Structured Lightweight Speech Summarization with On-Device LLMs and Retrieval-Augmented Refinement

Xu Gu[§], Qun Wang[§], William Pan [§], and Xuhui Chen [§] Hao Yue[£], and Rose Qingyang Hu[‡]
[§]Department of Computer Science, San Francisco State University, San Francisco, CA, 94132
[£]Department of Computer Science and Engineering, University of California, Santa Cruz, CA, 95064
[‡]Bradley Department of Electrical and Computer Engineering,
Virginia Polytechnic Institute and State University, Blacksburg, VA 24061
Emails: xgu@sfsu.edu, claudqunwang@ieee.org, william910122@gmail.com tracychen@sfsu.edu,
hyue11@ucsc.edu, rosehu@vt.edu

***Abstract*—Motivated by the growing need for private, low-latency language processing at the edge, we present an on-device speech summarization system that enables users to convert long-form spoken content into queryable summaries, eliminating the need for cloud services. Our system integrates Whisper for transcription, lightweight on-device Large Language Models (ODLLMs) for summarization, and a retrieval-augmented question answering (QA) module for post-hoc interaction. To address the limitations of ODLLM models with small context windows and constrained memory, we introduce a recursive refinement strategy that segments transcripts into overlapping chunks and conditions each summary on both local context and prior outputs. With supporting documents, the system further enhances understanding via semantic retrieval from a local vector store. Final outputs are serialized into a four-field summary schema to support downstream indexing and interactive retrieval. Experimental results demonstrate that this pipeline runs efficiently and produces coherent summaries suitable for QA applications.**

***Index Terms*—On-device language models, Speech summarization, Whisper ASR, Recursive refinement, RAG, Low-resource inference**

I. Introduction

In an era of information overload, audio content such as lectures, meetings, and interviews plays an increasingly critical role across educational, professional, and organizational settings. However, the unstructured and ephemeral nature of spoken data poses significant challenges for efficient information access and summarization. Manual note-taking is not only time-consuming but also error-prone.

The rise of large language models (LLMs) has enabled high-quality abstractive summarization by capturing contextual nuances and generating coherent narratives from long-form text. These capabilities have accelerated the development of speech summarization tools, especially when transcribed speech can be processed via cloud-hosted LLMs [1]. However, for many real-world use cases, such as sensitive meetings, corporate communications, or healthcare conversations, outsourcing data to external servers is not a viable option. Furthermore, cloud-hosted LLMs often require high-bandwidth connections and introduce unpredictable delays, limiting their usability in time-critical or offline scenarios [2] [3].

Deploying LLMs directly on local hardware offers a promising path forward for enhancing privacy and accessibility [4]. Small-scale LLMs with fewer than 10 billion parameters, such as Phi [5], Gemma, and TinyLlama (0.5B–3B), offer viable alternatives when paired with efficient runtimes and prompt tuning [6]. Prior work [7] has focused on large-scale end-to-end speech summarization using multi-stage training and Q-Former alignment, while systems like T5-small-gTTS [8] demonstrate compact document summarization with audio output, but without support for spoken input or on-device execution.

Nevertheless, small LLMs come with notable limitations: they support shorter input contexts, may generate less reliable outputs, and are more susceptible to hallucinations [9]. Moreover, evaluating summarization quality is challenging. Traditional metrics (ROUGE [10], BLEU [11]) offer lexical overlap scores but miss semantic and factual alignment. Embedding-based metrics (e.g., BERTScore [12], MoverScore [13]) better capture meaning but lack interpretability.

To address these challenges, we propose a lightweight and modular Voice Interpretation and Brief Extraction (VIBE) system that enables efficient on-device summarization of transcribed speech. The system is optimized for consumer-grade hardware to eliminate reliance on external services and better preserve user privacy. We adopt a recent LLM-based evaluation framework that uses multi-step prompting to assess faithfulness, completeness, and conciseness of the proposed system [14].

Our contributions are as follows:

- We design and implement a modular on-device speech-to-summary pipeline that supports structured summarization and retrieval-based QA.
- We conduct a comprehensive evaluation of 17 sub-3B LLMs under realistic hardware and context constraints using fine-grained LLM-based scoring.
- We analyze trade-offs between accuracy, latency, and conciseness, offering practical guidance for future on-device

summarization systems.

