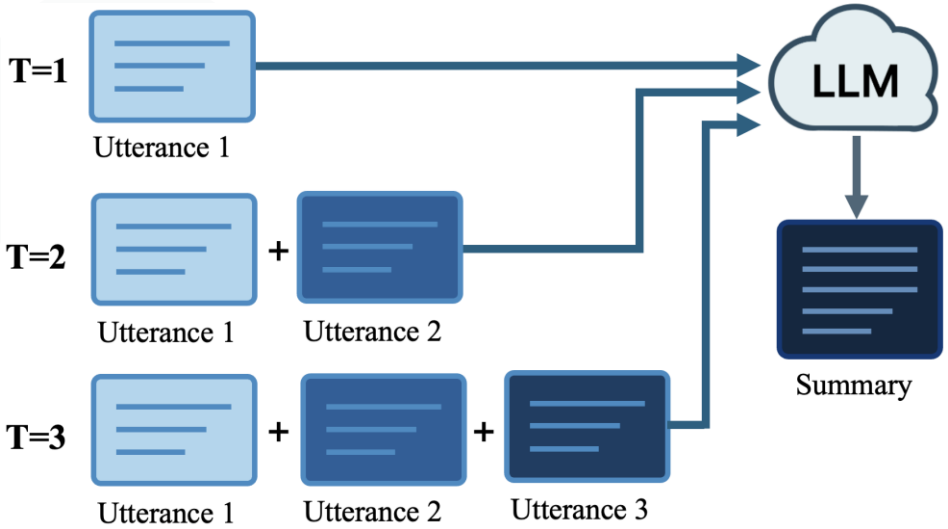
Time-stamped, speaker-attributed utterances

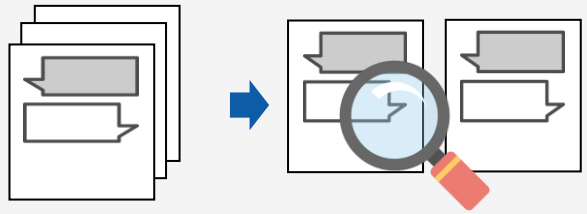
- Silero VAD isolates speech regions → Whisper [Radford et al., 2022] performs STT
- Pyannote [Bredin, 2023] attributes each segment to a speaker
- Each speaker-turn utterance becomes the unit of input to the local LLM

→ Study focus : how the LLM consumes these utterances as the meeting unfolds

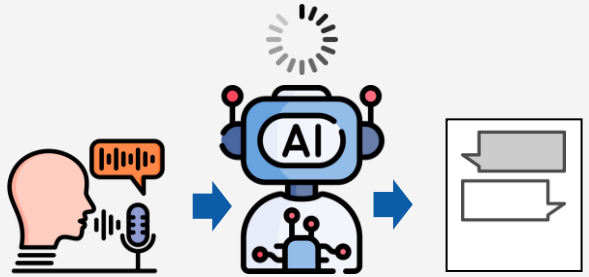
2-2. S1 · Full-Text Accumulation

Every new utterance → Re-feed the whole transcript and regenerate the summary from scratch





Always references the complete discussion history → maximal contextual access



Places the greatest per-step runtime burden on local inference

Concept

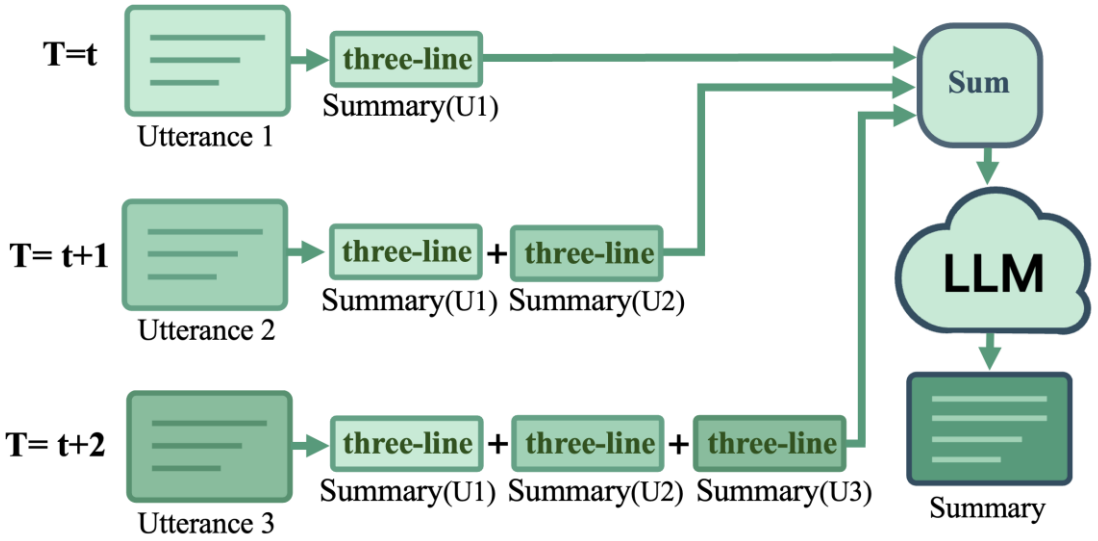
Whenever a new utterance arrives, the entire accumulated transcript from the start of the meeting is provided to the LLM.

The summary is regenerated from scratch on every update.

Input length therefore grows linearly with the number of utterances n.

2-3. S2 · Segment-wise Aggregation

Compress each utterance into a 3-line segment summary, then aggregate the accumulated segments



Concept

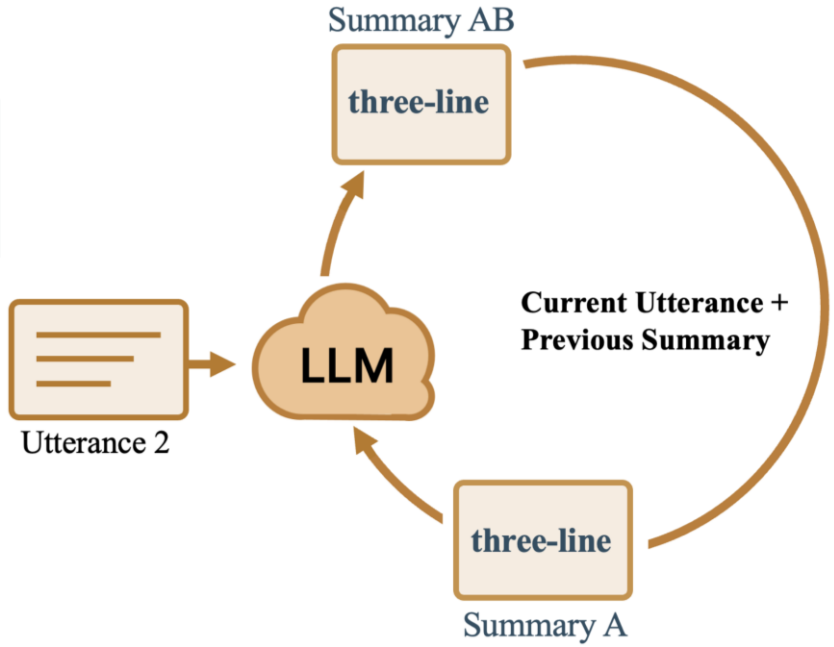
Each arriving utterance is independently compressed into a brief 3-line segment summary.

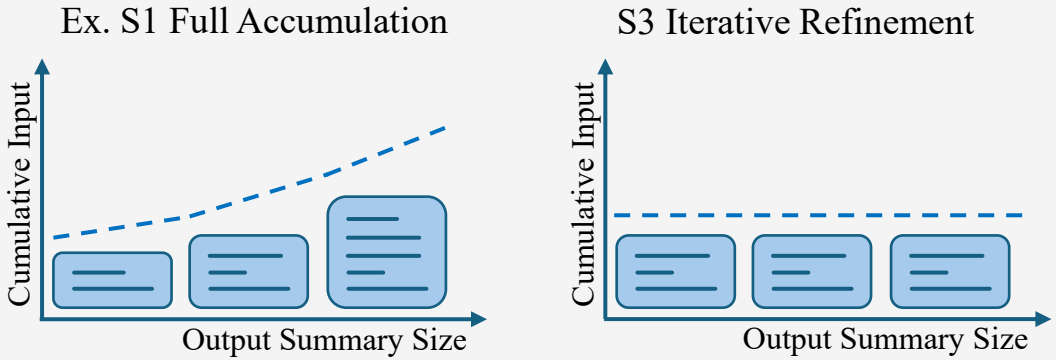
At every update, all segment summaries produced so far are aggregated into an updated global summary.