II. System Model

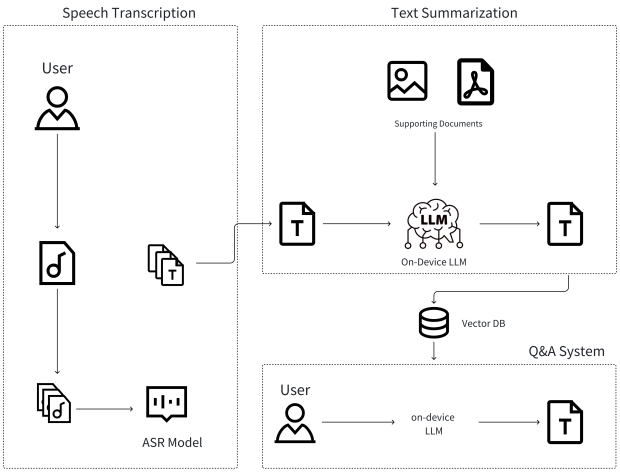

Fig. 1. VIBE System Workflow: Transcription, Summarization, and Q&A

The architecture of VIBE is shown in Figure 1. The system consists of three main stages: speech transcription, text summarization, and retrieval-augmented question answering. In the first stage, users either upload audio recordings or record live audio through the web interface. The audio is then transcribed into text using the Whisper automatic speech recognition (ASR) model developed by OpenAI [15]. This transcription is segmented and prepared as input for summarization. In the second stage, the transcript is summarized using lightweight ODLLM with our Recursive Refine Summarization design. Supporting documents such as images or PDFs can optionally be provided to enrich the summarization context. Finally, in the third stage, the generated summaries are embedded and stored in a vector database to enable semantic retrieval. Users can interact with the Q&A system by asking questions, and the system responds using retrieved context and local LLM inference. This modular pipeline ensures low-latency, privacy-preserving summarization and question answering, all performed on consumer-grade hardware.

A. Metrics for Summarization Performance

To effectively evaluate the quality of summaries generated by ODLLM, we adopt three complementary metrics: **Faithfulness**, **Completeness**, and **Conciseness** [14]. These dimensions capture different yet essential aspects of summary quality and are commonly used in recent research on LLM-based summarization evaluation.

We first evaluate the factual accuracy of each sentence in a model-generated summary. Given the source article (input text) and a generated summary, the measure model is prompted to classify each summary sentence into one of nine predefined factuality categories:

- **no error**, **out-of-context error**, **entity error**, **predicate error**, **circumstantial error**, **grammatical error**, **coreference error**, **linking error**, and **other error**.

Along with the classification, the measure model also returns a concise explanation for each judgment. The complete task is structured as a sentence-level categorization problem. Using the results, we compute the **Faithfulness** score as the proportion of summary sentences that are classified as **no error**. **Faithfulness** $\mathcal{F}$ is defined as:

$$\mathcal{F}(D, S) = \frac{|\mathcal{S}_{\text{fact}}|}{|S|}, \tag{1}$$

where $S = \{s_1, \ldots, s_N\}$ is the set of summary sentences and $\mathcal{S}_{\text{fact}} \subseteq S$ is the subset marked as **no error** [14]. It assesses the factual consistency of the summary with respect to the source document. A summary is considered faithful if it does not introduce hallucinated or incorrect information. This is a critical requirement for real-world summarization systems, as factual errors may mislead readers or distort the intended meaning of the original content.

We then evaluate how well the generated summary captures the essential information present in the reference summary. This involves two substeps: **Keyfact Extraction**: Using the reference summary, we apply the measure model to extract a list of up to 16 *key facts*. Each key fact is a standalone factual statement containing at most 2–3 entities [14] [16]. **Keyfact Matching**: The generated summary is compared against each extracted key fact. For every key fact, the measure model determines whether it can be inferred from the summary (**True/False**) and, if so, identifies the specific summary sentence(s) supporting it [14]. The alignment results are represented as a bipartite graph $\mathcal{M} = (K, S, E)$, where $K = \{k_1, \ldots, k_M\}$ is the set of key facts, S is the summary, and each edge $(k, s) \in E$ denotes alignment between key fact k and summary sentence s. Based on this, we compute two scores: **Completeness** and **Conciseness**. **Completeness** measures the proportion of key facts covered by the summary, while **Conciseness** reflects the proportion of summary sentences that successfully convey at least one key fact. **Completeness** $\mathcal{C}_{\text{pl}}$ is given as:

$$\mathcal{C}_{\text{pl}}(K, S) = \frac{|\{k \mid \exists s, (k, s) \in E\}|}{|K|}. \tag{2}$$

It measures the extent to which the summary covers the key information from the source text. A complete summary should include all essential facts or points to preserve the core message. Incomplete summaries may omit important context, reducing their usefulness [14]. **Conciseness** $\mathcal{C}_{\text{cn}}$ is defined as

$$\mathcal{C}_{\text{cn}}(K, S) = \frac{|\{s \mid \exists k, (k, s) \in E\}|}{|S|}. \tag{3}$$

It evaluates how efficiently the summary conveys the key information. A concise summary avoids redundancy and excessive verbosity, using the fewest sentences necessary to express the main ideas. This is particularly valuable for on-device applications, where output length and inference time are often constrained [14].