A refreshed overall summary is therefore available at every speaker turn.

2-4. S3 · Iterative Refinement

Carry only the most recent 3-line summary forward together with the newly arrived utterance





Lightest update scheme among the four strategies
→ Constant memory footprint

Concept

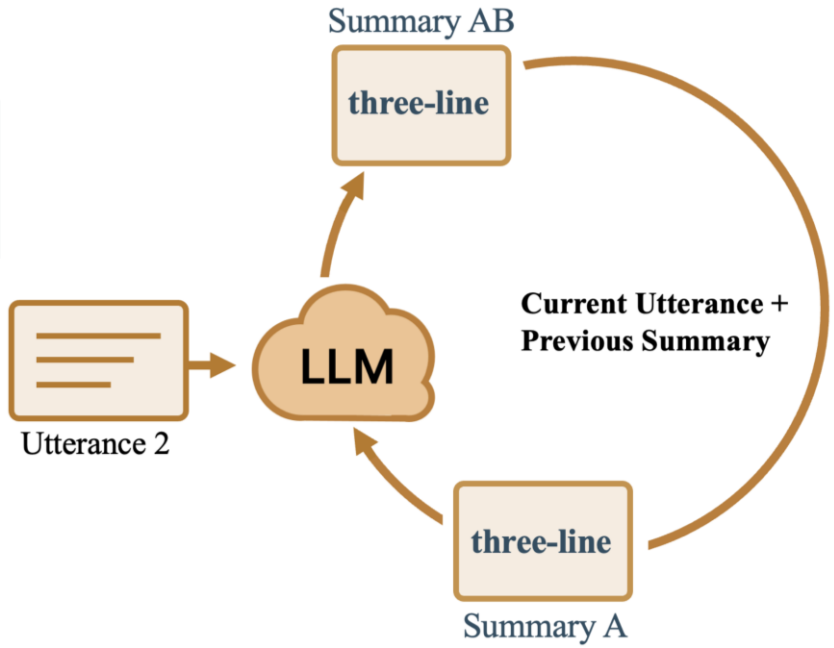
Two inputs only : ① **the 3-line summary produced at the previous step**, and ② **the newly added utterance.**

The LLM merges the two and outputs an updated 3-line summary.

The 3-line output cap keeps the carried-over summary short, so input size stays bounded regardless of meeting length.

2-5. S4 · Ablation of S3

Same iterative refinement as S3, but without the 3-line output constraint - to isolate the role of



Concept	Identical iterative architecture to S3: ① **the previous summary,** ② **the newly added utterance** **Key difference: the 3-line output cap is removed** The carried-over summary is no longer bounded → grows with meeting length

3. Experimental Setup

3-1. Hardware & Model

3-2. Evaluation Data & Metrics

3-1. Hardware & Model

HW / SW

Apple M4 Max

128 GB unified memory

macOS Tahoe 26.2 · Python 3.12.12

Air-gapped environment

Model

- Nuclear-domain large language model developed by KAERI
- Trained on nuclear-domain papers, glossaries, regulatory documents, and technical reports
- Deployment : Apple MLX framework · 4-bit quantization · 32k token context

3-3. Evaluation Method - LLM-Based Evaluation Framework (G-Eval)

What is G-Eval?

An automated scoring framework that employs an LLM as a judge to assess the quality of generated text.

How does it work?

- Input : Source transcript + generated summary
- The summary under evaluation and the original meeting transcript are jointly provided to the LLM
- For each criterion, a detailed scoring rubric is embedded in the prompt; the LLM assigns a score from 1 to 5 per criterion

* *G-Eval : LLM-based Evaluation Framework*

3-4. Evaluation Method - Evaluation Metrics Used

Evaluation Metrics Used

Metric	Assessment Focus
Faithfulness	Factual consistency with the source and absence of hallucination
Coverage	Whether key information - technical issues, expert opinions, and core conclusions - is preserved without omission
Conciseness	Efficiency of information delivery, free of unnecessary repetition or redundancy
Coherence	Logical structure and naturalness of sentence-level flow at a professional-report level
Speaker Balance	Balanced reflection of multiple participants views, without bias toward any single speaker

4. Results & Analysis

4-1. G-Eval & Runtime

4-2. Quality ↔ Runtime Trade-off

4-1. Experimental Results - G-Eval Framework

Table I: Comparative analysis of summary quality and average runtime

Scenario	G-Eval Metric				
	Faithfulness	Coverage	Conciseness	Coherence	Speaker Balance
S1	4.90 ± 0.17	4.43 ± 0.23	5.00 ± 0.00	5.00 ± 0.00	5.00 ± 0.00
S2	4.43 ± 0.23	4.13 ± 0.51	4.90 ± 0.17	5.00 ± 0.00	4.33 ± 0.58
S3	2.67 ± 0.58	2.00 ± 0.00	4.10 ± 0.85	4.10 ± 0.35	2.20 ± 0.17
S4					

Per-Utterance Processing Time		
Min (s)	Mean (s)	Max (s)
16.9 ± 3.9	106.1 ± 12.9	276.8 ± 32.5

S1	4.90 ± 0.17	4.43 ± 0.23	5.00 ± 0.00	5.00 ± 0.00	5.00 ± 0.00
S2	4.43 ± 0.23	4.13 ± 0.51	4.90 ± 0.17	5.00 ± 0.00	4.33 ± 0.58

S1 · Full-Text Accumulation

- Average G-Eval score across 5 metrics : 4.87 - highest among all scenarios
- Perfect scores (5.00) on Coherence and Speaker Balance → benefit of always referencing full context
- Per-utterance processing time : Min 16.9s · Mean 106.1s · Max 276.8s
- Input length grows as utterances accumulate → processing time increases progressively

4-1. Experimental Results - G-Eval Framework

Table I: Comparative analysis of summary quality and average runtime

Scenario	G-Eval Metric				
	Faithfulness	Coverage	Conciseness	Coherence	Speaker Balance
S1	4.90 ± 0.17	4.43 ± 0.23	5.00 ± 0.00	5.00 ± 0.00	5.00 ± 0.00
S2	4.43 ± 0.23	4.13 ± 0.51	4.90 ± 0.17	5.00 ± 0.00	4.33 ± 0.58
S3	2.67 ± 0.58	2.00 ± 0.00	4.10 ± 0.85	4.10 ± 0.35	2.20 ± 0.17
S4					

Per-Utterance Processing Time		
Min (s)	Mean (s)	Max (s)
15.1 ± 5.7	39.0 ± 3.7	72.3 ± 2.8

S1	4.90 ± 0.17	4.43 ± 0.23	5.00 ± 0.00	5.00 ± 0.00	5.00 ± 0.00
S2	4.43 ± 0.23	4.13 ± 0.51	4.90 ± 0.17	5.00 ± 0.00	4.33 ± 0.58

S2 · Segment + Aggregation

- Average quality : 4.56 - only ~0.31 lower than S1
- Per-utterance processing time : Min 15.1s · Mean 39.0s · Max 72.3s — about 1/3 of S1
- Compresses each utterance into a 3-line segment summary, keeping per-step input compact while continuously refreshing the overall summary

4-1. Experimental Results - G-Eval Framework

Table I: Comparative analysis of summary quality and average runtime

Scenario	G-Eval Metric				
	Faithfulness	Coverage	Conciseness	Coherence	Speaker Balance
S1	4.90 ± 0.17	4.43 ± 0.23	5.00 ± 0.00	5.00 ± 0.00	5.00 ± 0.00
S2	4.43 ± 0.23	4.13 ± 0.51	4.90 ± 0.17	5.00 ± 0.00	4.33 ± 0.58
S3	2.67 ± 0.58	2.00 ± 0.00	4.10 ± 0.85	4.10 ± 0.35	2.20 ± 0.17
S4	2.57 ± 0.42	3.57 ± 0.42	2.87 ± 0.61	3.33 ± 0.74	3.43 ± 0.42

Per-Utterance Processing Time		
Min (s)	Mean (s)	Max (s)
4.1 ± 0.5	7.1 ± 0.4	14.1 ± 1.2