B. Problem Statement

Let $\mathbf{x} \in \mathbb{R}^n$ denote the raw audio waveform, $\mathcal{W}(\cdot)$ is the ASR function implemented by Whisper-tiny, and $\mathbf{t} = \mathcal{W}(\mathbf{x}) =$

$(w_1, \ldots, w_{|T|})$ is the tokenized transcript. If a set of supporting documents $\mathcal{D} = \{d_1, \ldots, d_{|\mathcal{D}|}\}$ is uploaded, each document is chunked into passages $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ and embedded as $\mathbf{v}_i = \mathcal{E}(c_i) \in \mathbb{R}^d$ using a sentence-transformer, yielding a vector database $\mathcal{V} = \{\mathbf{v}_i\}_{i=1}^{|\mathcal{C}|}$.

We aim to produce a structured summary $\mathbf{S} = (S_{\text{setting}}, S_{\text{topic}}, S_{\text{key}}, S_{\text{body}})$ that maximizes factual consistency while respecting on-device resource limits. Formally,

$$\min_{\theta} \ \mathcal{L}_{\text{sum}}\big(\mathbf{S}, \mathbf{t}, \mathcal{D}\big) \tag{4}$$

Here θ denotes the parameters of the chosen ODLLM model; $\mathcal{L}_{\text{sum}}$ can be instantiated as the negative metrics' score corresponding to Faithfulness ($\mathcal{F}$), Completeness ($\mathcal{C}_{\text{pl}}$) and Conciseness ($\mathcal{C}_{\text{cn}}$) defined in Eq. 1-3, so $\mathcal{L}_{\text{sum}} = -\big(\lambda_1\mathcal{F} + \lambda_2\mathcal{C}_{\text{pl}} + \lambda_3\mathcal{C}_{\text{cn}}\big)$.

Problem (4) is tackled by a cascade of three modules: (1) ASR: $\mathbf{x} \xrightarrow{\mathcal{W}} \mathbf{t}$. (2) Retrieval (optional): $\mathbf{q} = \mathcal{Q}(\mathbf{t}) \rightarrow \text{ANN}(\mathbf{q}, \mathcal{V}) \rightarrow \mathcal{R}(\mathbf{t}, \mathcal{D})$. (3) LLM Summarizer: $\mathcal{S}_\theta\big(\mathbf{t}, \mathcal{R}\big) \rightarrow \mathbf{s}$. Each module's complexity and contribution are analyzed next.

III. Methodology

A. Whisper Integration and Parallel Transcription for Speech-to-Text Processing

For the speech transcription component, we integrated Whisper, which provides several model variants of increasing size and accuracy: `tiny`, `base`, and `small`. To meet on-device performance constraints, we conducted a comparison of these models in terms of both inference speed and transcription quality.

To quantify the trade-off between accuracy and latency under device-level constraints, we evaluated all three checkpoints on **251** clips sampled from the Mozilla *Common Voice* corpus and measured their Word Error Rate (WER) [17].

$$\text{WER} = \frac{\text{Substitutions} + \text{Deletions} + \text{Insertions}}{\text{Total Words in Reference}}. \tag{5}$$

The experiment reveals two key trends: (1) Differences in WER among `tiny`, `base`, and `small` are not statistically significant.(2) Inference time scales almost linearly with model size, making `tiny` substantially faster than the larger checkpoints while preserving accuracy. Consequently, the `tiny` model is chosen as the default on-device ASR engine; its output latency remains well within our system's real-time budget while delivering transcription quality comparable to the larger variants.

To further enhance performance, the input audio is preprocessed and segmented into smaller chunks based on silence duration. These chunks are then processed in parallel using multithreading. Once transcribed individually, the results are aggregated to form the final transcript. This parallel processing strategy reduces latency and makes the transcription process more efficient on consumer-grade hardware.

B. Recursive Refine Summarization

Figure 2 illustrates our end-to-end, on-device speech summarization pipeline. The system is orchestrated using **LangChain**, and runs ODLLMs in the GPT-Generated Unified Format (GGUF) [18]. The pipeline is explicitly designed to address the two key constraints of ODLLM: *limited context window size* and *restricted compute/memory budgets*. To overcome these limitations, we adopt a multi-stage summarization process that integrates chunk-wise refinement, optional retrieval augmentation, and structured output formatting.

Algorithm 1 Speech-to-Summary via Recursive Refinement

Input: Audio waveform $\mathbf{x}$; document set $\mathcal{D}$; chunk length L; overlap δ; vector DB $\mathcal{V}$; local LLM $\mathcal{S}_\theta$
Output: Structured summary $\mathbf{s}$

$\mathbf{t} \leftarrow \textsc{WhisperTranscribe}(\mathbf{x})$ `// speech-to-text`
$\mathcal{C} \leftarrow \textsc{Chunk}(\mathbf{t}, L, \delta)$ `// token chunks with overlap`
$\mathbf{s}_{\text{prev}} \leftarrow \varnothing$
foreach $c_i \in \mathcal{C}$ **do**
  $\mathbf{q}_i \leftarrow \textsc{GenerateQueries}(c_i)$ **if** $\mathcal{D} \neq \varnothing$ **then**
    $\mathcal{R}_i \leftarrow \text{ANN}(\mathbf{q}_i, \mathcal{V})$
  else
    $\mathcal{R}_i \leftarrow \varnothing$
  $\mathbf{s}_i \leftarrow \mathcal{S}_\theta\big(c_i \oplus \mathbf{s}_{\text{prev}}, \mathcal{R}_i\big)$ $\mathbf{s}_{\text{prev}} \leftarrow \mathbf{s}_i$
$\mathbf{s} \leftarrow \textsc{MergeSummaries}\big(\{\mathbf{s}_i\}\big)$ **return** $\mathbf{s}$