S3	2.67 ± 0.58	2.00 ± 0.00	4.10 ± 0.85	4.10 ± 0.35	2.20 ± 0.17
S4	2.57 ± 0.42	3.57 ± 0.42	2.87 ± 0.61	3.33 ± 0.74	3.43 ± 0.42

S3 · Iterative Refinement

- Per-utterance processing time : Min 4.1s · Mean 7.1s · Max 14.1s - shortest among all scenarios
- Inputs only the previous summary and the new utterance → input size remains bounded regardless of meeting length
- However, average quality drops to 3.01, lower than S1 and S2
- Notably low Coverage (2.00) and Speaker Balance (2.20) - early-speaker contributions and detailed discussion appear to be progressively lost through repeated compression

4-1. Experimental Results - G-Eval Framework

Table I: Comparative analysis of summary quality and average runtime

Scenario	G-Eval Metric				
	Faithfulness	Coverage	Conciseness	Coherence	Speaker Balance
S1	4.90 ± 0.17	4.43 ± 0.23	5.00 ± 0.00	5.00 ± 0.00	5.00 ± 0.00
S2	4.43 ± 0.23	4.13 ± 0.51	4.90 ± 0.17	5.00 ± 0.00	4.33 ± 0.58
S3	2.67 ± 0.58	2.00 ± 0.00	4.10 ± 0.85	4.10 ± 0.35	2.20 ± 0.17
S4	2.57 ± 0.42	3.57 ± 0.42	2.87 ± 0.61	3.33 ± 0.74	3.43 ± 0.42

Per-Utterance Processing Time		
Min (s)	Mean (s)	Max (s)
13.0 ± 0.4	109.6 ± 27.8	259.4 ± 57.5

S3	2.67 ± 0.58	2.00 ± 0.00	4.10 ± 0.85	4.10 ± 0.35	2.20 ± 0.17
S4	2.57 ± 0.42	3.57 ± 0.42	2.87 ± 0.61	3.33 ± 0.74	3.43 ± 0.42

S4 · Ablation of S3

- Identical to S3 except that the 3-line output constraint is removed
- Per-utterance processing time: Min 13.0s · Mean 109.6s · Max 259.4s — comparable to S1
- Changes after removing the constraint:

  Coverage: 2.00 → 3.57 (improved)

  Speaker Balance: 2.20 → 3.43 (improved)

  Conciseness: 4.10 → 2.87 (degraded)

  Coherence: 4.10 → 3.33 (degraded)

4-2. Overall Analysis - Quality-Runtime Trade-off

Scenario	Quality	Runtime	Characteristic
S1 · Full-Text Accumulation	Highest	Longest	Quality-first; impractical for real-time
S2 · Segment + Aggregation	***Near-highest***	***Reduced***	***Balanced — final recommendation***
S3 · Iterative Refinement	Low	Shortest	Speed-first; detail loss
S4 · Ablation of S3	Low	Longest	Efficiency collapses when constraint removed

Key Findings

- S1 and S3 represent quality and speed extremes; S4 confirms that strict output control is the source of S3's efficiency
- S2 maintains 4.56 quality while reducing runtime to ~1/3 of S1 → Pareto-optimal in the quality-efficiency trade-off
- Most practical strategy for real-time meeting summarization in air-gapped environments

5. Conclusions and Future Works

Conclusions

- Under on-premise constraints, AtomicGPT was used to compare four summarization strategies (S1–S4)
- A clear quality ↔ runtime trade-off was observed across strategies
- S2 (Segment-wise + Incremental Aggregation) provides the best overall balance
- Ablation (S4) confirms that the weakness of S3 comes from the 3-line output cap, not iterative refinement itself

Future Works

- Saliency-based filtering in the segment-compression stage to preserve critical expert statements
- Hybrid strategy combining S1 and S2 - retain important utterances in full while compressing peripheral ones

Thank you

Questions & Discussion

Inhye Park[14] · Janghwan Kim[23] · Hogeon Seo[34]*

[1] Daegu University [2] Hanbat National University [3] Korea Atomic Energy Research Institute [4] University of Science & Technology

* Corresponding author : hogeony@kaeri.re.kr

KAERI 한국원자력연구원 Korea Atomic Energy Research Institute

UST 과학기술연합대학원대학교 UNIVERSITY OF SCIENCE & TECHNOLOGY