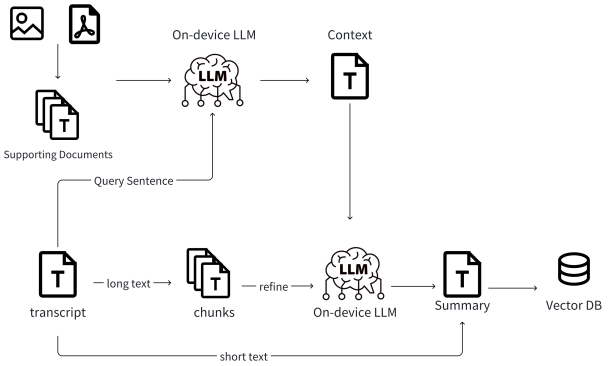

Fig. 2. Text Summarization Workflow

The pipeline begins with an audio waveform $\mathbf{x}$, which is transcribed using Whisper-`tiny` into a token sequence $\mathbf{t} = (w_1, w_2, \ldots, w_{|T|})$. In parallel, if users upload supporting documents—such as images, PDFs, or scanned pages—they are parsed, chunked, and embedded into a local vector store $\mathcal{V}$ via ODLLM-compatible embedding models. These documents are optional but can enrich summarization quality through retrieval-augmented reasoning.

To handle the long and potentially noisy transcript $\mathbf{t}$ under limited context capacity, we implement a recursive refinement mechanism. The transcript is first segmented into overlapping chunks using `chunk_size` $L = 2048$ and `chunk_overlap` $\delta = 200$ tokens. For each chunk c_i, the ODLLM conducts a lightweight preliminary analysis to identify ambiguous or incomplete content. Based on this analysis, it generates clarification-style queries, which are used to retrieve relevant passages from the vector store $\mathcal{V}$ via Approximate Nearest

Neighbor (ANN) search. The model then summarizes the enriched context consisting of c_i and the retrieved evidence, while also conditioning on the previous chunk's summary $\mathbf{s}_{\text{prev}}$ as auxiliary context. This recursive structure ensures that the final output is both locally coherent and globally consistent. When no supporting documents are available, the retrieval step is skipped, and summarization is conducted solely based on the transcript. This conditional design provides adaptability to different use cases while preserving computational efficiency on resource-constrained devices. Each final output is serialized into a structured summary $\mathbf{s}$ as defined below:

Definition 1 (Summary Schema)**.** *A summary instance is a tuple* $\mathbf{s} = (s_{\text{set}}, s_{\text{top}}, \mathbf{s}_{\text{key}}, s_{\text{body}})$ *where* $s_{\text{set}}, s_{\text{top}}, s_{\text{body}} \in \Sigma^*$ *and* $\mathbf{s}_{\text{key}} = \{k_1, \ldots, k_m\}$ *with* $k_i \in \Sigma^*$*. The four fields capture* setting, topic, key terms*, and the main* body *respectively, facilitating downstream indexing and retrieval.*

Each field in the `Summary` schema serves a specific purpose in structuring the summarization output:

- **setting**: Describes the context or environment in which the original audio took place, such as a classroom lecture, team meeting, interview, or informal conversation. This helps users quickly understand the situational background of the transcript.
- **topic**: Identifies the main subject or focus of the discussion. This field provides a high-level overview of the content and can aid in indexing or organizing summaries across multiple sessions.
- **key_terms**: A list of important terms, concepts, or named entities extracted from the content. These serve as concise indicators of what the conversation is about and support search or tagging functionalities.
- **summary**: A concise but informative textual summary that captures the key points, outcomes, or insights from the original transcript. It is the primary output intended for human consumption.

This structured format not only improves the readability and usability of summaries but also facilitates integration into downstream components such as retrieval systems or user dashboards.

Finally, the structured summary $\mathbf{s}$ is embedded and indexed into the local vector database. This allows the system to support retrieval-augmented generation (RAG) for future question-answering tasks, even when the original transcript or audio is no longer available. By embedding high-quality summaries rather than raw input, the system achieves fast and lightweight QA without compromising relevance. The Speech-to-Summary via Recursive Refinement process is summarized as in Algorithm 1.

C. RAG-Based Question Answering Module

The final stage of the system enables users to interact with previously summarized content via a RAG module for question answering. This component elevates the system from a one-way summarization tool to an interactive assistant capable of generating context-aware answers grounded in stored summaries.

When a user submits a query, the system first embeds the question using the same embedding model employed during summarization. It then performs a semantic similarity search over the Chroma vector database, which contains the structured summaries $\{\mathbf{s}_i\}$ generated and indexed in earlier sessions. If relevant matches are found, the system extracts the most pertinent portions of the summaries and packages them as contextual input. This retrieved context, combined with the user's query, is passed to a lightweight local language model executed via ODLLM. The model generates an answer that is both relevant and grounded in prior content. If no sufficiently similar summary is found, the system gracefully falls back to a default response indicating that no answer is currently available.

The RAG module is designed to operate entirely on-device, ensuring that user data—including both queries and summaries—remains local and private. This architecture supports flexible post-hoc access to previously captured knowledge without requiring reprocessing of transcripts or audio. Typical use cases include querying past lecture points, reviewing decisions from team meetings, or retrieving specific facts from long discussions. More broadly, this module demonstrates how structured summaries can serve not only as consumable outputs but also as persistent, queryable knowledge stores—bridging the gap between summarization and real-time interactive reasoning.

D. Web Platform Interface and User Flow

The VIBE web application provides an intuitive, step-by-step interface for users to transcribe, summarize, and interact with audio-based content. The user experience is divided into three main modules: **Transcript**, **Summary**, and **Ask**.

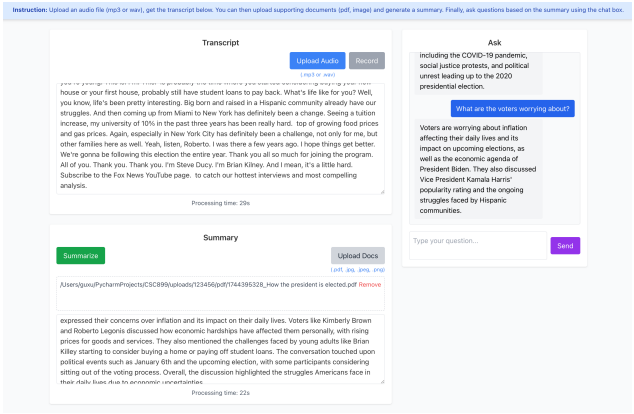

Fig. 3. VIBE User Page: Transcript, Summary and Intelligent Q&A

Transcript Module: Users begin by uploading an audio file (`.mp3` or `.wav`) or recording directly in the browser. The system uses Whisper to perform speech transcription, and the output is displayed in a scrollable text box. Below the transcription result, the system displays the processing time for transparency. **Summary Module**: After the transcript is generated, users can optionally upload supporting documents (e.g., `.pdf`, `.jpg`, `.png`) to enrich the context. By clicking the "Summarize" button, the transcript—along with any auxiliary

Model Name	Parameter Size (Billion)
ChatGPT-4o-latest	**200+**
llama3.2-1b	1
llama3.2-3b	3
gemma2-2b	2
qwen2.5-0.5b	0.5
qwen2.5-1.5b	1.5
qwen2.5-3b	3
opencoder-1.5b	1.5
smollm-1.7b	1.7
deepseek-r1-1.5b	1.5
tinyllama-1.1b	1.1
tinydolphin-1.1b	1.1
phi-2.7b	2.7
orca-mini-3b	3
hermes3-3b	3
stablelm-zephyr-3b	3
stablelm2-1.6b	1.6
granite3.1-dense-2b	2

TABLE I
PARAMETER SIZES OF ALL MODELS EVALUATED IN OUR SUMMARIZATION EXPERIMENTS.

documents—is passed to an ODLLM, which generates a concise summary. The summary is presented in a text area below, along with processing time for the summarization step. **Ask Module**: In the final stage, users can ask questions about the summarized content in a conversational format. The question is answered by the same language model, based on the summary and uploaded documents. The chat interface supports dynamic, follow-up queries, providing an interactive and intelligent Q&A experience.

This structured, modular flow enables end users—such as students, researchers, or meeting participants—to quickly extract meaningful insights from spoken content using lightweight, local models without relying on cloud-based services.

IV. SIMULATION RESULTS

To identify the most effective ODLLM model for VIBE, a total of 17 language models with parameter sizes under 3 billion were selected as candidates, as listed in Table I [6]. Each model was assessed using two different measure models—**ChatGPT-4o-latest** and **LLaMA3.3-70B**—to ensure robustness and reduce evaluation bias [19] [20]. We used the publicly available **BBC News Summary Dataset** [21]. This dataset consists of professionally written news articles across five distinct categories: **Business**: 510 articles, **Entertainment**: 386 articles, **Politics**: 417 articles **Sport**: 511 articles, **Tech**: 401 articles. In total, the dataset contains **2,225 articles**, each accompanied by a corresponding human-written summary. These summaries serve as high-quality references for evaluating the completeness and conciseness of model-generated outputs.

A. Performance of Individual Models

Table II presents the average scores of all models across the three dimensions under both evaluators. From the table, we can observe that `ChatGPT-4o-latest` (used here as a candidate model) achieved the highest scores in all dimensions under both evaluations. Among on-device models, `gemma2:2b`, `llama3.2:3b`, and `stablelm-zephyr:3b` consistently performed well, with relatively high scores across all metrics.

By contrast, smaller models such as `tinydolphin:1.1b` and `tinyllama:1.1b` scored lower, especially in faithfulness, highlighting potential factual consistency issues. A notable finding is that some models, such as `granite3.1-dense:2b` and `phi:2.7b`, showed strong performance under one evaluator but moderate performance under the other, reflecting how different LLMs may interpret or detect errors differently.

B. Quantitative Comparison and Analysis

To better understand model performance from different perspectives, we analyzed the top-performing models under each evaluation metric (faithfulness, completeness, and conciseness) based on rankings by both measure models. Table III summarizes the top one-third of models (top six out of eighteen) under each dimension and evaluator.

Faithfulness. As expected, `ChatGPT-4o-latest` itself ranks first under both evaluators, with a significant lead when evaluated by its own model. Other strong models include `gemma2:2b`, `granite3.1-dense:2b`, and `stablelm-zephyr:3b`, which appear in both rankings. `qwen2.5:3b` is ranked fifth by ChatGPT-4o-latest and seventh by LLaMA3.3, still indicating consistent quality. In contrast, `hermes3:3b` ranks second under LLaMA3.3-70B but drops to ninth under ChatGPT-4o-latest, reflecting some divergence in error detection strictness between evaluators.

Completeness: Remarkably, the top one-third models under this metric are identical for both evaluators. `granite3.1-dense:2b` even slightly outperforms ChatGPT-4o-latest, ranking first in both cases. Additionally, `opencoder:1.5b` and `smollm:1.7b`, though smaller in size, demonstrate strong key fact coverage, suggesting their summaries retain more information, albeit with longer lengths.

Conciseness: The conciseness ranking is also highly consistent across evaluators, with five out of six models appearing in both lists. Notably, `hermes3:3b` ranks first in both, often generating extremely short summaries, sometimes consisting of a single sentence. While this boosts conciseness, it significantly harms completeness—`hermes3:3b` ranks last in completeness under both evaluators.

Cross-metric analysis. Among all candidates, `stablelm-zephyr:3b` is the only on-device model that appears in all six top-1/3 lists. This suggests it maintains a good balance between factuality, informativeness, and brevity. Conversely, models like `smollm:1.7b` and `opencoder:1.5b` rank high in completeness but perform poorly in conciseness, as they tend to generate lengthy and sometimes repetitive summaries. `deepseek-r1:1.5b` exhibits abnormal behavior by including internal reasoning steps in the output, despite explicit instructions to avoid that—an issue not observed in other models.

Remark on completeness scores: One notable observation is that all models, including the strongest on-device ones, exhibit relatively low completeness scores. This may be attributed to

TABLE II
AVERAGE SCORES OF 18 CANDIDATE MODELS ACROSS THREE DIMENSIONS, EVALUATED BY CHATGPT-4O-LATEST AND LLAMA3.3-70B.

Model	Faithfulness		Completeness		Conciseness	
	GPT-4o	LLaMA3.3	GPT-4o	LLaMA3.3	GPT-4o	LLaMA3.3
llama3.2:1b	0.4343	0.6245	0.2997	0.3394	0.7663	0.7266
llama3.2:3b	0.5357	0.8069	0.3152	0.3506	0.8408	0.8130
gemma2:2b	0.6203	0.8156	0.3337	0.3664	0.7172	0.6696
qwen2.5:0.5b	0.2808	0.4128	0.2828	0.3269	0.6856	0.6153
qwen2.5:1.5b	0.3032	0.5904	0.2951	0.3322	0.8041	0.7682
qwen2.5:3b	0.6063	0.7670	0.4711	0.4942	0.6836	0.6558
opencoder:1.5b	0.5005	0.5455	0.4963	0.5149	0.5389	0.4893
smollm:1.7b	0.5982	0.6821	0.4201	0.4368	0.4931	0.4374
deepseek-r1:1.5b	0.3009	0.4400	0.3869	0.4196	0.6523	0.6065
tinyllama:1.1b	0.2013	0.3006	0.3627	0.4017	0.6625	0.6052
tinydolphin:1.1b	0.2565	0.3800	0.2991	0.3452	0.6044	0.5333
phi:2.7b	0.5151	0.6788	0.3691	0.4037	0.6786	0.6379
orca-mini:3b	0.6051	0.7584	0.3649	0.3967	0.6897	0.6489
hermes3:3b	0.5217	0.8757	0.2029	0.2378	**0.9385**	**0.9130**
stablelm-zephyr:3b	0.6665	0.8218	0.4312	0.4574	0.7151	0.6832
stablelm2:1.6b	0.4497	0.5883	0.3790	0.4121	0.6577	0.6174
granite3.1-dense:2b	0.6235	0.7792	**0.5767**	**0.5940**	0.6607	0.6374
chatgpt-4o-latest	**0.8691**	**0.9512**	0.5430	0.5610	0.6781	0.6598

several factors. First, the reference summaries often produce a large number of fine-grained key facts, while most generated summaries are considerably shorter. Second, the alignment process requires strict evidence of factual inference, making it difficult for short summaries to align with many key facts. Finally, the summarization prompts do not explicitly encourage exhaustive coverage, which naturally biases the models toward brevity rather than recall.

TABLE III
TOP ONE-THIRD MODELS UNDER EACH EVALUATION METRIC, AS RANKED BY CHATGPT-4O-LATEST AND LLAMA3.3-70B.

Metric	Rank by GPT-4o	Rank by LLaMA3.3-70B)
Faithfulness	1. ChatGPT-4o-latest 2. stablelm-zephyr:3b 3. granite3.1-dense:2b 4. gemma2:2b 5. qwen2.5:3b 6. orca-mini:3b	1. ChatGPT-4o-latest 2. hermes3:3b 3. stablelm-zephyr:3b 4. gemma2:2b 5. llama3.2:3b 6. granite3.1-dense:2b
Completeness	1. granite3.1-dense:2b 2. ChatGPT-4o-latest 3. opencoder:1.5b 4. qwen2.5:3b 5. stablelm-zephyr:3b 6. smollm:1.7b	1. granite3.1-dense:2b 2. ChatGPT-4o-latest 3. opencoder:1.5b 4. qwen2.5:3b 5. stablelm-zephyr:3b 6. smollm:1.7b
Conciseness	1. hermes3:3b 2. llama3.2:3b 3. qwen2.5:1.5b 4. llama3.2:1b 5. stablelm-zephyr:3b 6. gemma2:2b	1. hermes3:3b 2. llama3.2:3b 3. qwen2.5:1.5b 4. llama3.2:1b 5. gemma2:2b 6. stablelm-zephyr:3b

C. Case Study: FDA and COX-2 Inhibitors

To complement the quantitative analysis in Section 4.4, we present a detailed case study based on a real-world article concerning Parmalat, an Italian dairy company. This example offers a closer look at how different models behave under the evaluation framework and further supports several key observations from earlier sections, such as the trade-off between conciseness and completeness, and variance in factual consistency. Since ChatGPT-4o-latest and LLaMA3.3-70B yield largely consistent evaluation results (see Section 4.4), we only analyze evaluations conducted by ChatGPT-4o-latest in this section for clarity.

We examine summaries from five representative models: ChatGPT-4o-latest, stablelm-zephyr:3b, smollm:1.7b, hermes3:3b, opencoder:1.5b, and deepseek-r1:1.5b.

This case study focuses on an article reporting on FDA's regulatory decision around COX-2 inhibitors, such as Vioxx, and its implications on pharmaceutical companies. The article discusses Merck's legal and financial risks, potential drug market returns, and the FDA's structural response.

a) ChatGPT-4o-latest (candidate): ChatGPT's summary captures the main developments: Vioxx's heart risk, Merck's legal liabilities, possible market returns, and the FDA's internal changes. All sentences are factually accurate, and 9 out of 10 key facts are correctly aligned. This again establishes the benchmark performance.

b) stablelm-zephyr:3b: Stablelm-zephyr:3b generates a balanced and coherent summary that includes key points about Vioxx's risks, the FDA panel vote, and Merck's legal exposure. It aligns with 8 of the 10 key facts and contains no major factual inaccuracies. Its summary is appropriately concise, making it one of the better-performing on-device models in this case. This supports its consistent appearance in the top-performing group across all metrics in Section 4.4.

c) smollm:1.7b: Smollm generates a very long and repetitive summary, restating the FDA panel vote and the risks of Vioxx multiple times. While it includes all key information and aligns with 9 key facts, it also introduces minor inconsistencies and irrelevant phrasing. This mirrors the pattern observed in Section 5.4, where smollm achieves high completeness but low conciseness.

d) opencoder:1.5b: Opencoder also produces an overly long summary, reiterating content and including generic statements. It aligns with 8 key facts but introduces minor factual issues (e.g., misstating the degree of panel support). Its conciseness score is low, and its faithfulness is degraded by one "out-of-context error." This supports the previous quantitative analysis that opencoder tends toward verbosity.

e) hermes3:3b: Hermes3 outputs a single-sentence summary: *"FDA panel supports return of painkillers linked to heart risks."* While concise and grammatically clean, it aligns with only 3 out of 10 key facts and lacks crucial detail about Merck, legal risks, and regulatory changes. This continues the pattern of extremely high conciseness but poor completeness.

f) deepseek-r1:1.5b: As in previous cases, Deepseek's summary includes reasoning-like preamble phrases such as *"I need to summarize this carefully"*. Although some content is correct and 7 key facts are matched, this stylistic violation, along with minor factual errors, results in lower faithfulness. This again confirms the unique behavioral flaw discussed in Section 4.4.

This example further validates several trends noted in earlier sections. Hermes3 over-optimizes for brevity, Smollm and Opencoder struggle with length control, and Deepseek includes unintended reasoning text. Stablelm-zephyr demonstrates a strong balance between informativeness and succinctness. Models like ChatGPT and Stablelm-zephyr achieve more balanced performance, confirming their high rankings in both quantitative and qualitative assessments. These behavior-level observations provide context for the performance results and further justify the model rankings derived from empirical evaluation. They also offer practical guidance for model selection in resource-constrained applications where specific summary traits (e.g., brevity vs. coverage) are desired.

V. CONCLUSIONS

We have introduced a fully on-device pipeline for speech summarization and retrieval-augmented question answering using Whisper and lightweight ODLLMs. By combining recursive refinement, conditional retrieval, and a structured summary schema, the system operates under tight resource constraints while maintaining strong content fidelity. Unlike cloud-based solutions, our design prioritizes privacy and latency, making it suitable for use in settings such as meetings, lectures, and mobile devices. Through chunk-based processing and local vector search, the system is able to mitigate ODLLM limitations such as small context windows and limited memory capacity. Furthermore, the structured output format not only improves readability but also enables persistent semantic indexing for future QA interactions. This work highlights a practical pathway toward conversational intelligence at the edge. Future directions include adaptive chunking based on discourse boundaries, hybrid cloud–edge inference, and expansion to multilingual and multimodal summarization scenarios.

REFERENCES

[1] Picovoice, "On-device speech recognition with cloud quality," 2023. [Online]. Available: https://picovoice.ai/blog/on-device-speech-recognition/

[2] L. Li, S. Qian, J. Lu, L. Yuan, R. Wang, and Q. Xie, "On-device query intent prediction with lightweight llms to support privacy-sensitive applications," *Scientific Reports*, vol. 14, no. 1, p. 63380, 2024.

[3] N. Hao, Y. Li, K. Liu, S. Liu, Y. Lu, B. Xu, C. Li, J. Chen, L. Yue, T. Fu *et al.*, "Artificial intelligence-aided digital twin design: A systematic review," 2024.

[4] J. Xu, Q. Wang, Y. Cao, B. Zeng, and S. Liu, "A general purpose device for interaction with llms," in *Proceedings of the Future Technologies Conference*. Springer, 2024, pp. 613–626.

[5] S. Gunasekar *et al.*, "The phi-1 and phi-2 language models: Small models that perform well," *Microsoft Research*, 2023, https://www.microsoft.com/en-us/research/project/phi-1-and-phi-2/.

[6] J. Xu, Z. Li, W. Chen, Q. Wang, X. Gao, Q. Cai, and Z. Ling, "On-device language models: A comprehensive review," *arXiv preprint arXiv:2409.00088*, 2024.

[7] H. Shang, Z. Li, J. Guo, S. Li, Z. Rao, Y. Luo, D. Wei, and H. Yang, "An end-to-end speech summarization using large language model," *arXiv preprint arXiv:2407.02005*, 2024.

[8] A. Raj, M. Raj, N. Umasankari, and D. Geethanjali, "Document-based text summarization using t5 small and gtts," in *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, 2024, pp. 1–6.

[9] S. Verma, Q. Wang, and E. W. Bethel, "Intelligent iot attack detection design via odllm with feature ranking-based knowledge base," in *Proceedings of the AAAI Symposium Series*, vol. 5, no. 1, 2025, pp. 188–195.

[10] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, 2004, pp. 74–81.

[11] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[12] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," in *International Conference on Learning Representations (ICLR)*, 2020.

[13] W. Zhao, M. Peyrard, F. Liu, Y. Gao, C. M. Meyer, and I. Gurevych, "Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019, pp. 563–578.

[14] Y. Song, Z. Wang, Y. Liu, Y. Zhang, and R. Wang, "Finesure: Fine-grained summarization evaluation using llms," *arXiv preprint arXiv:2407.00908*, 2024. [Online]. Available: https://arxiv.org/abs/2407.00908

[15] C. Graham and N. Roll, "Evaluating openai's whisper asr: Performance analysis across diverse accents and speaker traits," *JASA Express Letters*, vol. 4, no. 2, 2024.

[16] Y. Lu, M. Shen, H. Wang, X. Wang, C. van Rechem, T. Fu, and W. Wei, "Machine learning for synthetic data generation: a review," *arXiv preprint arXiv:2302.04062*, 2023.

[17] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, "Common voice: A massively-multilingual speech corpus," *arXiv preprint arXiv:1912.06670*, 2019.

[18] A. Kumar, S. Sharma, S. Gupta, and D. Kumar, "Mental healthcare chatbot based on custom diagnosis documents using a quantized large language model," in *2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2024, pp. 1–6.

[19] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[20] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[21] S. Yunus, C. Hark, and F. Okumuş, "Comparison of extractive and abstractive approaches in automatic text summarization: An evaluation on bbc-news and pubmed datasets," in *2024 8th International Artificial Intelligence and Data Processing Symposium (IDAP)*. IEEE, 2024, pp. 1–7.